Context-aware recommender systems for real-world applications

Marie Al-Ghossein

▶ **To cite this version:**

Marie Al-Ghossein. Context-aware recommender systems for real-world applications. Artificial Intelligence [cs.AI]. Université Paris Saclay (COmUE), 2019. English. NNT : 2019SACLT008 . tel-02275811

HAL Id: tel-02275811
https://pastel.hal.science/tel-02275811

Submitted on 2 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université PARIS-SACLAY

NNT : 2019SACLT008

Thèse de doctorat

TELECOM ParisTech

Context-Aware Recommender Systems for Real-World Applications

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom ParisTech

Ecole doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Paris, le 11 février 2019, par

MARIE AL-GHOSSEIN

Composition du Jury :

Talel Abdessalem Professeur, Télécom ParisTech	Directeur de thèse
Anthony Barré Directeur Data, JCDecaux SA	Examinateur
Dario Colazzo Professeur, Université Paris-Dauphine	Rapporteur
Antoine Cornuéjols Professeur, AgroParisTech	Président
Noam Koenigstein Chercheur, Microsoft	Examinateur
Nuria Oliver Directrice de recherche, Vodafone	Examinateur
Fabrice Rossi Professeur, Université Paris 1 Panthéon-Sorbonne	Rapporteur

Abstract

Recommender systems have proven to be valuable tools to help users overcome the information overload, and significant advances have been made in the field over the last two decades. In particular, contextual information has been leveraged to model the dynamics occurring within users and items. Context is a complex notion and its traditional definition, which is adopted in most recommender systems, fails to cope with several issues occurring in real-world applications. In this thesis, we address the problems of *partially observable* and *unobservable* contexts in two particular applications, hotel recommendation and online recommendation, challenging several aspects of the traditional definition of context, including accessibility, relevance, acquisition, and modeling.

The first part of the thesis investigates the problem of hotel recommendation which suffers from the continuous cold-start issue, limiting the performance of classical approaches for recommendation. Traveling is not a frequent activity and users tend to have multifaceted behaviors depending on their specific situation. Following an analysis of the user behavior in this domain, we propose novel recommendation approaches integrating *partially observable* context affecting users and we show how it contributes in improving the recommendation quality.

The second part of the thesis addresses the problem of online adaptive recommendation in streaming environments where data is continuously generated. Users and items may depend on some *unobservable context* and can evolve in different ways and at different rates. We propose to perform online recommendation by actively detecting drifts and updating models accordingly in real-time. We design novel methods adapting to changes occurring in user preferences, item perceptions, and item descriptions, and show the importance of online adaptive recommendation to ensure a good performance over time.

Acknowledgements

This thesis was accomplished with the help of many amazing people that I would like to thank from the bottom of my heart.

Thank you to my supervisor Talel, for your wise guidance, for your valuable advices, for your support during these years, and for always lifting me up when I needed it. This whole adventure started on a day of spring when I entered your office with a few questions. If it were not for our discussion then, I would have been on a completely different path.

Thank you to my (other) supervisor Anthony, for making me *feel welcome* at Accor and for ensuring that I have an excellent work environment. Most of all, I am grateful for your continuous presence until the very end and for not leaving me hanging when circumstances suggested so. Thank you for always being available, positive, thoughtful, and encouraging.

Thank you to my reviewers, Dario Colazzo and Fabrice Rossi, for taking the time to review my thesis and for providing me with constructive feedback. Thank you to the members of my jury, Antoine Cornuéjols, Noam Koenigstein, and Nuria Oliver, for doing me the honor of being part of it.

Thank you to the entire DIG (ex-DBWeb) team, always gathering amazing people, for setting an incredible and welcoming atmosphere. I have had the pleasure to spend my days at Télécom with you and have learned so much from our discussions and from our diversity. Thank you to Albert, Antoine, Atef, Fabian, Jacob, Jean-Benoit, Jean-Louis, Jonathan, Julien, Luis, Marc, Mauro, Maximilien, Mikael, Ned, Oana, Pierre, Thomas, and Thomas. Special thanks to Maroua, our sweetest footballer; to Pierre-Alexandre: even though I keep on repeating that you are an old tourist, you are still my youngest co-author with whom I had the craziest deadlines; to Quentin for taking charge of being our sociologist; and to Ziad, my compatriot, for your help and for our numerous discussions.

Thank you to the colleagues at AccorHotels and to the Data Science team that took too many different forms since my first day there, the only constant thing being the fun and pleasant atmosphere that was reigning. Thank you for the efforts you all did to keep me involved and up-to-date, while filling me in on the exciting news that were happening during my absence. Special thanks to Agnès, Allan, Amaury, Assitan, Julien, Matthieu, Nicolas, Romain, Vincent, and Zyed.

Far away from the world of research, I am very lucky to have incredible friends that have been filling my life since forever or only more recently. I do not say it much, which cost me my reputation, but I am extremely grateful for your devotion, for all of your efforts, and for always being here for me without me even asking. Special thanks to my date every Friday night, to my best shortfriend, to the one who was bad influence during our youth, to our tourist guide in our own hometown, and to my neighbor during my first few years in Paris. Special thanks also to those who took a day off, missed welcoming their first intern, or even had to cross the sea to attend my defense.

Thank you to my amazing mentors who have guided me in accomplishing things outside the world of science. You taught me so much, contributed in making me who I am today, and were a great source of inspiration.

Thank you to all of my relatives, mainly spread across the US, France, Lebanon, and Dubai, for showering me with love and for sharing so many unforgettable moments. We have followed different paths but we have always stayed close in our hearts. Those who left us have had the biggest contributions and will never be forgotten.

Most importantly, thank you to my parents, Eliane and Imad. You have been my rock since day one and the greatest role models. Watching you excel in your respective domains has always inspired me to be a better person. Thank you for teaching me strong values, for encouraging me to set high ambitions, and for always being there for me even when it involves getting on the plane as many times as needed. Thank you for the privilege of a fulfilled life and for sacrificing yours in the process. You have gifted me with the three treasures I cherish most, Najib, Nabil, and Rima. They have always been looking over my shoulder, even when I did not deserve it, and provided tremendous emotional support. You are all source of an immeasurable joy in my life.

Contents

List of Figures xi

List of Tables xiii

List of Abbreviations xv

I Introduction and Background 1

1 Introduction 3
1.1 Context in Real-World Recommender Systems 4
1.2 Contributions 7
1.3 Organization of the Thesis 9
1.3.1 Publications 10

2 Recommender Systems 13
2.1 The Recommendation Problem 14
2.1.1 Problem Formulation 14
2.1.2 Types of Feedback 15
2.1.3 Challenges and Limitations 15
2.1.4 Data Representation and Notations 17
2.2 Evaluation of Recommender Systems 19
2.2.1 Evaluation Methods 19
2.2.1.1 Offline Evaluation 19
2.2.1.2 User Studies 20
2.2.1.3 Online Evaluation 20
2.2.2 Evaluation Criteria 21
2.2.3 Evaluation Metrics 22
2.3 Recommendation Approaches 24
2.4 Content-Based Filtering Approaches 24
2.4.1 Example of a Content-Based Filtering Approach 25
2.4.2 Advantages and Disadvantages 26
2.4.3 Related Recommendation Approaches 26
2.5 Collaborative Filtering Approaches 27
2.5.1 Memory-Based Approaches 28

2.5.1.1 User-Based Collaborative Filtering . . . 28
2.5.1.2 Item-Based Collaborative Filtering . . . 29
2.5.1.3 Extensions, Complexity, Advantages and Disadvantages . . . 29
2.5.2 Matrix Factorization Approaches . . . 30
2.5.2.1 Matrix Factorization Framework . . . 30
2.5.2.2 Singular Value Decomposition . . . 31
2.5.2.3 Minimizing Squared Loss and Other Loss Functions . . . 32
2.5.2.4 Dealing with Implicit Feedback . . . 36
2.5.2.5 Probabilistic Models . . . 38
2.6 Hybrid Approaches . . . 39
2.7 Context-Aware Approaches . . . 39
2.7.1 Paradigms for Incorporating Context . . . 41
2.7.2 From Context-Aware to Context-Driven Recommender Systems . . . 42
2.8 Conclusion . . . 42

II Partially Observable Context in Hotel Recommendation 43

3 The Hotel Recommendation Problem 45
3.1 Introduction . . . 45
3.2 Scope of Our Work . . . 46
3.3 Comparison with Recommendation in Other Domains . . . 50
3.4 Challenges and Limitations . . . 52
3.5 Related Work . . . 53
3.6 Context in the Hotel Domain . . . 56
3.7 Conclusion . . . 58

4 Leveraging Explicit Context 59
4.1 Introduction . . . 59
4.2 Influence of the Physical Context . . . 60
4.3 Influence of the Social Context . . . 62
4.3.1 Collaborative Topic Modeling . . . 63
4.3.2 Handling Positive and Negative Reviews . . . 65
4.4 Influence of the Modal Context . . . 66
4.5 Overview of the System . . . 68
4.6 Experimental Results . . . 68
4.6.1 Contribution of the Physical Context . . . 69
4.6.2 Contribution of the Social and Modal Contexts . . . 70
4.6.3 User Segmentation and Performance . . . 72
4.7 Conclusion . . . 75

5 Leveraging Implicit Context 77
5.1 Introduction . . . 77
5.2 Event Recommendation . . . 79
5.3 Problem Formulation . . . 80

5.4 Data Collection and Analysis . . . 81
5.4.1 Event Dataset . . . 81
5.4.2 Booking Dataset . . . 84
5.5 Proposed Framework . . . 84
5.5.1 Overview . . . 84
5.5.2 Notations and Definitions . . . 85
5.5.3 Modules . . . 85
5.5.3.1 Building All-Inclusive Profiles Based on Location and Time . 86
5.5.3.2 Measuring Events' Similarities . . . 87
5.5.3.3 Building Limited Profiles Based on Cohesiveness . . . 88
5.5.3.4 Learning Preferences for Hotels and Events . . . 89
5.5.3.5 Recommending Hotels and Events . . . 89
5.6 Experimental Results . . . 90
5.6.1 Qualitative Evaluation Through Concrete Examples . . . 90
5.6.2 Quantitative Evaluation Through Offline Experiments . . . 92
5.7 Discussion . . . 95
5.8 Conclusion . . . 96

6 Transferring Context Knowledge Across Domains 97
6.1 Introduction . . . 97
6.2 Cross-Domain Recommendation . . . 98
6.3 Proposed Approach . . . 101
6.3.1 Mapping Items from Both Domains . . . 102
6.3.2 Mapping Users from Both Domains . . . 104
6.3.3 Merging Preferences from Both Domains . . . 104
6.4 Experimental Results . . . 106
6.5 Conclusion . . . 109

III From Unobservable Context to Online Adaptive Recommendation 111

7 The Online Adaptive Recommendation Problem 113
7.1 Introduction . . . 113
7.2 Time Dimension in Recommender Systems . . . 115
7.3 Adaptive Data Stream Mining . . . 117
7.4 Online Adaptive Recommendation . . . 121
7.4.1 Memory Module . . . 121
7.4.2 Learning Module . . . 122
7.4.2.1 Incremental Memory-Based Approaches . . . 122
7.4.2.2 Incremental Matrix Factorization and Other Model-Based Approaches . . . 123
7.4.3 Retrieval Module . . . 126
7.4.4 Evaluation Module . . . 127
7.5 Conclusion . . . 129

8 Dynamic Local Models **131**
8.1 Introduction . . . 131
8.2 Local Models for Recommendation . . . 132
8.3 Proposed Approach . . . 134
8.4 Experimental Results . . . 136
8.5 Conclusion . . . 139

9 Adaptive Incremental Matrix Factorization **141**
9.1 Introduction . . . 141
9.2 Learning Rate Schedules for Matrix Factorization . . . 142
9.3 Proposed Approach . . . 146
9.4 Experimental Results . . . 148
9.4.1 Performance of AdaIMF on Synthetic Datasets . . . 149
9.4.2 Performance of AdaIMF on Real-World Datasets . . . 151
9.5 Conclusion . . . 157

10 Adaptive Collaborative Topic Modeling **159**
10.1 Introduction . . . 159
10.2 Related Work . . . 160
10.3 Proposed Approach for Online Topic Modeling . . . 162
10.4 Proposed Approach for Online Recommendation . . . 166
10.5 Experimental Results . . . 166
10.5.1 Performance of AWILDA for Online Topic Modeling . . . 169
10.5.1.1 Results for Topic Drift Detection . . . 170
10.5.1.2 Results for Document Modeling . . . 172
10.5.2 Performance of CoAWILDA for Online Recommendation . . . 172
10.6 Conclusion . . . 177

IV Concluding Remarks **179**

11 Conclusions and Future Work **181**
11.1 Summary and Conclusions . . . 181
11.2 Future Work . . . 183

A Résumé en français **185**
A.1 La notion de contexte dans les systèmes de recommandation du monde réel . 187
A.2 Contributions . . . 189
A.3 Organisation de la thèse . . . 192
A.3.1 Publications . . . 194

Bibliography **195**

List of Figures

2.1 Representation of the feedback matrix . . . 18
2.2 Representation of Matrix Factorization (MF) . . . 31
2.3 Graphical model for Probabilistic Matrix Factorization (PMF) . . . 38

3.1 Brands owned by AccorHotels [AHP, 2018] . . . 47
3.2 Example of an email received by *Le Club* customers containing promotional offers and hotel recommendations . . . 49
3.3 Percentage of users enrolled in the loyalty program per number of bookings made, represented by *filled bars*, and per number of distinct visited hotels, represented by *empty bars*, on a period of four years . . . 52
3.4 Internal and external factors influencing the traveler's decision. Figure adapted from [Horner and Swarbrooke, 2016]. . . . 57

4.1 Generative model for Latent Dirichlet Allocation (LDA) . . . 63
4.2 Answering Q4. Recall@10 and NDCG@10 of traditional recommendation approaches for the dataset AH-MAXI . . . 73
4.3 Answering Q5. Recall@10 and NDCG@10 of context-aware approaches per category of user for the dataset AH-TRIP . . . 74

5.1 Examples of hotel-centric and event-centric recommendations . . . 81
5.2 Architecture of the proposed framework for recommending hotels and events 86

6.1 Hierarchical structure used to map items from the source and target domains 103
6.2 Proposed approach for cross-domain recommendation considering the LBSN and the hotel domains . . . 105
6.3 Recall@5, NDCG@5, recall@10, and NDCG@10 for the dataset AH, represented with respect to the number of bookings present in the training set . . . 108

7.1 Concept drift forms represented over time. Figure adapted from [Gama et al., 2014]. . . . 119
7.2 A generic framework for online adaptive learning. Figure adapted from [Gama et al., 2014]. . . . 119

9.1 Experimental results for the dataset SYN_1 showing the behavior of `AdaIMF` . . . 150
9.2 MRR@m of `AdaIMF` and `FS-IMF` for the dataset SYN_2 . . . 150
9.3 Recall@N and DCG@N of `AdaIMF` and other variants and incremental methods for the dataset LASTFM, using different values of N, for $K = 50$ . . . 156

9.4 Recall@N and DCG@N of `AdaIMF` and other variants and incremental methods for the dataset Gowalla, using different values of N, for $K = 50$ 156
9.5 Recall@N and DCG@N of `AdaIMF` and other variants and incremental methods for the dataset Foursquare, using different values of N, for $K = 50$ . . 156

10.1 Topic drift detection for the dataset Sd_4 using `AWILDA` and its variants 170
10.2 Topic drift detection using `AWILDA` for synthetic and semi-synthetic datasets . 171
10.3 Performance evaluation of `AWILDA` and `OnlineLDA` for the task of document modeling using the measure of perplexity 173
10.4 DCG@m of `CoAWILDA` and other variants and incremental methods, where m is the number of available items. The evolution of DCG@m with the number of evaluated observations is reported. 175
10.5 Recall@5, recall@10, recall@50, and recall@100 of `CoAWILDA` and its variants for the dataset Ml-100k 176
10.6 Recall@5, recall@10, recall@50, and recall@100 of `CoAWILDA` and its variants for the dataset Plista 177

List of Tables

2.1 Common notations used in this thesis 18
2.2 Examples of MF models based on different objective functions 35

3.1 Proportion of available data by user feature for a set of customers enrolled in the loyalty program 48
3.2 Statistics of the booking dataset, Ah, used for the hotel recommendation problem 50

4.1 Proportion of bookings made by Australian residents in selected destinations during periods of one month 61
4.2 Proportion of bookings made by French residents in selected destinations during periods of one month 61
4.3 Examples of clusters of countries which residents have similar behaviors regarding visited destinations and periods of visit (non-exhaustive list per cluster) 62
4.4 Statistics of the booking datasets used in this chapter 69
4.5 Answering Q1. Recall@N and NDCG@N of the `Knni` and `BPR` methods under the `globalRS` and $\texttt{localRS}_\texttt{k}$ settings for the dataset Ah 70
4.6 Answering Q2. Recall@N and NDCG@N of `CTRk+` and other variants for the dataset Ah-Trip 71
4.7 Answering Q3. Recall@N and NDCG@N of `BPRx3` and other variants for the dataset Ah-Mini 72
4.8 Answering Q5. Recall@10 of the proposed recommendation methods for the dataset Ah-Trip represented per category of users 73
4.9 Answering Q5. NDCG@10 of the proposed recommendation methods for the dataset Ah-Trip represented per category of users 74

5.1 Examples of crawled events from Eventful, considered as major events susceptible of attracting travelers 82
5.2 Examples of crawled events from Eventful, considered as local events that do not have a major impact on travelers 83
5.3 Statistics of the crawled dataset, Event-Full, and the filtered dataset, Event 83
5.4 Statistics of the booking dataset Ah-Event used for recommending packages of hotels and events 84
5.5 Nearest neighbors of examples of venues 91
5.6 Nearest neighbors of examples of performers 91
5.7 Nearest neighbors of examples of events 92

5.8 Two examples of users' history of bookings, the association with events, and generated recommendations 94
5.9 Recall@N and NDCG@N for `MFev` and other recommendation approaches . . 94

8.1 Statistics of the real-world datasets used to evaluate DOLORES 136
8.2 Recall@N and DCG@N for `DOLORES`, its variants, and other methods for the dataset AH_EUR 137
8.3 Recall@N and DCG@N for `DOLORES`, its variants, and other methods for the dataset ML-1M+ 137
8.4 Recall@N and DCG@N for `DOLORES`, its variants, and other methods for the dataset ML-10M+ 138

9.1 Statistics of the real-world datasets used to evaluate AdaIMF 152
9.2 Performance of `AdaIMF` and its variants for the dataset AH-EUR for $K = 20$. 153
9.3 Performance of `AdaIMF` and its variants for the dataset ML-1M for $K = 20$. 154
9.4 Performance of `AdaIMF` and its variants for the dataset LASTFM for $K = 20$. 154
9.5 Performance of `AdaIMF` and its variants for the dataset GOWALLA for $K = 20$ 154
9.6 Performance of `AdaIMF` and its variants for the dataset FOURSQUARE for $K = 20$155

10.1 Statistics of the real-world datasets used to evaluate CoAWILDA 168

List of Abbreviations

AdaIMF	Adaptive Incremental Matrix Factorization
ADWIN	ADaptive WINdowing
ALS	Alternating Least Squares
AWILDA	Adaptive Window based Incremental Latent Dirichlet Allocation
BPR	Bayesian Personalized Ranking
CARS	Context-Aware Recommender System
CBF	Content-Based Filtering
CDRS	Context-Driven Recommender System
CF	Collaborative Filtering
CoAWILDA	Collaborative Adaptive Window based Incremental Latent Dirichlet Allocation
CTR	Collaborative Topic Regression
DCG	Discounted Cumulative Gain
DOLORES	Dynamic Local Online REcommender System
EBSN	Event-Based Social Networks
FM	Factorization Machines
IDF	Inverse Document Frequency
IMF	Incremental Matrix Factorization
IT	Information Technology
LBSN	Location-Based Social Networks
LDA	Latent Dirichlet Allocation
MAE	Mean Absolute Error
MF	Matrix Factorization
MRR	Mean Reciprocal Rank
MSE	Mean Squared Error
NDCG	Normalized Discounted Cumulative Gain

PMF	Probabilistic Matrix Factorization
POI	Point Of Interest
RMSE	Root Mean Squared Error
RS	Recommender System
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
TARS	Time-Aware Recommender System
TF	Term Frequency
TnF	Tensor Factorization
WRMF	Weighted Regularized Matrix Factorization

Part I

Introduction and Background

Chapter 1

Introduction

In the early 1960s, a number of works related to selective dissemination of information emerged, where intelligent systems were designed to filter streams of electronic documents according to individual preferences [Hensley, 1963]. These systems leveraged explicit document information such as keywords, to perform filtering. In the next decades, with the increasing use of emails and in an attempt to control the flood of information and filter out spam emails, the Tapestry email filtering system [Goldberg et al., 1992] was developed. The novelty of Tapestry relied in leveraging opinions given by all users to benefit each one of them, which proved to be a powerful approach. These early efforts paved the way to a category of systems that were later referred to as Recommender Systems (RS) [Resnick and Varian, 1997].

In their most general form, RS are used to recommend various types of items to users by filtering the most relevant ones. Such items can include products, books, movies, news articles, and friends, among others. Marketing studies, highlighting the multiple benefits of product recommendations, accompanied the first technical advances related to RS [West et al., 1999]. Their main advantage relies on helping users overcome the information overload in domains where the catalog of items is enormous, thus improving the user experience and satisfaction. The potential of increasing user loyalty and sales volume attracted online services that raced to implement RS in an attempt to boost their performance. One of the early adopters was Amazon.com which reported the huge impact RS had on its business [Linden et al., 2003]. This success story motivated many other players to apply the concepts in their respective domains [Bennett and Lanning, 2007; Celma, 2010; Mooney and Roy, 2000] and fueled research in the field.

The research community has been developing ideas and techniques for RS ever since, combining multi-disciplinary efforts from various neighboring areas such as artificial intelligence, data mining, and human computer interaction. The core element of a RS is the recommender algorithm that offers recommendations by estimating items' relevance for each user and can be seen as performing a prediction task [Adomavicius and Tuzhilin, 2005]. Past user behavior, collected under multiple forms and assumed to exhibit user preferences, is analyzed and then exploited to predict items' relevance.

Non-stationarity in RS. Looking at the general problem of learning from data and building predictive models [Mitchell, 1997], the main assumption made is that observations that will be generated in the future follow similar patterns to observations previously collected within the same environment: Data observed overall is expected to be generated by the same distribution. In the scope of RS, this implies that previously recorded observations can help in predicting future behavior, provided that user interests and item perceptions are consistent over time. In a dynamic world where users and items are constantly evolving and being influenced by many varying factors [Koren, 2009], this assumption does not hold.

Motivating example. Imagine Alice to be a tourist visiting the city of Paris. This is not her first visit to the city as she traveled there for business before. While she usually books a hotel near her company's offices, she would prefer a hotel located in downtown this time in order to be closer to the multiple touristic sites. Alice usually likes to wander around the streets and admire the beautiful buildings. However, since the weather channels are forecasting heavy rains then, she will most probably end up visiting various museums. On Friday night, she is meeting Carl, an old friend of hers. Their ritual over the past few years consisted in eating street food and hanging around in a park. Yet, since both of them have high-paying jobs now, they can afford dinner at a fancy restaurant. Imagine now the assistant that is supposed to help Alice organize the trip. Based on her past behavior, the assistant would recommend a hotel next to *La Défense*, a walking tour of the *Champs-Élysées* under the rain, and a hot-dog stand by the *Jardin des Tuileries*. However, an ideal assistant is expected to take into account *contextual factors* that would impact Alice's preferences, e.g., trip's intent, weather, and social status, in order to deliver relevant suggestions.

Context-Aware RS (CARS). The notion of *situated actions* [Suchman, 1987], taking into account that preferences may differ based on the context, has led to the development of *Context-Aware RS (CARS)* [Adomavicius and Tuzhilin, 2015]. These RS incorporate contextual information and tailor recommendations to specific circumstances. Research on CARS began in the early 2000s with a series of work showing the interest of using context in RS and applying them in several domains [Hayes and Cunningham, 2004; Van Setten et al., 2004]. Nowadays, CARS cover a wide range of paradigms and techniques, and are subject to multiple classifications. They also intersect with other families of RS [Campos et al., 2014; Cantador et al., 2015], but most importantly they consider, under all their forms, the dynamics existing in user-generated data due to multiple factors. The main challenges around CARS concern, first, understanding and modeling the notion of context, and second, developing algorithms that incorporate this notion. While the second challenge strongly depends on the first one, several limitations related to context understanding and modeling persist in today's existing solutions.

1.1 Context in Real-World Recommender Systems

Context is a complex notion that has been studied across different research disciplines [Adomavicius and Tuzhilin, 2015]. The definition introduced by [Dey, 2001] has been widely adopted for CARS and states the following: "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application.*". Without loss

of generality, the broad notion of context can be seen as a set of various relevant factors. Several efforts have been made to model and represent these factors in CARS, and most CARS proposed in the literature adopt relatively similar concepts that are presented in the following.

Context definition. The *representational* view of context, introduced by [Dourish, 2004], is the standard approach to defining context in CARS. It assumes that contextual factors are represented by a predefined set of observable attributes with a structure that does not change over time and values that are known *a priori*. In contrast, the *interactional* view defines context as a relational property held between activities and determined dynamically: It is not possible to enumerate relevant contextual factors before the user activity arises.

Context integration. While contextual factors can be of various types, it is often assumed that attribute values are nominal and atomic, e.g., "*family*" or "*colleagues*" for the factor *company* and "*summer*" or "*winter*" for the factor *season*. There are actually two data models that have been extensively used to represent context alongside users and items in CARS: the *hierarchical* model and the *multidimensional dataspace* model. Following the hierarchical model [Palmisano et al., 2008], context is modeled as a set of contextual factors and each factor as a set of attributes having a hierarchical structure. Each level of the hierarchy defines a different level of granularity with regards to the contextual knowledge. On the other hand, the multidimensional dataspace model [Adomavicius et al., 2005] considers the Cartesian product of contextual factors where each factor is again the Cartesian product of one or more attributes. Each situation is described by the combination of attribute values for all attributes of all factors.

Limitations in real-world RS. While it is extensively adopted in CARS, the representational view of context fails to address several issues occurring in real-world applications which remain unexplored. In addition, there exists a gap between the traditional context modeling in CARS and the context as emerging in real-world RS, making previously proposed CARS insufficient. This gap is related to multiple aspects of context, arises at different levels for any considered factor, and is described in the following:

- **Context accessibility.** The representational view assumes that contextual factors affecting the user behavior are explicitly observed by the system. This is naturally not always the case, especially when dealing with factors that cannot be easily accessible like the user's mood or intent. Moreover, in some cases, context is only revealed once the action is achieved, not before nor independently of its occurrence. A classification of approaches to represent context that goes beyond the representational and interactional views was presented in [Adomavicius et al., 2011] and is based on the aspect of observability, i.e., what the RS knows about the contextual factors, their structure, and their values. This knowledge falls into one of the three following categories: *fully observable*, *partially observable*, and *unobservable*. The representational view of context considers the fully observable setting while the two others were not thoroughly studied.

- **Context relevance.** Context is modeled as a multidimensional variable where all dimensions, referring to the multiple factors, are treated equally. Since generic recommendation methods do not integrate domain knowledge, all contextual factors are

assumed to affect the user behavior in the same way. In reality, this is not always valid given that users tend to prioritize some factors over others, which leads to factors not contributing equally to the decision-making process.

- **Context acquisition.** Traditionally, context can be obtained *explicitly*, *implicitly* or by *inference* [Adomavicius and Tuzhilin, 2015]. Explicit context, e.g., mood, is provided by the user and implicit context, e.g., time and location, is collected by the system and does not require an action from the user. Context can also be inferred using statistical or data mining methods. In reality, context frequently appears in domains other than the target domain, i.e., the domain where the recommendation is performed. Identifying the action's context is then not trivial. It requires establishing connections between domains and transferring knowledge from one to the other.

- **Context modeling.** The vast majority of CARS considers nominal and atomic values for attributes of contextual factors. However, this is not always feasible as the context may occur under complicated forms involving unstructured data, and may result in the loss of valuable information. Moreover, attribute values may not be known in advance and unexpected new events may be accompanied by new contextual values.

The definition of context under the representational and interactional views does not cover the limitations we mentioned and occurring in real-world applications. Therefore, we rely on the definitions of *partially observable* and *unobservable* contexts, originally proposed in [Adomavicius et al., 2011], that we extend to consider real-world constraints.

Partially observable context. Context is considered partially observable in cases where only some information about contextual factors is explicitly known while other information is missing. This notion of partial observability may concern one or several of the previously defined aspects, as explained in the following:

- **Context accessibility.** Context is considered partially observable when some of the relevant contextual factors are unobservable or inaccessible. This is also the case when, given an observable factor, the availability of context values is delayed: Context is not available at the moment of recommendation but after the interaction is completed.

- **Context relevance.** Context is considered partially observable when the relevance of contextual factors is not clearly defined. This involves cases where contextual factors are not equally pertinent for all users and where this information is missing.

- **Context acquisition.** Context is considered partially observable when the context knowledge is available in a domain other than the target domain and has to be transferred to benefit the target domain where the recommendation is performed. In this scope, the direct relation between the contextual factor and the RS' entities, i.e., users and items, may not be observed and has to be established.

- **Context modeling.** Context is considered partially observable when the structure of some contextual factors is unknown. This also occurs when all possible values for context attributes are not known *a priori*.

Unobservable context. On the other hand, context is considered unobservable when related information is totally unknown or inaccessible, e.g., user's mood or intent. Therefore, it cannot be explicitly exploited for recommendation. Nevertheless, the main concern in these cases is that unobservable context causes the emergence of temporal dynamics in user-generated data and should be taken into account in order to offer accurate recommendations. As an example, and following our definition, time-aware RS [Campos et al., 2014] that model the evolution of users and items over time are considered to handle a sort of unobservable context that is expected to influence users and items and affect their behavior over time. While the RS only has access to users' behaviors, it models the dynamics occurring due to some underlying hidden context.

1.2 Contributions

This thesis addresses the problems of *partially observable* and *unobservable* contexts in two different applications respectively: hotel recommendation and online recommendation. Overall, each problem is thoroughly analyzed, novel context-aware approaches are proposed considering the dynamics existing within users and items, and the impact on the recommendation performance is studied.

Hotel recommendation. Hotel recommendation [Zoeter, 2015] is an interesting and challenging problem that has received little attention. The goal is to recommend hotels that the user may like to visit based on his past behavior. While the development of RS is challenging in general, the development of such systems in the hotel domain, in particular, needs to satisfy specific constraints making the direct application of classical approaches insufficient. There is an inherent complexity to the problem, starting from the decision-making process for selecting accommodations, which is sharply different from the one for acquiring tangible goods, to the multifaceted behavior of travelers who often select accommodations based on contextual factors. In addition, travelers recurrently fall into the *cold-start* status due to the volatility of interests and the change in attitudes depending on the context. While CARS are a promising way to address this problem, the notion of context is complex and not easily integrated into existing CARS.

Partially observable context in hotel recommendation. Using context to boost hotel recommendation implies integrating data from different sources, each of them providing different information about the user's context while introducing new modeling challenges. After identifying the main characteristic of the hotel recommendation problem, we explore the incorporation of several contextual factors, considered as partially observable, into the hotel RS to improve its performance. Our main contributions can be summarized as follows:

- **Leveraging explicit context.** We propose a CARS for hotel recommendation that takes into account the physical, social, and modal contexts of users. We design context-aware models that integrate geographical and temporal dimensions, textual reviews extracted from social media, and the trips' intents, in order to alleviate the shortcomings of only using information related to past user behavior. *Explicit context* is used in reference to contextual information directly related either to users or hotels and provided by the users. We demonstrate the effectiveness of using context to improve

the recommendation quality and we further study the sensibility of users with regards to the different factors.

- **Leveraging implicit context.** Given that planned events constitute a major motive for traveling, we propose a novel framework that leverages the schedule of forthcoming events to perform hotel recommendation. Events are available within a rather short time span and may occur under novel forms. While they are considered to be part of the context influencing users' decisions, hotels and events belong to two different domains and there is no explicit link established between users and hotels on one side, and events on the other side. *Implicit context* is used in reference to the contextual information collected by the system. We propose a solution to this problem and show the advantages of exploiting event data for hotel recommendation.

- **Transferring context knowledge across domains.** We propose a cross-domain RS that leverages check-ins information from Location-Based Social Networks (LBSN) to learn mobility patterns and use them for hotel recommendation, given that the choice of destination is an important factor for hotel selection. Cross-domain recommendation is a way to face the sparsity problem by exploiting knowledge from a related domain where feedback can be easily collected. Context knowledge related to the mobility of users is learned in the LBSN domain and then transferred to the hotel domain. We propose an approach for cross-domain recommendation in this setting and show in which cases such information can be beneficial for hotel recommendation.

Following the definitions previously introduced, contextual factors relevant to hotel recommendation are considered *partially observable* for several reasons that are detailed in relevant parts of the thesis, and thus require the design of adapted approaches. On the other hand, the problem of *unobservable* context is studied in the scope of online recommendation.

Online recommendation. With the explosion of the volume of user-generated data, designing online RS that learn from data streams has become essential [Vinagre et al., 2014b]. Most RS proposed in the literature build first a model from a large static dataset, and then rebuild it periodically as new chunks of data arrive and are added to the original dataset. Training a model on a continuously growing dataset is computationally expensive and the frequency of model updates usually depends on the model's complexity and scalability. Therefore, user feedback that is generated after a model update cannot be taken into account by the RS before the next update, which means that the model cannot adapt to quick changes and hence will come up with lower quality recommendations. One way to address this issue is to approach the recommendation problem as a data stream problem and develop online RS that learn from continuous data streams and adapt to changes in real-time.

Unobservable context in online recommendation. The difficulty with learning from real-world data is that the concept of interest may depend on some hidden context that is not given explicitly [Tsymbal, 2004]. Changes in this hidden context can induce changes in the target concept, which is known as *concept drift* and constitute a challenge in learning from data streams [Gama et al., 2014]. Several efforts have been made to develop drift detection techniques and adapt the models accordingly. The problem is even more complicated in online RS since several concepts, i.e., users and items, are evolving in different ways and at

different rates. We study the problem of online adaptive recommendation, which remains an underexplored problem albeit its high relevance in real-world applications, and show the limitations of the relatively few existing approaches within a framework we introduce. Our contributions can be summarized as follows:

- **Dynamic local models.** We propose dynamic local models that learn from streams of user interactions and adapt to *user drifts* or changes in user interests that may occur due to some hidden context. Local models are known for their ability to capture diverse preferences among user subsets. Our approach automatically detects the drift of preferences that leads a user to adopt a behavior closer to users of another subset and adjusts the models accordingly. We show the interest of relying on local models to account for user drifts.
- **Adaptive incremental matrix factorization.** We propose an adaptive learning rate method for Incremental Matrix Factorization (IMF), accounting for *item drifts* or changes in item perceptions occurring in real-time and independently for each item. The learning rate is dynamically adapted over time based on the performance of each item model and manages to maintain the models up to date. We demonstrate the effectiveness of adaptive learning in non-stationary environments instead of learning at a constant pace.
- **Adaptive collaborative topic modeling.** We design a hybrid online RS that leverages textual content to model new items received in real-time in addition to preferences learned from user interactions. Our approach accounts for *item drifts* or changes occurring in item descriptions, by combining a topic model with a drift detection technique. In addition to addressing the item cold-start problem in real-time, we ensure that models are representing the current states of users and items. We highlight that in the absence of a drift detection component, textual information introduces noise and deteriorates the recommendation quality.

We show, for all of the proposed approaches, how the RS performance evolves over time as more adaptive learning is done and we prove the interest of considering the drifts and dynamics occurring due to some hidden context, for a better recommendation quality.

1.3 Organization of the Thesis

Based on the structure presented in the previous section, the thesis is organized as follows:

Chapter 2. We introduce preliminaries related to RS. We cover the definition of the recommendation problem, its challenges and limitations, methodologies used to evaluate the recommendation quality, and the diverse set of existing recommendation approaches with additional details about those relevant to this thesis.

Chapter 3. We present the hotel recommendation problem as occurring in real-world applications and we discuss the particular challenges it faces and its relation to other recommendation problems. We also give insights about travelers' behaviors and describe the notion of context arising in the domain.

Chapter 4. We propose our CARS for hotel recommendation, integrating the physical, social, and modal contexts of users, including geography, temporality, textual reviews extracted from social media, and the trips' intents. We present the architecture of the system developed in industry and we show the impact of considering contextual factors and user segmentation on improving the quality of recommendation.

Chapter 5. We propose our framework integrating information related to planned events into hotel RS and addressing the *hotel-centric* and *event-centric* problems, which are also introduced in that chapter. We demonstrate the functioning of our framework through a qualitative and a quantitative evaluation, and show the advantages of leveraging event data for hotel recommendation.

Chapter 6. We propose our cross-domain RS leveraging knowledge about user mobility extracted from LBSN to benefit hotel recommendation. We present how we map users and items from both domains and show how knowledge from LBSN contributes in boosting hotel recommendation.

Chapter 7. We introduce the problem of online adaptive recommendation and discuss its relation with existing work in the RS and data stream mining fields. We present a framework for online adaptive recommendation that we use to review previous work and highlight its limitations for the problem considered.

Chapter 8. We present DOLORES, our approach to adapt to user drifts occurring in online RS. DOLORES is based on local models that are able to capture diverse and opposing preferences among user groups. We show the effectiveness of using local models to adapt to changes in user preferences.

Chapter 9. We present AdaIMF, our approach to adapt to drifts in item perceptions occurring in online RS. AdaIMF leverages a novel adaptive learning rate schedule for Incremental Matrix Factorization (IMF). We show how AdaIMF behaves in the presence of item drifts and the importance of accounting for drifts in item perceptions.

Chapter 10. We present CoAWILDA, our approach to adapt to drifts in item descriptions occurring in online RS, and to address the item cold-start problem. CoAWILDA combines techniques from Collaborative Filtering (CF), topic modeling, and drift detection. We show how textual information deteriorates the recommendation quality in the absence of a drift detection component.

Chapter 11. We summarize our contributions and indicate directions for future work.

1.3.1 Publications

Some of the results presented in this thesis are based on the following publications.

Chapter 4 contains results from [Al-Ghossein et al., 2018d]:

[Al-Ghossein et al., 2018d] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Exploiting Contextual and External Data for Hotel Recommendation. In *Adjunct Publication*

of the 26th Conference on User Modeling, Adaptation and Personalization (UMAP), pages 323–328, 2018d

Chapter 5 contains results from [Al-Ghossein et al., 2018a]:

[Al-Ghossein et al., 2018a] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Open data in the hotel industry: leveraging forthcoming events for hotel recommendation. *Journal of Information Technology & Tourism*, pages 1–26, 2018a

Chapter 6 contains results from [Al-Ghossein and Abdessalem, 2016; Al-Ghossein et al., 2018c]:

[Al-Ghossein and Abdessalem, 2016] Marie Al-Ghossein and Talel Abdessalem. SoMap: Dynamic Clustering and Ranking of Geotagged Posts. In *Proc. 25th International Conference Companion on World Wide Web (WWW)*, pages 151–154, 2016

[Al-Ghossein et al., 2018c] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Cross-Domain Recommendation in the Hotel Sector. In *Proc. Workshop on Recommenders in Tourism at the 12th ACM Conference on Recommender Systems (RecTour@RecSys)*, pages 1–6, 2018c

Chapter 8 contains results from [Al-Ghossein et al., 2018b]:

[Al-Ghossein et al., 2018b] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Dynamic Local Models for Online Recommendation. In *Companion Proc. of the The Web Conference (WWW)*, pages 1419–1423, 2018b

Chapter 10 contains results from [Al-Ghossein et al., 2018e,f; Murena et al., 2018]:

[Murena et al., 2018] Pierre-Alexandre Murena, Marie Al-Ghossein, Talel Abdessalem, and Antoine Cornuéjols. Adaptive Window Strategy for Topic Modeling in Document Streams. In *Proc. International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2018

[Al-Ghossein et al., 2018f] Marie Al-Ghossein, Pierre-Alexandre Murena, Antoine Cornuéjols, and Talel Abdessalem. Online Learning with Reoccurring Drifts: The Perspective of Case-Based Reasoning. In *Proc. Workshop on Synergies between CBR and Machine Learning at the 26th International Conference on Case-Based Reasoning (CBRML@ICCBR)*, pages 133–142, 2018f

[Al-Ghossein et al., 2018e] Marie Al-Ghossein, Pierre-Alexandre Murena, Talel Abdessalem, Anthony Barré, and Antoine Cornuéjols. Adaptive Collaborative Topic Modeling for Online Recommendation. In *Proc. 12th ACM Conference on Recommender Systems (RecSys)*, pages 338–346, 2018e

Chapter 2

Recommender Systems

Recommender Systems (RS) [Ricci et al., 2015] are software tools and techniques that provide personalized suggestions of items for users, the items being drawn from a large catalog. RS rely on a set of multidisciplinary theories and techniques from varied fields such as information retrieval, machine learning, decision support systems, human computer interface, marketing, and others. There has been extensive research trying to tackle the different aspects and challenges of RS especially due to their usefulness in scenarios where users are overwhelmed by information overload. They have wide applicability since they can increase core business metrics such as user satisfaction, user loyalty, and revenue. In fact, RS have been used to recommend movies at Netflix [Gomez-Uribe and Hunt, 2016], products at Amazon [Linden et al., 2003], videos at Youtube [Covington et al., 2016], music at Spotify [Jacobson et al., 2016], and friends in social networks like Facebook or Twitter [Hannon et al., 2010], among others.

In all their forms, RS analyze the past behavior of individual users that reveals their preferences towards certain items and that is recorded under various forms of actions like clicks, ratings, and purchases. These actions and the detected patterns are then used to predict items' relevance and compute recommendations matching the user profiles.

This chapter provides a general overview of the area of RS, focusing on the definitions, concepts, and techniques that are relevant to this thesis. We start by defining the recommendation problem and we outline the main challenges and limitations encountered in the field. We then provide the data representation and summarize the main notations used throughout the thesis, for the reader's convenience. After discussing the methodologies used to evaluate RS, we present a broad classification of recommendation techniques and particularly focus on two of them: Collaborative Filtering (CF) and context-aware approaches.

2.1 The Recommendation Problem

2.1.1 Problem Formulation

In October 2006, Netflix announced the Netflix Prize [1] [Bennett and Lanning, 2007], a competition to predict movie ratings on a 5-star scale. The company released a large movie rating dataset and challenged the community to develop a RS that would beat the accuracy of their own for a $1 million prize. This event highlighted the value of generating personalized recommendations, and the academic interest towards the RS field has increased dramatically since then.

Launching the competition required formulating a recommendation problem that could be easily evaluated and quantified. The formulated problem, known as the *rating prediction* problem, became the standard one adopted when developing and evaluating recommendation approaches and has been widely studied for several applications.

Definition 2.1. Rating prediction. The rating prediction problem states that the task of a RS is to estimate a utility function that predicts the rating that a user will give to an item he did not rate, i.e., predicting the preference that a user has for an item.

The objective in rating prediction is to define a utility function that minimizes the error between predicted ratings and actual ratings for all observed ratings. The predicted ratings are then used to order the list of items for each user and perform recommendation. Further advances in the field realized that in real-world RS, users tend to look at the top provided recommendations without being interested in the items situated at the middle or the bottom of the list. It is therefore a lot more valuable to provide relevant top recommendations rather than accurately predicting ratings for all observed items, including those at the bottom of the list that are not reached by the user. The recommendation problem took then a more adapted formulation which is known as the *top-N recommendation* problem [Deshpande and Karypis, 2004].

Definition 2.2. Top-N recommendation. The top-N recommendation problem states that the task of a RS is to estimate a utility function that predicts whether the user will choose an item or not, i.e., predicting the relevance of an item for a user.

In both formulations, the core of the RS that is expected to generate recommendations is considered to follow the same process [Adomavicius and Tuzhilin, 2005]. The recommender algorithm predicts the utility of each item, i.e., the rating in the *rating prediction* problem and a relevance score in the *top-N recommendation* problem, for a target user. This utility measure reveals the usefulness of an item for a user. Items chosen then for recommendation are the ones maximizing the predicted utility. Initially, the utility function is not observed on the whole space but only on a subset of it: Users only interact with a small subset of items. The central problem of RS lies in correctly defining this utility function and extrapolating it on the whole space.

[1] http://www.netflixprize.com

2.1.2 Types of Feedback

The dataset released by Netflix within the context of the Netflix Prize gathered 100 million ratings with their dates from over 480 thousand anonymous users on nearly 18 thousand movies. Feedback was thus recorded in the form of ratings explicitly provided by users and consists what is called *explicit feedback*.

Definition 2.3. Explicit feedback. Explicit feedback is a form of feedback directly reported by the user to the system and is often provided in the form of *ratings* on a numerical scale, e.g., 5-star scale. Other forms of explicit feedback also exist [Schafer et al., 2007] such as binary feedback, e.g., like or dislike, and feedback on ordinal scales, e.g., disagree; neutral; agree. Explicit feedback can also be collected in the form of textual tags or comments.

The emergence of new application domains for RS highlighted the fact that rating data is not always available: Users may be unwilling to provide ratings or the system may be unable to collect this sort of data. RS rely then on *implicit feedback* data to uncover user preferences.

Definition 2.4. Implicit feedback. Implicit feedback [Oard and Kim, 1998] is collected by the system without the intervention of the user. It is inferred from the user behavior and includes user interactions like clicks on items, bookmarks of pages, and item purchases.

Implicit feedback is more abundant than explicit feedback as it is easier to be acquired and requires no user involvement. However, it is inherently noisy due to its nature and has to be treated carefully when inferring user preferences.

2.1.3 Challenges and Limitations

The winning algorithm of the Netflix Prize succeeded in reducing the prediction error by 10% compared to the existing system at Netflix and was based on the combination of hundreds of models [Bell and Koren, 2007a]. However, this algorithm, as originally designed, was never used in industry: The additional increase in performance did not justify the engineering effort required to deploy the solution in a production environment [Amatriain and Basilico, 2012]. Furthermore, the original solution was built to handle 100 million ratings while Netflix had over 5 billion [Amatriain and Basilico, 2016]. It was also designed to work on a static dataset and had no capacity of adapting as new ratings were provided by users.

This interesting fact spotlights the gap between research contributions and industrial applications where RS are actually deployed. Even though the research and development in the field of RS contributed to improve the accuracy of recommendations and enhance user satisfaction, it also highlighted several open challenges and issues that limit the usefulness of recommendations in real-world applications and that should be specifically addressed. We outline the most important ones in the following, knowing that we do not tackle all of them in this thesis.

Rating data sparsity. Users usually only interact with a small number of items selected from a very large catalog of available items. Some recommendation approaches are not able

to infer a proper utility function from insufficient data, resulting in a bad recommendation quality.

Cold-start. In a system that is constantly evolving, new users regularly arrive to use the service for the first time and new items are regularly added to the catalog. Since historical data is missing, it is difficult to generate recommendations for new users and to recommend new items for existing users. This is an important challenge in RS, commonly known as the *cold-start* problem [Kluver and Konstan, 2014]. An efficient RS should be able to produce recommendations under cold-start conditions. When evaluating recommendation approaches, a common practice among researchers is to discard users that have made less than a certain number of interactions which prevents a proper analysis of the recommendation performance in such conditions. Nevertheless, several efforts have been made to develop approaches that specifically cope with cold-start scenarios [Gantner et al., 2010a; Park and Chu, 2009; Schein et al., 2002] by mainly leveraging features and content information describing users and items.

Overspecialization. Recommendation algorithms tend to recommend items that are too similar to what the user already experienced, resulting in *overspecialized* recommendations [Adamopoulos and Tuzhilin, 2014]. Although similarity to previous user selections is a good predictor of user relevance, it does not serve the core purpose of a RS which includes allowing the discovery of new and unexpected content. Providing diverse recommendations is thus essential to address the variety of user interests and avoid what is commonly called the *filter bubble* effect [Nguyen et al., 2014].

Popularity bias. Most item catalogs exhibit the *long tail* effect where a small number of items are popular and concerned by the majority of user interactions, and the largest portion of items are unpopular and have none or few interactions. The long tail effect raises two issues in particular. On one hand, some recommendation approaches, suffering from a popularity bias, run the risk of recommending popular items to everyone [Zhao et al., 2013]. On the other hand, it becomes harder for the RS to promote long tail items with little available feedback.

Implicit feedback. Implicit feedback is much more available than explicit feedback and is more relevant in several applications where common user actions include clicking, buying, watching, listening, or reading. Dealing with this type of feedback introduces important challenges [Hu et al., 2008]. Observations recorded for each user consist of *positive items*, i.e., items the user interacted with, and *negative items*, i.e., items the user did not interact with. In reality, every positive item is not necessarily an item liked by the user. For example, the user may be buying a product for someone else or he may be unsatisfied with it. It is also not possible to determine which positive item is preferred over any other positive one. On the other hand, negative items are a mixture of *unliked items* and *unknown items*. An absence of interaction may indicate that the user does not like the item or that he is not aware of its existence. Handling implicit feedback requires the development of specific recommendation approaches that account for these challenges [Rendle et al., 2009].

New feedback integration. Most recommendation approaches are meant to run on a static dataset and are not equipped to integrate new feedback, new users, and new items, which are constantly observed and generated in real-world scenarios. Recommendations are thus not adapted to the current user preferences and item descriptions, which deteriorates

the RS performance. Online RS are designed to address this problem by continuously integrating new observations [Rendle and Schmidt-Thieme, 2008].

Temporal dynamics. User preferences and item perceptions are changing over time due to the emergence of new items and to seasonality factors, for example. While modeling temporal dynamics in recommendation approaches is essential to offer accurate recommendations, it introduces specific challenges related, in particular, to the fact that users and items are shifting simultaneously in different ways [Koren, 2009].

Scalability. The main focus of researches in the RS field has been on improving the quality of recommendations. From an industry perspective, providing accurate recommendations is not the only thing that matters. In particular, a recommendation algorithm should have the ability to scale with the growing number of users and items, and recommendations should be computed in a reasonable time [Aiolli, 2013].

2.1.4 Data Representation and Notations

General notations. We present in this section the notations used throughout this thesis. All *matrices* are represented by bold upper case letters, e.g., $\mathbf{A}, \mathbf{B}, \mathbf{C}$. All *vectors* are represented by bold lower case letters and are column vectors, e.g., $\mathbf{a}, \mathbf{b}, \mathbf{c}$. The i-th row of a matrix $\mathbf{A}$ is denoted by $\mathbf{a}_i^\top$ and the j-th column by $\mathbf{a}_j$. The element of a matrix $\mathbf{A}$ corresponding to the i-th row and the j-th column is denoted by a_{ij}. All *sets* are represented by calligraphic letters, e.g., $\mathcal{A}, \mathcal{B}, \mathcal{C}$. Aside from the mathematical notations, datasets are represented by small capitals, e.g., DATASET, and evaluated recommendation methods by a typewriter font, e.g., `Method`.

RS notations. Without loss of generality, the recommendation problem can be formulated as suggesting a limited number of elements selected from a catalog of items to a target user. We denote by $\mathcal{U}$ the set of users handled by the system and by n the number of users $|\mathcal{U}|$. We denote by $\mathcal{I}$ the set of items contained in the catalog and by m the number of items $|\mathcal{I}|$.

Making personalized recommendations requires some knowledge about the feedback given by users to items through interactions of various types, e.g., ratings, clicks, and purchases. We denote by $\mathcal{D}$ the set of observed interactions, having that one interaction is represented by the triple (u, i, t) where u denotes the individual user, i the item, and t the timestamp at which the interaction happened. The feedback data is encoded in a matrix $\mathbf{R}$ of size $n \times m$, called the *feedback matrix* or *rating matrix* and illustrated in Figure 2.1. The entry r_{ui} represents the feedback given by user u to item i, if any feedback is given. The vector $\mathbf{r}_u^\top$, i.e., the row u of the matrix $\mathbf{R}$, represents the feedback provided by user u to all items, and the vector $\mathbf{r}_i$, i.e., the column i of the matrix $\mathbf{R}$, represents the feedback provided by all users to item i.

In settings where users provide explicit feedback in the form of numerical ratings, r_{ui} takes the value of the rating given by user u to item i. When considering implicit feedback, r_{ui} takes the value 1 if user u interacted with item i. Computing recommendations requires predicting the values of the missing elements from $\mathbf{R}$ and a predicted value is denoted by $\hat{r}_{ui}$.

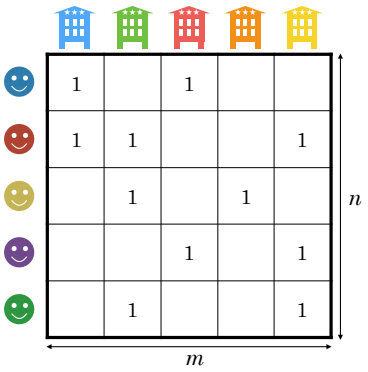


FIGURE 2.1: Representation of the feedback matrix

TABLE 2.1: Common notations used in this thesis

Symbol	Definition
$\mathcal{U}$	set of users
n	number of users
$\mathcal{I}$	set of items
m	number of items
u	individual user
i	individual item
(u, i, t)	interaction at time t
$\mathcal{D}$	set of interactions
$\mathbf{R}$	feedback matrix
$\mathbf{r}_u^\top$	feedback vector of user u
$\mathbf{r}_i$	feedback vector for item i
r_{ui}	feedback of user u for item i
$\hat{r}_{ui}$	predicted feedback of user u for item i
$\mathcal{U}_i$	set of users that have rated item i
$\mathcal{I}_u$	set of items rated by user u
N	number of recommended items

We refer to the items the user has interacted with as *observed* or *rated* items, even if the interaction is under a form different than a numerical rating. The set of users that have rated item i is denoted by $\mathcal{U}_i$ and the set of items rated by user u is denoted by $\mathcal{I}_u$. Table 2.1 is a reference table containing all common notations used in this thesis and is provided for the reader's convenience.

Algorithm 1 presents a standard recommendation algorithm as described in Section 2.1.1 and following the notations introduced in this section. We note that this version of the recommendation algorithm does not consider already rated items as candidates for recommendation. Domains where repeated interactions are expected consider the whole set of items $\mathcal{I}$ for recommendation.

Algorithm 1 A recommendation algorithm

Input: user u, number of items to recommend N

for i in $\mathcal{I} \setminus \mathcal{I}_u$ **do**
    Predict $\hat{r}_{ui}$
end for
Create list of $\mathcal{I} \setminus \mathcal{I}_u$ items ordered by decreasing order of $\hat{r}_{ui}$
Return N first items of the list

2.2 Evaluation of Recommender Systems

In order to evaluate the performance of RS and compare different approaches, it is necessary to define proper evaluation methodologies and metrics measuring the quality of recommendation. The goal of the evaluation is to assess the ability of the RS to meet its core objectives including, but not limited to, suggesting relevant items to users. While some effects can be easily measured, others result in the need to define plausible proxies. Nevertheless, evaluating the RS performance requires choosing a proper methodology, the criteria to evaluate, and the metrics to measure.

2.2.1 Evaluation Methods

The evaluation methodology defines the experimental protocol followed to evaluate the RS and falls into one of the three following categories: the offline setting, the user studies, and the online setting. While this section provides the corresponding main guidelines, further detailed information can be found in [Gunawardana and Shani, 2015].

2.2.1.1 Offline Evaluation

Offline experiments are performed using previously collected datasets gathering user interactions and try to simulate the user behavior when interacting with the RS. Offline evaluation is among the most popular settings for evaluating RS since it does not require any interaction with real users and allows replicability and comparison of approaches at low cost. The main assumption made is that the user behavior at the time the data was collected is similar to the one that will be adopted when the RS will be deployed so that consistent conclusions can be drawn from the simulation. However, offline experiments cannot measure the influence of the RS on the user behavior. The experimental setting described next corresponds to the *batch protocol* which is by far the most commonly used method for RS evaluation.

The basic structure for offline evaluation is based on the train-test and cross-validation techniques, widely used in machine learning. The data is usually divided into two distinct parts: the *training set*, used to estimate the utility function, and the *test set*, used to measure the RS performance. There are several ways to split the data which are mentioned in the following, knowing that the selected option depends on the application domain and its constraints.

Random split. The separation of sets is done by shuffling user interactions and randomly selecting a certain percentage for each set, e.g., 80% of the interactions in the training set and 20% in the test set. The random selection is done without replacement, i.e., using a hold-out method, to prevent one interaction from being used both for training and for testing. The *k-fold cross-validation* method is carried out by repeating this procedure k times and measuring the performance each time.

***Given-n* split.** The training set is constituted by randomly selecting a fixed number n of interactions for each user. The rest of the interactions form the test set. By setting a small value for n, experiments allow to test the RS robustness to user cold-start scenarios where little information about users is available.

Chronological split. When the temporal information of interactions is available, it is possible to divide the dataset by considering interactions that have occurred before a certain time threshold in the training set and those after the threshold in the test set. This setting simulates the case where the utility function is estimated at the time threshold and then used to perform recommendations without accounting for interactions received afterward.

Further realism can be obtained by simulating what the RS predictions could have been if it had been running at the time the dataset was collected, which is the idea adopted in the *sequential protocol*. In this setting, interactions are considered one by one in temporal order, provided that temporal information is available [Burke, 2010; Lathia et al., 2009]. For each considered interaction, we attempt to perform recommendation and then add the interaction to the set of interactions used to estimate the utility function.

As mentioned before, offline evaluation faces some limitations. In particular, there is no certainty that the results obtained from this evaluation will align with the actual results obtained in a real-world RS. It is also not possible to measure the real impact of recommendations on the user behavior. Therefore, more costly evaluation protocols requiring user involvement should be considered.

2.2.1.2 User Studies

In order to properly evaluate RS, real user responses to suggestions should be collected. One way to achieve this is by conducting user studies involving a small number of users who are recruited and asked to interact with the RS. The user behavior is recorded while running controlled experiments and each user may also provide explicit feedback about the experience. The user studies methodology delivers more insights than offline evaluation but is also subject to limitations. The small number of users involved in addition to their awareness of participating in a study where their behavior is being recorded may introduce biases in the collected data.

2.2.1.3 Online Evaluation

Online evaluation is performed when evaluating real-world RS and is only possible when the RS is deployed online. This type of evaluation is the one that provides the strongest

evidence as the RS is being used by real users in a real setting. The idea is to measure the change in user behavior when interacting with different RS and draw conclusions related to the performance.

A/B testing is an online evaluation method widely used to evaluate new algorithms or design features [Amatriain, 2013]. It can be seen as a two-sample hypothesis testing methodology where two versions of the RS are compared, i.e., the current one and the new one. The user traffic is randomly divided between both of them, and some metrics are measured for a certain period of time. Comparing the results drives the conclusion of whether or not the new alternative should be adopted.

The risk taken when performing online evaluation is to negatively affect the experience of real users when testing under-performing approaches. Some companies follow an offline-online testing process to get the best of the offline and online evaluations [Amatriain, 2013]. Offline evaluation is first used as an indicator to make informed decisions and A/B tests are then performed to prove hypothesis previously validated by offline evaluation.

2.2.2 Evaluation Criteria

RS are expected to meet a variety of criteria, each of them answering a different requirement which importance is determined by the application domain where the RS is deployed. The overall performance is obtained by trading off the multiple criteria. Some of them are cited in the following while a complete list and further information can be found in [Gunawardana and Shani, 2015].

Accuracy. At the core of the RS lies a prediction engine that predicts the utility of an item for a user. A RS providing more accurate predictions is expected to be preferred by users, and several metrics have been introduced to measure the accuracy of predictions [Gunawardana and Shani, 2015].

Coverage. A typical item catalog suffers from the long tail effect where the largest part of items is rarely selected for consumption. It is often desirable to optimize a RS for a wider catalog coverage by including and promoting long-tail items [Ge et al., 2010]. User coverage is also relevant: The RS is evaluated with regards to the proportion of users for which it can deliver recommendations.

Novelty. RS are expected to recommend items which the user is not aware of. In particular, recommending items that have been previously rated or seen by the user is not valuable in applications that require novel recommendations [Celma and Herrera, 2008; Vargas and Castells, 2011].

Diversity. RS should be capable of recommending items that span the whole set of user preferences and not only very similar items related to a single user interest. Recommendations should thus be diverse and include items that are different from each others [Zhang and Hurley, 2008; Ziegler et al., 2005].

Serendipity. Serendipity measures how surprising the relevant recommendations are [Ge et al., 2010; Murakami et al., 2007]. RS are not expected to recommend obvious options

but rather unexpected ones that users would not have been able to find while searching on their own.

Explainability. Recommendations may not be considered reliable from a user point of view unless they can be reasonably explained and justified [Tintarev and Masthoff, 2015]. This criteria affects the user experience and raises ethical concerns around the opacity of algorithms and the manipulation of users.

2.2.3 Evaluation Metrics

Accuracy is by far the most used property to evaluate RS in the literature [Gunawardana and Shani, 2015]. We focus in this part on the different metrics used to evaluate accuracy. These metrics can be divided into two categories: *prediction accuracy metrics* and *top-N metrics*.

Prediction accuracy metrics. Prediction accuracy metrics are used to evaluate rating prediction approaches for recommendation by measuring the accuracy with which the ratings are predicted for each rating included in the test set $\mathcal{T}$. This category of metrics includes the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) which was adopted as the standard metric for the Netflix Prize [Bennett and Lanning, 2007]. The lower the error value, the better the predictive power of the system is.

- The *Mean Absolute Error (MAE)* measures the average absolute deviation between each rating r_{ui} contained in the test set $\mathcal{T}$ and its prediction $\hat{r}_{ui}$. It is defined as follows:

$$MAE = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |r_{ui} - \hat{r}_{ui}| \tag{2.1}$$

- The *Root Mean Squared Error (RMSE)* relies on the *Mean Squared Error (MSE)* which penalizes large errors compared to the MAE, and is defined as follows:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (r_{ui} - \hat{r}_{ui})^2} \tag{2.2}$$

These metrics are applicable in domains where explicit ratings are available and where the RS is optimized for rating prediction.

Top-N metrics. Top-N metrics evaluate the quality of top-N recommendation lists generated by RS. Often borrowed from information retrieval, these metrics measure the relevance of recommendations, knowing that the item relevancy can be determined in different ways, e.g., the user bought the item, rated it, or clicked on it.

In particular, precision and recall [Baeza-Yates and Ribeiro-Neto, 2011] are two metrics measuring the *usage prediction* of the RS. Given that the algorithm recommends N items, we denote by $\mathcal{L}_u(N)$ the set of items recommended to user u, by $\mathcal{S}_u$ the set of relevant items among all items for user u, and by $\mathcal{T}_u$ the set of users having interactions in the test set $\mathcal{T}$.

- The *precision* measures the fraction of relevant recommended items and is defined as follows:

$$Precision@N = \frac{1}{|\mathcal{T}_u|} \sum_{u \in \mathcal{T}_u} Precision_u@N = \frac{1}{|\mathcal{T}_u|} \sum_{u \in \mathcal{T}_u} \frac{|\mathcal{L}_u(N) \cap \mathcal{S}_u|}{|\mathcal{L}_u(N)|} \tag{2.3}$$

- The *recall* measures which fraction of the relevant items are present in the recommendation list and is defined as follows:

$$Recall@N = \frac{1}{|\mathcal{T}_u|} \sum_{u \in \mathcal{T}_u} Recall_u@N = \frac{1}{|\mathcal{T}_u|} \sum_{u \in \mathcal{T}_u} \frac{|\mathcal{L}_u(N) \cap \mathcal{S}_u|}{|\mathcal{S}_u|} \tag{2.4}$$

Increasing the size of the recommendation list, N, may increase the recall since a longer recommendation list has more chances of including relevant items, but it also decreases the precision. The *F_1 measure* evaluates the balance between these two measures and is defined as follows:

$$F_1@N = \frac{1}{|\mathcal{T}_u|} \sum_{u \in \mathcal{T}_u} 2 \cdot \frac{Precision_u@N \ . \ Recall_u@N}{Precision_u@N \ + \ Recall_u@N} \tag{2.5}$$

Ranked lists introduce a position bias according to which the probability of selection of an item decays when its rank increases in the list [Craswell et al., 2008]. It is thus important to evaluate not only the relevance of the items in the recommendation list but also the *ranking quality*. Two of the most popular ranking measures are the Normalized Discounted Cumulative Gain (NDCG) and the Mean Reciprocal Rank (MRR) introduced in the following.

- The *Discounted Cumulative Gain (DCG)* is first defined for user u as follows:

$$DCG_u@N = \sum_{i=1}^{N} \frac{rel_{ui}}{log_2(i+1)} \tag{2.6}$$

  where $rel_{ui} = 1$ if the item displayed at rank i is relevant for user u and $rel_{ui} = 0$ otherwise.

  The *Normalized Discounted Cumulative Gain (NDCG)* [Järvelin and Kekäläinen, 2002] is then defined as follows:

$$NDCG@N = \frac{1}{|\mathcal{T}_u|} \sum_{u \in \mathcal{T}_u} \frac{DCG_u@N}{DCG^*_u@N}, \tag{2.7}$$

  where $DCG^*_u@N$ is the best possible DCG_u obtained if all recommended items were relevant.

- The *Mean Reciprocal Rank (MRR)* computes the reciprocal of the rank of the first relevant item in the full ordered list of items, denoted by $rank_u$ for user u, and is defined as follows:

$$MRR = \frac{1}{|\mathcal{T}_u|} \sum_{u \in \mathcal{T}_u} MRR_u = \frac{1}{|\mathcal{T}_u|} \sum_{u \in \mathcal{T}_u} \frac{1}{rank_u} \tag{2.8}$$

Other measures used for evaluating the accuracy of RS include the mean average precision [Baeza-Yates and Ribeiro-Neto, 2011], the area under the ROC curve [Bradley, 1997], the expected reciprocal rank [Chapelle et al., 2009], the hit-rate [Deshpande and Karypis, 2004], and the average-reciprocal hit rank [Deshpande and Karypis, 2004].

2.3 Recommendation Approaches

A recommendation approach, also referred to as recommendation method or recommendation algorithm, is expected to predict the utilities of items for a set of target users and generate appropriate recommendations. A large range of approaches have been proposed to offer accurate recommendations and can be classified according to multiple criteria including the recommendation problem they address, i.e., rating prediction or top-N recommendation, or the type of feedback they employ, i.e., explicit or implicit feedback (Section 2.1.1).

The most common classification found in the literature refers to how the information related to users and items is exploited for the recommendation task and establishes the following categories:

- *Content-Based Filtering (CBF) approaches.* These approaches make explicit use of domain knowledge related to users or items.
- *Collaborative Filtering (CF) approaches.* Recommendations are only based on the user behavior and on previous interactions, without further explicit information about users or items.
- *Hybrid approaches.* These approaches combine the two previous strategies using both user and item information, and user interactions.
- *Context-aware approaches.* These approaches leverage contextual information about users and items in order to propose appropriate recommendations.

2.4 Content-Based Filtering Approaches

Content-Based Filtering (CBF) approaches [Pazzani and Billsus, 2007] use content information about users and items in order to generate recommendations. This information can take various forms such as features, textual descriptions, and tags. In order to recommend relevant items, the main idea is to match user profiles and item profiles based on user preferences for item attributes. Therefore, users receive suggestions about items that are similar to the ones they previously interacted with. Deploying a CBF approach requires extracting relevant information about the content of items, building item profiles and user profiles, and filtering items according to the similarity between profiles.

2.4.1 Example of a Content-Based Filtering Approach

CBF techniques build models using either machine learning techniques, e.g., naïve Bayes or decision trees, based on underlying data [De Gemmis et al., 2008; Lops et al., 2011; Pazzani and Billsus, 1997] or heuristic functions inspired from information retrieval methods [Balabanović and Shoham, 1997; Diederich and Iofciu, 2006]. In order to give an example of the process followed by CBF approaches, we illustrate in the following the functioning of a CBF technique relying on a function from the information retrieval field to represent items [Cantador et al., 2010].

Building item profiles. Most CBF systems exploit textual features describing items and extracted from various sources like Web pages and product descriptions. The representation is traditionally based on the vector space model [Lops et al., 2011] which is a spatial representation of text documents. Each document is represented by a vector in a n-dimensional space, denoted by $\mathbf{v}_i$ for the document related to item i, where each dimension corresponds to a term from the vocabulary $\mathcal{V}$. Weighting terms is done with *TF-IDF* which is the product of *Term Frequency (TF)* and *Inverse Document Frequency (IDF)* defined as follows, for each word w in document d_i:

$$TF_{w,d_i} = \frac{f_{w,d_i}}{\max\limits_{w' \in d_i} f_{w',d_i}}, \quad IDF_{w,d_i} = \log \frac{|\mathcal{I}|}{1 + |\{i \in \mathcal{I} : w \in d_i\}|} \tag{2.9}$$

where f_{w,d_i} is the number of occurrences of the word w in the document d_i.

The normalized TF-IDF is used for representation and is given by:

$$\mathbf{v}_{i,w} = \frac{TF_{w,d_i} \,.\, IDF_{w,d_i}}{\sqrt{\sum\limits_{w' \in d_i} TF^2_{w',d_i}} \cdot \sqrt{\sum\limits_{w' \in d_i} IDF^2_{w',d_i}}} \tag{2.10}$$

In cases where items are described by a set of attributes having specific values, the item profile is built naturally by allocating one dimension to each feature.

Building user profiles. In order to match users and items, CBF approaches require the creation of vectors describing user preferences in the same space where items are represented. User profiles indicate the user interest in the various item dimensions. An estimate of a user profile, which is denoted by $\mathbf{v}_u$ for user u, can be obtained by aggregating the profiles of the items he rated.

Measuring similarities. Predicting the user's interest in a particular item can be done by computing the similarity between user and item profiles. There are several existing measures computing the proximity of two vectors. Cosine similarity is one of the most widely used and is defined as follows:

$$sim_{CS}(u,i) = \frac{\sum\limits_{w \in \mathcal{V}} \mathbf{v}_{u,w} \cdot \mathbf{v}_{i,w}}{\sqrt{\sum\limits_{w \in \mathcal{V}} \mathbf{v}_{u,w}^2} \cdot \sqrt{\sum\limits_{w \in \mathcal{V}} \mathbf{v}_{i,w}^2}} \tag{2.11}$$

2.4.2 Advantages and Disadvantages

CBF approaches offer a number of advantages, some of which are mentioned in the following:

- *Item cold-start.* CBF approaches exploit user interactions made by the target user and content information about items in order to generate recommendations. Interactions made by other users are not needed and items are recommended based on their descriptions even if they have not been rated by any user. CBF approaches are therefore able to deliver recommendations in *item cold-start* scenarios (Section 2.1.3).
- *Explainability.* It is possible to explain the results of recommendations by providing the set of content features that caused an item to appear in the recommendation list. Those features indicate whether or not recommendations can be trusted by the user and increase the system's transparency (Section 2.2.2).

However, CBF approaches suffer from several shortcomings:

- *Content availability.* One obvious limitation is due to the fact that the performance of these techniques is tied to the number and type of features associated with items. Domain knowledge is often required and enough information about items, which is not always available, should be gathered in order to appropriately discriminate items.
- *Overspecialization.* CBF approaches propose items that are similar to the ones the user has previously interacted with and cannot provide unexpected suggestions. They suffer from overspecialization and are not able to offer novel or serendipitous recommendations (Section 2.1.3).
- *User cold-start.* Building a consistent user profile requires gathering enough user interactions, making CBF approaches unsuitable for cases where only a few interactions are available or, more specifically, in *user cold-start* scenarios (Section 2.1.3).

2.4.3 Related Recommendation Approaches

Classifications of recommendation approaches identify additional categories than the one mentioned in Section 2.3. Some of them are related to the CBF category and are mentioned in the following, for the sake of completeness:

- *Demographic filtering approaches.* These techniques are based on the user demographic information [Krulwich, 1997; Pazzani, 1999] and are related to CBF approaches since

they leverage *content* about users. Different recommendations are generated for different demographic profiles that include for example the user's country, age, and gender. Given that user preferences cannot be inferred based on their demographic features, these approaches do not perform very well. However, demographic information can be exploited to boost the performance of other well-performing approaches.

- *Knowledge-based approaches.* Given the specific needs and preferences of users, these approaches use domain knowledge around item features to determine if an item is adapted or not. Compared to CBF approaches, knowledge-based approaches do not rely on previous user interactions but rather analyze the user's current query or need in addition to item descriptions. Knowledge-based RS are essentially case-based [Bridge et al., 2005; Chen and Pu, 2012]. When enough information about user interactions is available, these RS are surpassed by approaches leveraging user interactions.

2.5 Collaborative Filtering Approaches

The term *Collaborative Filtering (CF)* first appeared in Tapestry [Goldberg et al., 1992], an experimental mail system which was developed to control the flood of information received by users by filtering out spam emails. It designated the fact that users *collaborate* to help each others *filter* the emails by annotating the usefulness of a particular email and propagating the information to the others. This was the first idea of filtering information based on the feedback of other users, knowing that details about how similar users were found or how personalization was performed were not provided.

Several systems in other domain applications were then developed based on CF [Hill et al., 1995; Resnick et al., 1994; Shardanand and Maes, 1995], and a large number of CF approaches were proposed for performing personalized recommendation. These approaches are based on the following principle: Users who are similar with regards to the history of interactions are considered to be like-minded and are susceptible to share similar preferences in the future. Recommendations are thus based on the user history and on the past behavior of similar users. In contrast to CBF approaches, no additional information about items is required to perform recommendation.

Relation to classification. CF can be formulated as the problem of jointly solving many classification problems, having one classification problem for each available item [Verstrepen et al., 2017]. In every single problem, the corresponding item i serves as the class, the other items as the features, and the users that have interacted with i as the labeled examples. Since all the problems share several features, jointly solving them allows the exchange of information and the collaboration between them, which is more efficient than independently solving many classification problems.

CF approaches can be divided into the two general categories of *memory-based* and *model-based* methods [Koren and Bell, 2015; Ning et al., 2015]. While memory-based approaches directly use the recorded user interactions to compute recommendations, model-based approaches exploit these interactions to learn a predictive model which is then used for recommendation. We provide in the following an overview of these two categories of approaches.

2.5.1 Memory-Based Approaches

Memory-based approaches for CF, i.e., neighborhood-based or heuristic-based approaches, are based on the idea that similar users are interested in the same items or that similar items interest the same users, assuming that user preferences remain stable over time. These approaches can be user-based or item-based [Desrosiers and Karypis, 2011]. In a user-based approach, and to select items to suggest to user u, we refer to users who are the most similar to u. In an item-based approach [Deshpande and Karypis, 2004], we rely on items that are similar to the items that u has rated in the past. In both cases, a neighborhood method first determines the neighbors of an entity using a similarity measure and then relies on the neighbors and on previous user interactions to generate recommendations. Details about the functioning of each approach are provided in the following.

2.5.1.1 User-Based Collaborative Filtering

In a user-based approach, the first step is to find the neighbors of the target user u. Given a similarity measure able to assess the resemblance between two users, we compute the similarities between u and every other user in $\mathcal{U}$. The k most similar users to u constitute the set of neighbors, denoted by $\mathcal{B}(u)$. To predict the rating $\hat{r}_{ui}$, we rely on the ratings given by the k neighbors to item i. We may compute the simple average of the available ratings given by the neighbors or the weighted average taking into account the degree of similarity between users and defined as follows:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in \mathcal{B}(u)} sim(u,v) \ . \ r_{vi}}{\sum\limits_{v \in \mathcal{B}(u)} sim(u,v)} \tag{2.12}$$

where $sim(u,v)$ is the similarity measure applied to users u and v.

Similarity measures play an essential role as they intervene in the neighborhood selection and in the rating prediction. Several measures have been proposed for this task, and we mention in the following some of the most widely used.

Pearson Correlation.

$$sim_{PC}(u,v) = \frac{\sum\limits_{x \in \mathcal{I}_{uv}} (r_{ux} - \overline{r}_u)(r_{vx} - \overline{r}_v)}{\sqrt{\sum\limits_{x \in \mathcal{I}_{uv}} (r_{ux} - \overline{r}_u)^2} \sqrt{\sum\limits_{x \in \mathcal{I}_{uv}} (r_{vx} - \overline{r}_v)^2}} \tag{2.13}$$

where $\overline{r}_u$ and $\overline{r}_v$ are the average ratings given by user u and v respectively, and $\mathcal{I}_{uv}$ is equal to $\mathcal{I}_u \cap \mathcal{I}_v$.

Cosine similarity.

$$sim_{CS}(u,v) = \frac{\mathbf{r}_u^\top \ . \ \mathbf{r}_v^\top}{\|\mathbf{r}_u^\top\| \, \|\mathbf{r}_v^\top\|} \tag{2.14}$$

where $\mathbf{x}$. $\mathbf{y}$ is the scalar product between vectors $\mathbf{x}$ and $\mathbf{y}$, and $\|\mathbf{x}\|$ is the norm of $\mathbf{x}$.

Jaccard similarity.

$$sim_{JS}(u,v) = \frac{|\mathcal{I}_u \cap \mathcal{I}_v|}{|\mathcal{I}_u \cup \mathcal{I}_v|} \tag{2.15}$$

2.5.1.2 Item-Based Collaborative Filtering

Item-based CF [Deshpande and Karypis, 2004] focuses on finding and exploiting the neighbors of items. To predict the rating $\hat{r}_{ui}$, an item-based approach examines the neighbors of item i and the ratings given by user u to these neighbors. We first compute the similarities between item i and every other item in $\mathcal{I}$, and select the k most similar items as neighbors of i forming the set denoted by $\mathcal{B}(i)$. Similarly to user-based CF (Section 2.5.1.1), we compute the predicted rating $\hat{r}_{ui}$ as follows:

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in \mathcal{B}(i)} sim(i,j) \ . \ r_{uj}}{\sum\limits_{j \in \mathcal{B}(i)} sim(i,j)} \tag{2.16}$$

where $sim(i,j)$ is the similarity measure applied to items i and j. The similarity measures defined in Equations (2.13)–(2.15) can be applied, using the vectors $\mathbf{r}_i$ and $\mathbf{r}_j$ instead of the vectors $\mathbf{r}_u^\top$ and $\mathbf{r}_v^\top$, and the sets $\mathcal{U}_i$ and $\mathcal{U}_j$ instead of the sets $\mathcal{I}_u$ and $\mathcal{I}_v$.

2.5.1.3 Extensions, Complexity, Advantages and Disadvantages

Extensions. Important extensions applicable to memory-based CF have been proposed [Breese et al., 1998]. *Default voting* was first introduced and consists in assuming a default value for unseen ratings. These values are then used to measure similarities instead of only using observed ratings, i.e., ratings in $\mathcal{I}_u \cap \mathcal{I}_v$. This is in particular effective in sparse settings where $|\mathcal{I}_u \cap \mathcal{I}_v|$ is relatively small. The default value is defined with regards to the application domain where the absence of observations may sometimes indicate a negative feedback. *Inverse user frequency* was then proposed and consists in giving more weight to less popular items in the process of computing similarities. It is based on the idea that popular items, i.e., items liked by a majority, are not as relevant as less popular items when it comes to capturing similarity. Another extension, *case amplification*, transforms similarities in the prediction phase by boosting high weights and punishing low ones. This is in particular effective to reduce noise.

Complexity. Memory-based approaches first compute nearest neighbors, i.e., the k most similar users or items, and then use these neighbors for prediction. The time complexity for computing neighbors is $O(n^2p)$ in user-based CF where $p = \max\limits_{u \in \mathcal{U}} |\mathcal{I}_u|$, and $O(m^2q)$ in item-based CF where $q = \max\limits_{i \in \mathcal{I}} |\mathcal{U}_i|$. The time complexity for retrieving recommendations is $O(mk)$ for both approaches.

Advantages and disadvantages. Memory-based CF is a simple and intuitive approach for recommendation that does not require more than one parameter, i.e., the number of neighbors k, as input. Recommendations are easy to explain since they are the result of exploiting similar users or items which are clearly identified. They supply statements like "Customers who bought this item also bought..." [Linden et al., 2003].

Item-based approaches perform more accurately than user-based approaches in most reported cases [Sarwar et al., 2001] mainly since item relationships tend to be more stable than user relationships. Item-based approaches are also more computationally efficient in terms of complexity in settings where the number of users exceeds the number of available items. On the other hand, user-based approaches provide more original recommendations and lead to a more satisfying experience [Desrosiers and Karypis, 2011].

Memory-based methods suffer from scalability issues and a lack of sensitivity to sparse data [Lemire and Maclachlan, 2005]. Since CF methods rely on user behavior, a good accuracy is only obtained when enough interactions are collected and made available. While this is valid for both memory-based and model-based approaches, model-based approaches have been proven to outperform memory-based approaches in terms of accuracy. They are therefore preferred in RS applications.

2.5.2 Matrix Factorization Approaches

Model-based approaches [Koren and Bell, 2015] exploit user interactions to learn a predictive model representing user preferences and item descriptions. Characteristics of users and items are captured by a set of model parameters. These parameters are learned from the data and used then for prediction. Several model-based approaches have been explored including clustering [Ungar and Foster, 1998], Bayesian methods [Miyahara and Pazzani, 2000], and neural networks [Salakhutdinov et al., 2007]. We provide in this section a presentation of Matrix Factorization (MF) methods [Koren et al., 2009] which are the most popular techniques among model-based approaches.

2.5.2.1 Matrix Factorization Framework

Matrix Factorization (MF) methods [Koren et al., 2009] model users and items by representing them in a common latent space of dimensionality K where the affinity between a user and an item is determined by the inner product of their embedded vectors. While many variations and extensions of MF methods have been proposed, almost all of them rely on the same framework, represented in Figure 2.2, but differ in the objective function they try to optimize and the method they use to learn parameters.

In MF methods, each user u is associated with a vector $\mathbf{p}_u \in \mathbb{R}^K$ and each item i with a vector $\mathbf{q}_i \in \mathbb{R}^K$. Latent factors represent characteristics inferred from the rating patterns and each value in the vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ expresses to which extent the user is attracted by or the item has a certain feature, respectively. The hypothesis of MF methods is that the value of a rating r_{ui} can be captured by the inner product between $\mathbf{p}_u$ and $\mathbf{q}_i^\top$:

$$\hat{r}_{ui} = \mathbf{p}_u \mathbf{q}_i^\top \tag{2.17}$$

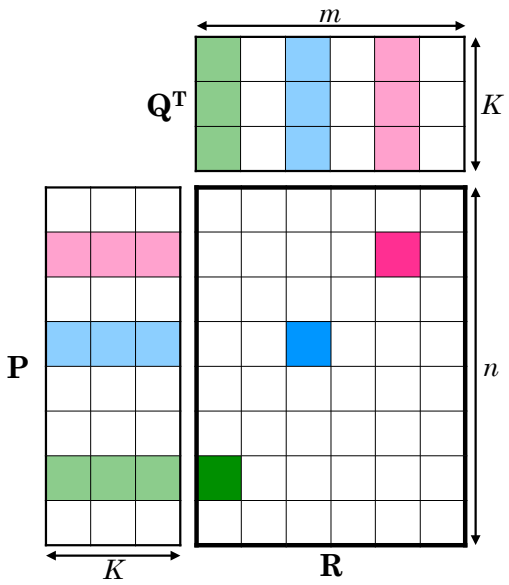


FIGURE 2.2: Representation of Matrix Factorization (MF)

The goal is to determine the vectors $\mathbf{p}_u$ for each user u and $\mathbf{q}_i$ for each item i. MF methods approximate then the rating matrix $\mathbf{R}$ with the product of two low-rank dense matrices $\mathbf{P} \in \mathbb{R}^{n \times K}$ for user latent factors and $\mathbf{Q} \in \mathbb{R}^{m \times K}$ for item latent factors:

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}^\top \tag{2.18}$$

where $K \ll \min(n, m)$.

The learning task consists in determining the values of $\mathbf{P}$ and $\mathbf{Q}$ based on the observed elements of $\mathbf{R}$. The learned latent factors are then used to predict the unobserved elements of $\mathbf{R}$. The exact way the learning of parameters is performed differs across several proposed approaches for MF. Some of the most representative methods are introduced in the following.

2.5.2.2 Singular Value Decomposition

Singular Value Decomposition (SVD) is a well-known technique used for matrix decomposition. SVD decomposes the matrix $\mathbf{R}$ into:

$$\mathbf{R} = \mathbf{P}\mathbf{\Sigma}\mathbf{Q}^\top = \sum_{s=1}^{d} \sigma_s \mathbf{p}_s \mathbf{q}_s^\top \tag{2.19}$$

where $\mathbf{P} \in \mathbb{R}^{n \times d}$ is the set of left singular vectors, $\mathbf{Q} \in \mathbb{R}^{m \times d}$ is the set of right singular vectors, and $\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$ is a diagonal matrix containing the d singular values on its diagonal. Given a SVD decomposition, the matrix $\mathbf{R}$ can be approximated by sorting the singular vectors in decreasing order of σ_s and selecting those corresponding to the K largest singular

values. The best rank-K approximation of $\mathbf{R}$ is then given by:

$$\hat{\mathbf{R}} = \sum_{s=1}^{K} \sigma_s \mathbf{p}_s \mathbf{q}_s^\top \tag{2.20}$$

SVD is closely related to the MF problem introduced in Section 2.5.2.1 and can be applied to uncover the user and item vectors used to approximate the rating matrix [Sarwar et al., 2000]. However, applying SVD for CF raises the issue of sparsity: While SVD requires a full matrix to be decomposed, the rating matrix is typically sparse and contains many unknown elements, i.e., user-item couples for which the preference is not specified. Some solutions have been proposed including filling the rating matrix with default values, e.g., zeros or an average rating per user or per item [Kurucz et al., 2007]. This solution is inaccurate since filling the matrix might falsify the represented concepts. It is also very expensive as it increases the amount of data. Further advances showed that it is more efficient to factorize the matrix $\mathbf{R}$ using only observed values [Paterek, 2007; Takács et al., 2007] and are discussed in the following.

2.5.2.3 Minimizing Squared Loss and Other Loss Functions

The goal in MF is to learn the user and item feature matrices $\mathbf{P}$ and $\mathbf{Q}$ that minimize the objective function, denoted by $\mathcal{L}(\mathbf{P}, \mathbf{Q})$, which is the squared error between observed and predicted ratings over known ratings that are included in the set $\mathcal{D}$ and defined as follows:

$$\mathcal{L}(\mathbf{P}, \mathbf{Q}) = \sum_{(u,i)\in\mathcal{D}} (r_{ui} - \mathbf{p}_u \mathbf{q}_i^\top)^2 \tag{2.21}$$

The minimization problem is solved over the observed entries of $\mathbf{R}$, but since only a few of them are available, the model can easily overfit or be affected by extreme values. It is therefore important to consider regularization for a better generalization capacity [Srebro et al., 2005] by adding the regularization terms $\Omega(\mathbf{P})$ and $\Omega(\mathbf{Q})$ which prevent the values of $\mathbf{P}$ and $\mathbf{Q}$ from being too large. While several forms of regularization can be applied, we use L_2-regularization, i.e., ridge regression, to make $\mathcal{L}(\mathbf{P}, \mathbf{Q})$ differentiable. The objective function is then defined as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{P}, \mathbf{Q}) &= \sum_{(u,i)\in\mathcal{D}} (r_{ui} - \mathbf{p}_u \mathbf{q}_i^\top)^2 + \lambda\Omega(\mathbf{P}) + \lambda\Omega(\mathbf{Q}) \\ &= \sum_{(u,i)\in\mathcal{D}} \left(r_{ui} - \sum_{l=1}^{K} p_{ul} q_{il} \right)^2 + \lambda \sum_{u=1}^{m} \sum_{l=1}^{K} p_{ul}^2 + \lambda \sum_{i=1}^{n} \sum_{l=1}^{K} q_{il}^2 \end{aligned} \tag{2.22}$$

where λ is the parameter controlling the relative importance of least square fitting and regularization.

The optimization of Equation 2.22 is non-convex due to the $\mathbf{p}_u\mathbf{q}_i^\top$ term where both $\mathbf{p}_u$ and $\mathbf{q}_i$ are unknown. The two most popular families of algorithms for minimizing non-convex objective functions in CF are *Stochastic Gradient Descent (SGD)* [Bottou and Bousquet, 2008] and *Alternating Least Squares (ALS)* [Bell and Koren, 2007b]. While SGD is in general easier and faster to converge, ALS can benefit from parallel execution. We present both of them in the following.

Learning the parameters: SGD. SGD [Bottou and Bousquet, 2008] is a version of Gradient Descent (GD), a numerical optimization algorithm, that converges faster than GD by approximating the true gradient using the gradient at a single observation. SGD first initializes the parameters, i.e., user and item latent features $\mathbf{P}$ and $\mathbf{Q}$, randomly. It then loops, until convergence, over all observations $(u, i) \in \mathcal{D}$ and iteratively update the parameters in the direction that reduces $\mathcal{L}(\mathbf{P}, \mathbf{Q})$ as follows:

$$\mathbf{p}_u \leftarrow \mathbf{p}_u - \eta \frac{\partial \mathcal{L}_{u,i}(\mathbf{P}, \mathbf{Q})}{\partial \mathbf{p}_u}, \quad \mathbf{q}_i \leftarrow \mathbf{q}_i - \eta \frac{\partial \mathcal{L}_{u,i}(\mathbf{P}, \mathbf{Q})}{\partial \mathbf{q}_i} \tag{2.23}$$

where η is a hyperparameter called the learning rate and $\mathcal{L}_{u,i}(\mathbf{P}, \mathbf{Q})$ is derived from $\mathcal{L}(\mathbf{P}, \mathbf{Q})$ for one observation $(u, i) \in \mathcal{D}$ as follows: $\mathcal{L}_{u,i}(\mathbf{P}, \mathbf{Q}) = (r_{ui} - \sum_{l=1}^{K} p_{ul}q_{il})^2 + \lambda \sum_{l=1}^{K} p_{ul}^2 + \lambda \sum_{l=1}^{K} q_{il}^2$.

Updates are performed for each element of the matrices $\mathbf{P}$ and $\mathbf{Q}$, and partial derivatives of $\mathcal{L}_{u,i}(\mathbf{P}, \mathbf{Q})$ with respect to single elements are given as follows:

$$\frac{\partial \mathcal{L}_{u,i}(\mathbf{P}, \mathbf{Q})}{\partial p_{ul}} = 2q_{il}(\mathbf{p}_u\mathbf{q}_i^\top - r_{ui}) \; + \; 2\lambda p_{ul} \tag{2.24}$$

$$\frac{\partial \mathcal{L}_{u,i}(\mathbf{P}, \mathbf{Q})}{\partial q_{il}} = 2p_{ul} \; (\mathbf{p}_u\mathbf{q}_i^\top - r_{ui}) \; + \; 2\lambda q_{il} \tag{2.25}$$

The algorithm of SGD is described in Algorithm 2. SGD is sensible to the learning rate chosen, η, and to the initialization of $\mathbf{P}$ and $\mathbf{Q}$ due to the non-convexity of the problem. The complexity of SGD per iteration is $O(|\mathcal{D}|K)$. Several approaches [Recht et al., 2011] have been proposed to parallelize SGD for a faster convergence, but this problem remains challenging and harder than parallelizing ALS.

Learning the parameters: ALS. While the optimization of Equation 2.22 is non-convex, it is quadratic if we fix one of the unknown parameters and thus convex in any single parameter. This setting requires alternating optimization algorithms such as ALS where we repeatedly iterate over the parameters, fix them, and solve the optimization problem. The cost function is improved with each parameter update, and since it is bounded by being non-negative, convergence is guaranteed. ALS [Bell and Koren, 2007b] consists in repeatedly executing, until convergence, the two following steps:

1. Fix the item latent matrix $\mathbf{Q}$. Solve the regularized linear least-square problem for the user latent matrix $\mathbf{P}$. The optimal value for $\mathbf{P}$ is given as follows, where $\mathbf{I}$ is the identity matrix:
$$\mathbf{P} = (\mathbf{Q}\mathbf{Q}^\top + \lambda\mathbf{I})^{-1}\mathbf{Q}\mathbf{R}^\top \tag{2.26}$$

Algorithm 2 Stochastic Gradient Descent (SGD)

Input: Rating matrix $\mathbf{R}$, set of observed ratings $\mathcal{D}$, stopping criterion ϵ, rank K, learning rate η, and regularization parameter λ
Output: Optimal user feature matrix $\mathbf{P}$, optimal item feature matrix $\mathbf{Q}$

Initialize $\mathbf{P}$ and $\mathbf{Q}$, e.g., randomly
while not converged, e.g., $\left\|\mathbf{R} - \mathbf{P}\mathbf{Q}^\top\right\| > \epsilon$, **do**
  for each $(u, i) \in \mathcal{D}$ **do**
    $e_{ui} = \mathbf{p}_u \mathbf{q}_i^\top - r_{ui}$
    $\mathbf{p}_u \leftarrow \mathbf{p}_u - 2\eta(e_{ui}\mathbf{q}_i + \lambda \mathbf{p}_u)$
    $\mathbf{q}_i \leftarrow \mathbf{q}_i - 2\eta(e_{ui}\mathbf{p}_u + \lambda \mathbf{q}_i)$
  end for
end while

2. Fix the user latent matrix $\mathbf{P}$. Solve the regularized linear least-square problem for the item latent matrix $\mathbf{Q}$. The optimal value for $\mathbf{Q}$ is given as follows:

$$\mathbf{Q} = (\mathbf{P}\mathbf{P}^\top + \lambda \mathbf{I})^{-1}\mathbf{P}\mathbf{R} \tag{2.27}$$

Algorithm 3 Alternating Least Squares (ALS)

Input: Rating matrix $\mathbf{R}$ and regularization parameter λ
Output: Optimal user feature matrix $\mathbf{P}$, optimal item feature matrix $\mathbf{Q}$

Initialize $\mathbf{P}$ and $\mathbf{Q}$, e.g., randomly
while not converged do
  for $u \in 1, \ldots, m$ **do** ▷ update users
    $\mathbf{p}_u \leftarrow (\sum_{i \in \mathcal{I}_u} \mathbf{q}_i \mathbf{q}_i^\top + \lambda \mathbf{I})^{-1} \sum_{i \in \mathcal{I}_u} r_{ui}\mathbf{q}_i$
  end for
  for $i \in 1, \ldots, n$ **do** ▷ update items
    $\mathbf{q}_i \leftarrow (\sum_{u \in \mathcal{U}_i} \mathbf{p}_u \mathbf{p}_u^\top + \lambda \mathbf{I})^{-1} \sum_{u \in \mathcal{U}_i} r_{ui}\mathbf{p}_u$
  end for
end while

The algorithm of ALS is described in Algorithm 3. Each user vector update costs $O(|\mathcal{I}_u|K^2 + K^3)$ and each item vector update costs $O(|\mathcal{U}_i|K^2 + K^3)$, assuming a time complexity of $O(K^3)$ for inverting the matrices $(\sum \mathbf{q}_i \mathbf{q}_i^\top + \lambda \mathbf{I})$ and $(\sum \mathbf{p}_u \mathbf{p}_u^\top + \lambda \mathbf{I})$. The complexity of ALS per iteration is then $O(|\mathcal{D}|K^2 + (n+m)K^3)$. Even though the time complexity is high, ALS is well-designed for parallelization [Zhou et al., 2008].

Beyond optimizing the squared loss function. The recommendation problem was historically first formulated as a rating prediction problem (Section 2.1.1), where the main goal is to accurately predict unobserved ratings by performing what is commonly called the *matrix completion* task. Within this scope, the RS performance is measured using the RMSE (Section 2.2.3) for example, and the main objective function to optimize is the squared loss function (Equation 2.21). Since then, the recommendation problem evolved beyond the

TABLE 2.2: Examples of MF models based on different objective functions

Model	**Objective function**
MF [Koren et al., 2009]	ℓ_{SL}^{point}
WRMF [Hu et al., 2008]	ℓ_{SL}^{point} with weights
MMMF [Rennie and Srebro, 2005]	ℓ_{HL}^{point}
BPR-MF [Rendle et al., 2009]	ℓ_{LL}^{pair}
CCF [Yang et al., 2011]	ℓ_{HL}^{pair}
COFI^{RANK} [Weimer et al., 2008]	NDCG loss
CLiMF [Shi et al., 2012]	MRR loss

matrix completion task and several objective functions for training recommendation models were further explored. These functions can roughly be divided into three categories: *point-wise*, *pair-wise*, and *list-wise* objective functions. Point-wise functions, such as the squared loss function, focus on the accuracy of predictions of individual preferences. Pair-wise functions approximate the loss by considering the relative ranking of predictions for pairs of items. Finally, list-wise functions reflect the distance between the complete recommended list and the reference one. While some CF models rely on list-wise functions [Shi et al., 2010], they are not as widely adopted as the other two types of functions and we omit details about them in the following.

We denote by $\ell(.)$ any loss function. A point-wise function evaluated for one data point r_{ui} is denoted by $\ell^{point}(r_{ui}, \hat{r}_{ui})$. Following previous definitions, we have:

$$\mathcal{L}_{u,i}(\mathbf{P}, \mathbf{Q}) = \ell^{point}(r_{ui}, \mathbf{p}_u \mathbf{q}_i^\top) = \ell^{point}(r_{ui}, \hat{r}_{ui}) \tag{2.28}$$

A pair-wise function is evaluated for triples in the form of (u, i, j) where user u is assumed to prefer item i over item j, i.e., $r_{ui} > r_{uj}$, and is denoted by $\ell^{pair}(r_{uij}, \hat{r}_{uij})$ where $r_{uij} = r_{ui} - r_{uj}$ and $\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$. Several options are possible for the definition of the loss function $\ell(.)$ in point-wise and pair-wise settings. A few commonly used possibilities are listed in the following:

- Squared Loss: $\ell_{SL}(y, \hat{y}) = (y - \hat{y})^2$
- Logistic Loss: $\ell_{LL}(y, \hat{y}) = \log(1 + \exp(-y \ . \ \hat{y}))$
- Hinge Loss: $\ell_{HL}(y, \hat{y}) = \max(0, 1 - y \ . \ \hat{y})$

Table 2.2 provides examples of recommendation models using the different loss functions presented, in addition to losses related to the ranking of items in the recommendation list. In general, pair-wise functions are considered to be more suitable for solving the top-N recommendation problem, especially when handling implicit feedback which requires special treatment.

2.5.2.4 Dealing with Implicit Feedback

Most research in the RS field was historically powered by applications where user preferences were collected in the form of explicit ratings. This led to the development of several algorithms that are meant to accurately predict ratings. However, in many real-world applications, only *implicit feedback* that is obtained without user intervention, e.g., purchases, clicks, and bookmarks, is available (Section 2.1.2). This type of feedback introduces new challenges (Section 2.1.3), preventing the direct application of recommendation models developed for the explicit feedback setting. Since implicit feedback is often "positive-only", i.e., only information about liked items is available, this problem is often designated as "One-Class Collaborative Filtering" [Pan et al., 2008].

Dealing with implicit feedback and, more specifically, learning a MF model based on implicit feedback should address two problems: transforming the original set of implicit observations by integrating additional negative feedback and defining the objective function.

Augmenting implicit feedback datasets. Given that implicit feedback is positive-only, only considering the observed feedback assumes that interactions are missing at random, which is not true and results in a bad performance [Marlin et al., 2007]. Several solutions have been proposed to transform implicit feedback datasets. The most widely used can be roughly decomposed in two families. The first one consists in *converting implicit feedback to ratings*. Each observed user-item interaction takes the value of 1 in the rating matrix, i.e., $r_{ui} = 1$. Unobserved interactions can be treated as negative feedback, i.e., $r_{ui} = 0$, resulting in the strategy known as *All Missing as Negative*. Fitting a model to this data without proper regularization will strongly be biased towards negative feedback and will tend to always predict 0 [Rendle et al., 2009]. Another strategy, *All Missing as Unknown*, consists in treating unobserved interactions as unknown by ignoring them. In this case, the model will strongly be biased towards positive feedback and this would result in a trivial solution [Srebro and Jaakkola, 2003]. To avoid the drawbacks of these two extreme cases, the second family of approaches takes an in-between solution by attempting to *discriminate negative items from the unobserved ones*. This is done by randomly sampling negative items [Rendle et al., 2009] or by using weighting techniques [Hu et al., 2008; Pan et al., 2008].

Defining the objective function. Dealing with implicit feedback usually implies solving a top-N recommendation problem where the focus is on the ranking quality of items rather than on the value of predicted ratings. We present in the following two seminal MF approaches proposed to deal with implicit feedback. The first relies on a weighted point-wise function [Hu et al., 2008] while the second on a pair-wise function [Rendle et al., 2009].

Weighted Regularized MF. In Weighted Regularized MF (WRMF) [Hu et al., 2008; Pan et al., 2008], the objective function is a weighted point-wise function where the weight indicates the confidence we have in the corresponding observation. If user u interacted with item i, then we are confident that u likes i. If u never interacted with i, we can make several assumptions with varying confidence levels. The objective function can be written as follows:

$$\mathcal{L}(\mathbf{P}, \mathbf{Q}) = \sum_{(u,i)\in\mathcal{D}} c_{ui}(r_{ui} - \mathbf{p}_u\mathbf{q}_i^\top)^2 + \lambda\Omega(\mathbf{P}) + \lambda\Omega(\mathbf{Q}) \tag{2.29}$$

where c_{ui} is the confidence we have in the observation r_{ui}. [Pan et al., 2008] suggest to set c_{ui} to 1 for positive observations, i.e., $r_{ui} = 1$, and to lower values, e.g., $c_{ui} = 0.01$, for unobserved interactions, i.e., $r_{ui} = 0$. [Hu et al., 2008] consider additional parameters when estimating c_{ui}, e.g., higher confidence for users who have rated more items or higher confidence for items who have been rated by more users. ALS is used for optimization and in cases where c_{ui} is constant for unobserved items, the cost of learning the parameters is $O(|\mathcal{D}|K^2 + (|\mathcal{U}| + |\mathcal{I}|)K^3)$ per iteration.

Bayesian Personalized Ranking. Bayesian Personalized Ranking (BPR) [Rendle et al., 2009] addresses the recommendation problem as a ranking task and relies on a pair-wise interpretation of the user feedback. The dataset is artificially augmented by including potential negative feedback. The goal of BPR is to find a personalized total ranking $>_u \subset \mathcal{I}^2$ for all users $u \in \mathcal{U}$ and pairs of items $(i, j) \in \mathcal{I}^2$ satisfying the properties of a total order. Therefore, BPR uses item pairs as training data and optimizes for correctly ranking them. The training data, denoted by $\mathcal{D}_{bpr}$, is derived from the dataset of positive-only observations and is based on the following assumption: User u who interacted with item i prefers i over every item j for which u did not provide any feedback, i.e., $j \in \mathcal{I} \setminus \mathcal{I}_u$. If u interacted with i and j, it is not possible to deduce the preference for one item over the other. $\mathcal{D}_{bpr}$ is then defined as follows:

$$\mathcal{D}_{bpr} = \{(u, i, j)|i \in \mathcal{I}_u \wedge j \in \mathcal{I} \setminus \mathcal{I}_u\} \tag{2.30}$$

A personalized ranking of items is obtained by optimizing a general criterion called BPR-Opt which can be applied to neighborhood models and MF. BPR-Opt is derived through a Bayesian analysis of the problem where the aim is to maximize the posterior probability. Denoting by Θ the parameter vector of the underlying model, the posterior probability is given by $p(\Theta\ |>_u) \propto p(>_u|\ \Theta)p(\Theta)$. BPR-Opt is formulated as follows:

$$\begin{aligned} \text{BPR-Opt} &:= \ln p(\Theta\ |>_u) \\ &= \ln p(>_u|\ \Theta)p(\Theta) \\ &= \sum_{(u,i,j)\in\mathcal{D}_{bpr}} \ln \sigma(\hat{r}_{uij}) - \lambda_\Theta \left\|\Theta\right\|^2 \end{aligned} \tag{2.31}$$

where $\sigma(.)$ is the logistic sigmoid, $\sigma(x) = \frac{1}{1+e^{-x}}$, $\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$, and λ_Θ designates model specific parameters for regularization. The optimization problem is solved using a SGD on the parameters Θ. Since it is computationally expensive to consider all triples $(u, i, j) \in \mathcal{D}_{bpr}$ due to their large number, triples are uniformly sampled from $\mathcal{D}_{bpr}$ during the learning process.

In its most general form, MF can also be approached by probabilistic models which are introduced in the following section.

2.5.2.5 Probabilistic Models

MF techniques for RS were extended to probabilistic models by two seminal papers [Salakhutdinov and Mnih, 2008a,b]. One of them, Probabilistic Matrix Factorization (PMF) [Salakhutdinov and Mnih, 2008a], is represented as a graphical model in Figure 2.3.

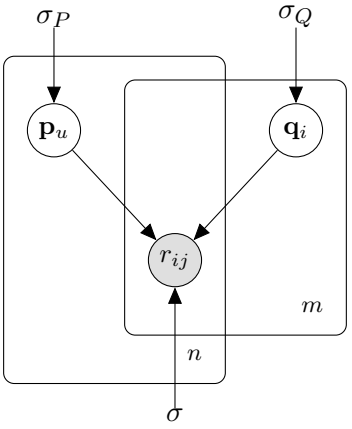


FIGURE 2.3: Graphical model for Probabilistic Matrix Factorization (PMF)

In PMF, we assume Gaussian priors on user and item feature matrices, given the hyperparameters σ_P^2 and σ_Q^2:

$$\mathbf{P} \sim \mathcal{N}(0, \sigma_P^2 \mathbf{I}) \tag{2.32}$$

$$\mathbf{Q} \sim \mathcal{N}(0, \sigma_Q^2 \mathbf{I}) \tag{2.33}$$

Ratings are generated according to a Gaussian with the hyperparameter σ^2 as follows:

$$r_{ij} \mid \mathbf{P}, \mathbf{Q} \sim \mathcal{N}(\mathbf{p}_u \mathbf{q}_i^\top, \sigma^2) \tag{2.34}$$

Having defined I_{ui} equal to 1 if $(u, i) \in \mathcal{D}$ and 0 otherwise, the log of the posterior distribution is given by:

$$\begin{aligned} \ln p(\mathbf{P}, \mathbf{Q} \mid \mathbf{R}, \sigma^2, \sigma_P^2, \sigma_Q^2) = &- \frac{1}{2\sigma^2} \sum_{u=1}^{n} \sum_{i=1}^{m} I_{ui} (r_{ui} - \mathbf{p}_u \mathbf{q}_i^\top)^2 - \frac{1}{2\sigma_P^2} \sum_{u=1}^{n} \mathbf{p}_u^\top \mathbf{p}_u - \frac{1}{2\sigma_Q^2} \sum_{i=1}^{m} \mathbf{q}_i^\top \mathbf{q}_i \\ &- \frac{1}{2} \left(\left(\sum_{u=1}^{n} \sum_{i=1}^{m} I_{ui} \right) \ln \sigma^2 + nK \ln \sigma_P^2 + mK \ln \sigma_Q^2 \right) + c \end{aligned} \tag{2.35}$$

where c is a constant independent of the parameters. Maximizing the log-posterior over user and item latent features is equivalent to minimizing the following objective function:

$$\mathcal{L} = \frac{1}{2} \sum_{u=1}^{n} \sum_{i=1}^{m} I_{ui} (r_{ui} - \mathbf{p}_u \mathbf{q}_i^\top)^2 + \frac{\lambda_P}{2} \sum_{u=1}^{n} \|\mathbf{p}_u\|_{Fro}^2 + \frac{\lambda_Q}{2} \sum_{i=1}^{m} \|\mathbf{q}_i\|_{Fro}^2 \tag{2.36}$$

where $\lambda_P = \sigma^2/\sigma_P^2$, $\lambda_Q = \sigma^2/\sigma_Q^2$, and $\| \cdot \|_{Fro}$ denotes the Frobenius norm. By fixing the hyperparameters $\{\sigma_P, \sigma_Q\}$, we recover an objective function similar to the one in Equation (2.22). Nevertheless, PMF is an extension of MF with probabilistic interpretation allowing further developments of the model, such as a Bayesian version of PMF, BPMF [Salakhutdinov and Mnih, 2008b], or the use of side information [Ma et al., 2008].

2.6 Hybrid Approaches

Since CBF and CF rely on different sources of input to make recommendations, each one of them has its own advantages and weaknesses, i.e., overspecialization for CBF and cold-start for CF. Hybrid methods [Burke, 2002] were then proposed to get the best of both worlds. They assume that various sources of input are available at the same time which allows the use of different recommendation approaches in one framework and avoids the limitations of each approach when used separately. On the other hand, competitions such as the Netflix Prize [Bennett and Lanning, 2007] and KDD cups [Tang et al., 2014] highlighted the fact that the best results are often achieved when different recommendation algorithms are combined in a single model. Hybrid methods can take various forms, and an existing classification covering the main trends of hybrid recommendation [Adomavicius and Tuzhilin, 2005] is presented in the following:

- *Combining separate recommendations.* The utilities predicted separately by several recommendation algorithms are combined to provide a single recommendation, using methods such as linear combination [Claypool et al., 1999] or voting schemes [Pazzani, 1999].
- *Incorporating content-based characteristics to CF approaches.* User-based neighborhood approaches can be for example adapted to compute similarities based on content-based user profiles [Pazzani, 1999].
- *Incorporating collaborative characteristics to CBF approaches.* CF models can be applied to a group of content-based profiles for text recommendation, for example [Soboroff and Nicholas, 1999].
- *Developing a unifying model incorporating content-based and collaborative characteristics.* Many approaches have been proposed within this scope such as a unified probabilistic method for combining collaborative and content-based recommendations [Popescul et al., 2001; Schein et al., 2002].

2.7 Context-Aware Approaches

Further advances in the RS field recognized the importance of contextual information in predicting item utility [Adomavicius and Tuzhilin, 2015]. User choices are not solely guided by a fixed set of preferences related to item features and uncovered by CBF and CF approaches, but strongly depends on their context, their current needs, and the situation they are in. There are several types of contextual information that could help provide better recommendations when taken into account, the main examples being:

- *Location information.* Location information can be useful in a variety of recommendation scenarios [Levandoski et al., 2012; Ye et al., 2011]. User geographical location can have a significant influence on preferences in terms of taste, culture, and habits. It can also indicate the type of the current user activity, e.g., work at the office or leisure in an entertaining place, which in turn affects the set of relevant preferences. In some specific applications, e.g., restaurant recommendation, user location is essential to filter out physically inaccessible options.

- *Temporal information.* Depending on the time of the day, on the day of the week, or on the season, users do not exhibit the same preferences across several domains, e.g., clothing and tourism. Temporal information has thus been exploited to improve the quality of recommendation [Campos et al., 2014; Koren, 2009].

- *Social information.* The user behavior varies according to social factors. For example, a user may make different choices depending on whether he is in the company of friends or family [Adomavicius and Tuzhilin, 2015]. In addition, and especially within the scope of online social networks, behaviors of close users are expected to affect the target user and should be considered when offering recommendations [Ma et al., 2008].

As mentioned in Chapter 1, the vast majority of Context-Aware RS (CARS) adopt the *representational view* of context [Dourish, 2004]. This view makes the following assumptions: (i) context is a *form of information*: It is something that can be known; (ii) context is *delineable*: It is possible to define what counts as the context of activities for some set of applications; (iii) context is *stable*: The elements of context do not vary between instances of an activity within the same application; (iv) *context and activities are separable*: An activity happens within a context. In comparison, the *interactional view* argues that: (i) context is a *relational property* between objects or activities: One dimension may be contextually relevant or not to a particular activity; (ii) the scope of contextual features is *defined dynamically*; (iii) context is an *occasioned property* relevant to particular events; (iv) context *arises from the activity*. While the first view considers that context is a stable feature of the environment that is independent of individuals' actions, the second view assumes a bidirectional relationship between context and activities.

Integrating context into RS requires the consideration of an additional dimension besides the user and item dimensions which are originally encoded in the feedback matrix. The context being itself a multifaceted variable, the representation space of feedback becomes then multidimensional [Adomavicius and Tuzhilin, 2015]. While traditional RS aim to estimate a utility function defined on the space $\mathcal{U} \times \mathcal{I}$ and which observed values are in $\mathbf{R}$, CARS cover the space $\mathcal{U} \times \mathcal{I} \times \mathcal{C}$ where $\mathcal{C}$ represents the set of relevant contextual factors. It is often assumed that context values are atomic and nominal, and that the context entity is the Cartesian product of several contextual dimensions, e.g., $Time \times Company$. Each dimension is a subset of a Cartesian product of some attributes, e.g., $Time \subseteq Year \times Month \times Day$, where each attribute defines a set of values. Three popular paradigms were identified for addressing this multidimensional recommendation problem [Adomavicius and Tuzhilin, 2015]: *contextual pre-filtering* where context is used for input data selection, *contextual post-filtering* where context is used to filter recommendations, and *contextual modeling* where context is directly incorporated into the model.

2.7.1 Paradigms for Incorporating Context

Contextual pre-filtering. In contextual pre-filtering, context drives the selection of data given as input to the recommendation algorithm. User interactions are filtered using contextual factors and the data is then fed to a traditional RS. An example of contextual pre-filtering consists of using *user micro-profiles* where users are split into several sub-profiles, each of them representing the user behavior in a particular context [Baltrunas and Amatriain, 2009]. Prediction within a particular context is done using the corresponding micro-profile instead of the full profile. Similarly to this *user splitting* approach, the *item splitting* technique creates several fictitious items according to the contexts in which they were selected [Baltrunas and Ricci, 2014]. It is based on the assumption that the nature of items may change when available in different contextual conditions. In addition, some approaches proposed to apply user and item splittings sequentially [Zheng et al., 2013].

Contextual post-filtering. In contextual post-filtering, the whole set of interactions is fed to a traditional RS while ignoring the contextual dimensions. The recommendation list delivered in output is then refined using the contextual dimensions [Hariri et al., 2012]. This is done either by removing recommendations that are irrelevant to the context or by adjusting the ranking of items in the recommendation list. [Panniello et al., 2009] provide an experimental comparison of the pre-filtering paradigm and the post-filtering paradigm in several application domains. The empirical results show that the choice of a strategy between pre-filtering and post-filtering depends on the recommendation problem. Nevertheless, one major advantage of both paradigms is their ability to leverage the various traditional recommendation algorithms proposed in the literature.

Contextual modeling. In contextual modeling, the whole set of interactions is exploited to train a multidimensional recommender. Tensor Factorization (TnF) is one of the most successful methods for integrating contextual information into the recommendation model. It can be seen as a generalization of MF in which a multidimensional cube is factorized instead of a 2-dimensional matrix, i.e., the feedback matrix. The additional dimensions correspond to the various contextual dimensions. Instead of learning two latent matrices, i.e., user and item feature matrices, TnF creates a latent matrix for each dimension, i.e., one for the users, one for the items, and one for each contextual dimension. A particularly notable example of TnF for contextual modeling is the *multiverse recommendation* method [Karatzoglou et al., 2010] that adapts the generic TnF approach to the recommendation setting. Another notable technique for contextual modeling is Factorization Machines (FM) [Rendle, 2012] which can be viewed as a generalization of all the previously discussed factorization models. The basic idea is to model ratings as the linear combination of interactions between input variables. Each rating r_{ui} is modeled by a feature vector $\mathbf{x}$ which can be seen as the horizontal stack of the corresponding one-hot-encoding of user, item, and contextual features. The multidimensional space is therefore flattened into a set of $(m+n+d)$-dimensional rows, where d is the number of values of the contextual dimensions. The target variable for each feature vector $\mathbf{x}$ is the corresponding rating value. Rating prediction using *second-order* FM is then performed as follows:

$$\hat{r}_{ui} = w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} w_{jj'} x_j x_{j'} \tag{2.37}$$

where w_0 is a global bias, p is the number of features, i.e., $p = m + n + d$, w_j are first order interaction parameters and $w_{jj'}$ are second order factorized interaction parameters and are defined as $w_{jj'} = \mathbf{v_j} \cdot \mathbf{v_{j'}}$ where $\mathbf{v_j}$ is a factorized vector for feature j. Model parameters, i.e., w_0, the different values of w_j, and each one of the vectors $\mathbf{v_j}$, are learned by optimizing a point-wise objective function defined over the data $\mathcal{D}$. We note that the two models mentioned for contextual modeling were originally designed to handle explicit feedback.

2.7.2 From Context-Aware to Context-Driven Recommender Systems

Recent work [Pagano et al., 2016] highlighted the emergence of the *contextual turn*, creating the need for *Context-Driven RS (CDRS)* for which context is critical rather than additional. CDRS aim to contextualize recommendations, i.e., tailor recommendations to the user intent and situation, rather than personalize, i.e., tailor recommendations to the individual. The main assumption is that users have more in common with other users in the same situation than with the past version of themselves. Recommendations are based on what is going around the user, i.e., the user's situation, and on what the user is trying to accomplish, i.e., the user's intent. CDRS are therefore able to generate recommendations without past information about users, i.e., in the cold-start setting. Several families of RS can be seen as a special case of CDRS. We mention for example *session-based RS* [Quadrana et al., 2018] that defines the context as a series of interactions carried out within a session.

2.8 Conclusion

Summary. In this chapter, we propose an overview of RS and present the problem formulation, challenges, and limitations. We discuss evaluation methodologies and present some recommendation techniques in addition to recent emerging trends. We mention the trade-offs involved with each recommendation approach and the advantages and disadvantages of adopting each one of them for recommendation. Research around RS was always inspired by real-world applications that constantly re-framed the problem and the focus of actual developments. This chapter is also an attempt to highlight this aspect, tracing the evolution of the problem and the considerations from an industry perspective. Nevertheless, developing RS requires the understanding of the user behavior in the corresponding domain and the role the system is expected to fulfill in order to define which criteria to optimize.

Relation to our work. In the scope of this thesis, we focus on the problem of top-N recommendation based on implicit feedback and contextual information. We propose to handle the data dynamics by considering two particular forms of context: *partially observable* and *unobservable* contexts, that cannot be taken into account using traditional methods and that are further detailed in relevant parts of the thesis following the definitions of Chapter 1. *Partially observable* context is studied for the application of hotel recommendation in Part II and *unobservable* context for the application of online recommendation in Part III.

Part II

Partially Observable Context in Hotel Recommendation

Chapter 3

The Hotel Recommendation Problem

This chapter presents the hotel recommendation problem as occurring in real-world applications in the hotel industry and as addressed in this thesis. We discuss its relation with other studied recommendation problems, its characteristics and specific challenges, and the implications on building accurate RS for recommending hotels. We also provide a description of the notion of context as emerging in the studied domain since it is proven to have an important influence on the traveler behavior.

3.1 Introduction

Information Technology (IT) and tourism. While tourism has a significant economic impact, the industry has been witnessing dramatic changes over the past few years [Horner and Swarbrooke, 2016]. The tourist behavior has been shifting with the evolution of the travel experience itself and with the change of expectations and motivations. It exhibits nowadays diverse personas that are complex to decode. In addition, the rapid development of IT has contributed to this radical transformation [Werthner et al., 2015], especially considering the way we access information and purchase products. On the other hand, the IT field is also opening new business opportunities resulting in several value-generating strategies and promising to enhance the tourism experience [Werthner and Ricci, 2004]. The challenges faced by these applications require the combination of research and development at the intersection of several disciplines including computer science, cognitive technologies, and tourism research. Relevant research topics can be divided into five layers [Werthner et al., 2015], covering the tourism ecosystem, which are stated in the following: (i) individual, (ii) group, (iii) corporate, (iv) industry, and (v) government.

The *individual* layer focuses on the interaction between any user from the demand or supply side and any IT service. Given the heterogeneity of users, there is a need for personalized applications and adaptive models in order to enhance the overall experience and reduce the cognitive load of users. *Groups* are formed by two or more users and can be

identified as sharing similar preferences or as being present together in the same situation. Specific services can be developed to support group formations, group experiences, or group decision-making. The *corporate* layer concerns any organization in the tourism domain and covers issues like knowledge management, the understanding of customers, and online reputation. The *industry* layer captures the overall structure of the sector and the interactions between organizations. It highlights most importantly the rise of online travel agencies. Finally, the *government* layer deals with rules and regulations of IT in the tourism domain.

RS in tourism. RS interfere at the *individual* and *group* layers and are becoming essential in the tourism domain. Travelers are actively searching for information to compose their vacation packages and are rapidly facing the curse of information overload [Xiang et al., 2015]. RS are able to support travelers in several ways and to accompany them during the whole experience [Borràs et al., 2014; Felfernig et al., 2007; Gretzel, 2011]. They can recommend destinations that suit the user preferences during the pre-trip destination selection [Ricci, 2002], and then accommodations, attractions, restaurants, events, and points of interest, in addition to personalized routes guiding travelers throughout several attractions [Kurata, 2011; Sebastia et al., 2009]. From a business point of view, RS participate in increasing the loyalty of users by improving their experiences. Therefore, personalization is becoming a priority for organizations, especially in the presence of a fierce competition in the market.

RS in the hotel domain. When making travel plans, accommodation is an essential part of the process and choosing the most appropriate option is an effortful and time-consuming task for most people. As in any industry sector where RS are deployed, deploying RS in the hotel industry facilitates trip planning and helps increase the profitability of hotel companies and the loyalty of users. The particular problem of hotel recommendation is a challenging one that has had relatively little research devoted to it. It raises specific issues that cannot be addressed using traditional approaches for recommendation. Given the user past behavior, the objective is to learn preferences for hotels and use them to perform hotel recommendations. The rest of this chapter focuses on the hotel recommendation problem and its main characteristics. We first start by defining the scope of our work in Section 3.2. Then, Section 3.3 provides a comparison of the problem addressed with other well-known studied recommendation problems and Section 3.4 describes its specific challenges and limitations. We discuss related work in Section 3.5. We define the notion of context as emerging in the hotel domain in Section 3.6 and conclude in Section 3.7.

3.2 Scope of Our Work

The work related to the problem of hotel recommendation reported in this thesis was conducted within the scope of *AccorHotels*[1] which is a leading hotel operator. AccorHotels offers a vast choice of accommodations in more than 90 countries spread across all continents. It operates over 4,500 hotels and gathers over 35 hotel brands represented in Figure 3.1. These brands range from economy to luxury and try to meet the needs of travelers moving around the world for different reasons. In an attempt to retain travelers, the group introduced the *Le Club AccorHotels*[2] loyalty program. Enrolled users benefit from preferential rates, early

[1] http://www.accorhotels.com
[2] http://www.accorhotels.com/leclub

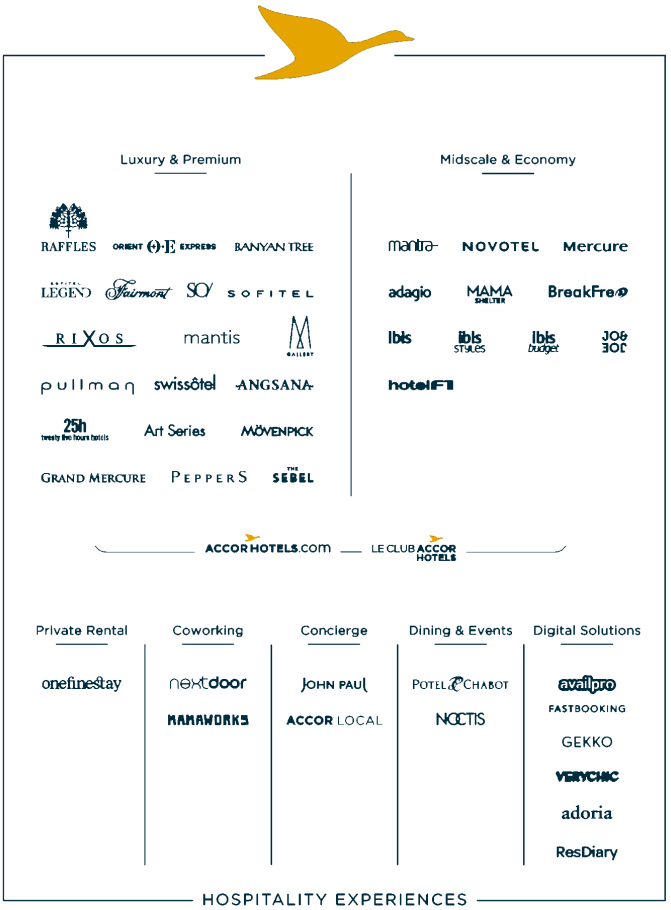


FIGURE 3.1: Brands owned by AccorHotels [AHP, 2018]

access to private sales, advantages from partners, and additional rewards from the network of owned brands.

In order to provide the best experience for its customers, AccorHotels announced in late 2014 the launching of its digital transformation with a five-year investment plan [AHT, Oct 2014]. One of the main pillars of this plan concerns the personalization and tailoring of services to increase customer fidelity. In a world where traveling has become a habit for millions of people, expectations are changing and tourism actors must strive to guarantee customer satisfaction. Existing and new online actors, e.g., *Expedia*[3] and *Airbnb*[4], are also altering the market structure and making it more challenging to attract customers. While personalization has become a must, it is also expected to accompany travelers in every step of their journey. Before the trip, they need assistance to select an appropriate option for accommodation, prepare their stay, and handle transportation concerns for example.

[3]http://www.expedia.com
[4]http://www.airbnb.com

Table 3.1: Proportion of available data by user feature for a set of customers enrolled in the loyalty program

Country	Gender	City	Birth date	Nationality	Profession	Preferences
100%	93.6%	75.6%	16.7%	3.2%	2.3%	0.7%

During the stay, several services can be offered like room service, access to hotel facilities, tour organization, and taxi booking. Feedback is then collected once the stay is over.

Hotel recommendation deals with three major entities: users, hotels, and interactions between users and hotels derived from user behavior and demonstrating the preferences that users hold for hotels. We expand the usage of the term *users* to cover the whole set of *customers*, knowing that AccorHotels is not restricted to an online platform. Registered bookings could have been made on the spot or over the phone for example.

Users. Users who are not enrolled in the loyalty program cannot be uniquely identified due to privacy concerns. In addition, their actions cannot be properly tracked given the multiple booking channels they interact with. They are also not able to receive recommendations. Therefore, our focus is on users enrolled in the loyalty program to which we aim to deliver relevant recommendations given their past behavior. When users register, they are asked to fill in personal information such as the gender, the country of residence, the nationality, and the address, among others, and they can grant consent of use of this information. The user profile can also be filled with preference information including the favorite brand and room options. Except for the country of residence, the fields are not mandatory and the information is often missing or wrong. Table 3.1 presents user features and the proportion of available data for each one of them for a set of 25 million registered users.

Hotels. Each hotel is described by a set of features that we enumerate in the following: the brand, the city and the country where it is located, the number of stars, the segment category, the location category, and available services. These services include Wi-Fi connection, swimming pool, parking, meeting facilities, and children playground. Segment categories cover the luxury, midscale, and economy categories. Location categories include, among others, the following categories: airport, beach, business district, shopping district, and entertainment district.

Interactions. User feedback can be collected through various forms. First, *hotel bookings* constitute a form of implicit feedback. Hotels selected by users are assumed to match their needs and tastes. Hotel bookings can therefore be leveraged to infer user preferences. A booking is mainly defined by the hotel visited, the arrival and departure dates, the number of adults and children accompanying the user, and the amount paid for the stay. Once the stay is over, users are invited to provide explicit feedback concerning their stay in the form of *numerical ratings* given on different aspects of the hotel, e.g., cleanliness, location, and service. However, only a small proportion of users perform this task. Other than being rarely available, there are actually two issues with this type of feedback. First, it is subject to biases that can occur on the user and hotel levels [Koren et al., 2009]: Some users tend to give higher ratings than others for a comparable experience and some hotels are widely perceived as better or worse than others. Second, ratings depend on the definition of the different hotel aspects that each user adopts and the lack of a unified view leads to inconsistencies in

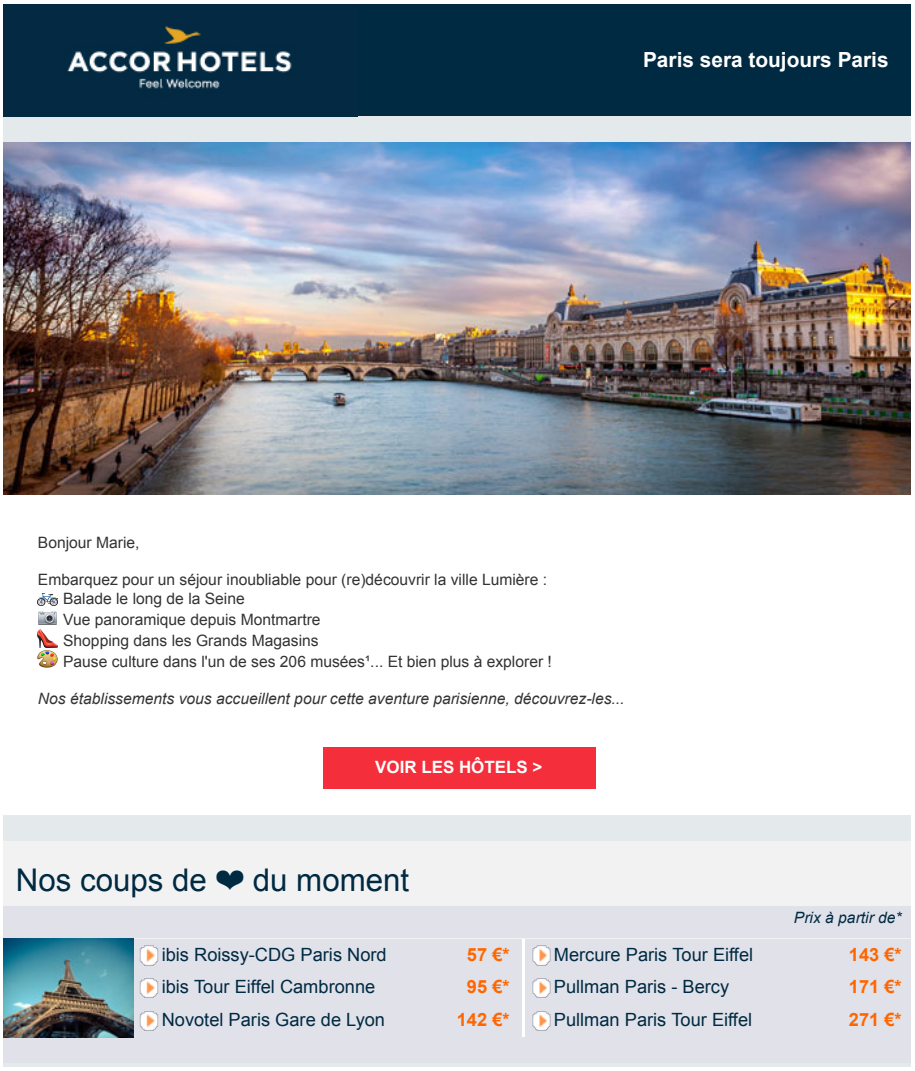


FIGURE 3.2: Example of an email received by *Le Club* customers containing promotional offers and hotel recommendations

ratings. On the other hand, the development of AccorHotels' online platform to book hotels allowed the collection of additional feedback such as *clicks* on hotel items, *viewing* of hotel descriptions, and destination *selections*, among others. Clickstream analysis being properly adopted only recently, the feedback collected is limited to a relatively short period of time and is not sufficient for the studied problem. Due to the previously mentioned reasons, we only rely in our work on hotel bookings as indicators of user preferences. We clearly state our problem in the following, based on the notations introduced in Section 2.1.4.

Definition 3.1. The hotel recommendation problem. Given a set of users $\mathcal{U}$, a set of hotels $\mathcal{I}$, and the set of bookings encapsulated in the feedback matrix $\mathbf{R}$, the goal is to provide a ranked list of N hotels for each user. These N elements represent hotel recommendations that the user is expected to be potentially interested in.

TABLE 3.2: Statistics of the booking dataset, AH, used for the hotel recommendation problem

Dataset	# users	# items	# transactions	Period
AH	7,802,637	4,574	34,709,006	4 yrs

Recommendations. One way to deliver hotel recommendations is through emailing campaigns, promoting a set of interesting hotels to the user in addition to tailored offers or services (Figure 3.2). Recommendations can also be displayed on website banners and potentially checked by the user when visiting the online platform. The first option performs proactive recommendations while the second one provides suggestions only when users access the service or query the system. In addition to selecting appropriate items, the challenge with proactive recommendation is to balance between pushing recommendations and not overly burdening users with a stream of suggestions. The system should learn to identify relevant contextual situations when recommendations should be made.

In the context of this thesis, we rely on offline evaluation to measure the quality of recommendations, mainly due to the cost of performing online evaluation (Section 2.2). The experimental protocol is further detailed in relevant parts. Comparing several recommendation approaches requires a method to evaluate the statistical significance, i.e., how significant the improvement of one approach versus another one is. We perform paired t-tests [Montgomery and Runger, 2010] and report statistically significant differences when the result falls within the 95% confidence interval. The dataset used to perform offline evaluation is introduced in the following.

Dataset. The main dataset used in our work on hotel recommendation, denoted by AH, is obtained from AccorHotels' databases by considering bookings done between 2012 and 2016. We select customers enrolled in the loyalty program and having done at least one booking as the set of users and the hotels they visited as the set of items considered for recommendation. The dataset consists of around 7.8M users, 4.5k hotels, and 34.7M bookings, and statistics are summarized in Table 3.2 for the reader's convenience. We present in the following the characteristics of the studied problem, starting with a comparison with recommendation in other domains.

3.3 Comparison with Recommendation in Other Domains

The problem of recommendation has been studied in its most generic form and also applied to specific domains including the movie, music, book, product, and Point Of Interest (POI) domains, among others. The decision-making process carried out when choosing a hotel is different than the one carried out when choosing a movie to watch, a song to listen to, or a product to buy. Understanding these differences is essential in order to design appropriate recommendation models and to benefit from existing advances in the field. The rest of this section compares the hotel recommendation problem with the following well-known problems: recommendation of tangible goods and multimedia items, and recommendation of Points Of Interest (POIs).

Recommending tangible goods and multimedia items. In the case of a movie or a product selection process for example, users are aware that they have the possibility to switch between movies or to return a product at a minimal charge if they are not satisfied with it. This is not the case when selecting accommodations given that users do not recover easily after a bad choice of accommodations. Managing to change hotel bookings once on a vacation or a business trip is not straightforward. It usually incurs significant costs and can spoil the travelers' trips. The availability of hotels is also not guaranteed given the limited number of rooms. The high-stakes nature of the problem leads to a more challenging and complex decision-making process since customers can often have high levels of insecurity during purchase [Zoeter, 2015]. Furthermore, traveling is not a frequent activity, especially compared to other activities such as watching movies or listening to music, and many people book a hotel only once or twice a year [Bernardi et al., 2015]. As a result, the available feedback is sparse which makes it harder to extract relevant user preferences. Another difference relies on the fact that decisions are made in a very specific context based on a dynamic inventory [Cremonesi et al., 2014]. The issue of the limited availability of resources does not arise in the movie or the music domain where the resource is usually available to an unlimited number of users. The price of the options also varies over time, which requires taking into account the context of decisions.

Recommending Points Of Interest (POIs). With the emergence of Location-Based Social Networks (LBSN), users easily post content associated with their location and share the positions of the places they are visiting. The availability of the vast amount of users' visiting history has led to an extensive study of the problem of POI recommendation [Liu et al., 2017]. Given the visiting history of users, the goal is to recommend for each user new POIs that he will be likely visiting in the future. Compared to the classical problem of recommendation, POI recommendation is greatly affected by *geography*, *time*, and *social relations*. First, geographical distances separating POIs have a significant influence on the user behavior [Ye et al., 2011]. Intuitively, users tend to visit POIs that are either close to their place of residence or far from it but close to POIs they are in favor of. Therefore, POIs visited by one user are rather grouped geographically. Power law distribution is usually used to model the visiting probability to the distance between a couple of POIs. Second, users may adopt different behaviors with respect to time and POIs may be accessible or not at different time periods. Finally, and given that users share their activity on platforms where they are interconnected via social links, social friends can be explored to perform recommendations. In particular, relevant information can be extracted from friends who have close social ties or who exhibit similar behavior with the target user [Ye et al., 2011].

While hotel recommendation can also be influenced by almost the same factors, the user behavior in the hotel domain follows different dynamics leading to a different set of challenges. POIs are usually visited successively and the distance between two consecutive visited locations is taken into account when evaluating possible recommendations. In comparison, hotel visits done by one user are, in most cases, independent and separated by a return to the traveler's residence. As a result, the assumptions made in POI recommendation concerning in particular the geographical influence do not hold in the case of hotel recommendation. In addition, information about social relations between users is missing and cannot be leveraged to improve recommendations. Users adopt different attitudes in each domain and consequently, specific challenges arise in each one of them. We present in the following the challenges and limitations occurring in the hotel domain.

FIGURE 3.3: Percentage of users enrolled in the loyalty program per number of bookings made, represented by *filled bars*, and per number of distinct visited hotels, represented by *empty bars*, on a period of four years

3.4 Challenges and Limitations

The decision-making process for selecting the appropriate accommodation is more complex than the one for acquiring tangible goods, for example. As a result, the hotel recommendation problem faces particular challenges making the direct application of classical recommendation approaches insufficient.

Sparsity. It is well-known that a common problem from which suffer most RS is data sparsity (Section 2.1.3). People only interact with a small number of items and the collected feedback is usually sparse. However, this problem is aggravated in the hotel domain. Traveling is not a frequent activity and many people book a hotel only once or twice a year. The available feedback is not always enough to extract relevant user preferences which tends to reduce the quality of recommendations. Figure 3.3 shows, in filled bars, the percentage of users enrolled in the loyalty program per number of bookings made between 2012 and 2016, highlighting the fact that more than 90% of users have made less than five bookings during the considered time period. In addition, some users are used to return to the same hotels they have visited in the past, generating repeated and less diverse feedback. This can be observed by examining the proportion of users per number of visited hotels, represented by empty bars in Figure 3.3. There are more users who have visited one hotel than users who have made one booking, and there are fewer users who have visited more than 10 different hotels than users who have made 10 bookings.

An extreme case of the data sparsity occurs when new users or new hotels are introduced to the system. This is known as the cold-start problem (Section 2.1.3). Several solutions have been proposed in the literature to temporarily deal with cold users until enough information is gathered (for example [Saveski and Mantrach, 2014; Schein et al., 2002]), under the assumption that users cannot return to the cold status. This assumption is not valid in the hotel domain: The cold-start problem experienced is relatively different from the classical one and is framed as the *continuous cold-start* problem [Bernardi et al., 2015].

Continuous cold-start. In the hotel domain, users remain cold for a long time, and even after collecting enough feedback, they may return to the cold status in specific situations, leading to the continuous cold-start setting. This formulation was first introduced by a major online travel site [Bernardi et al., 2015]. The problem is known to arise in the hotel domain in the three following cases:

- *The user is newly introduced to the system.* This case is equivalent to the classical cold-start problem and is due to the fact that many users stay cold for a long time since they only travel and book hotels a few times each year.
- *The user changes his interest.* This case is related to the volatility of preferences. The user interest changes over time bringing him back regularly to the cold status. For example, a phenomenon is observed when people's standards evolve, moving from booking hotels in one segment, e.g., economy segment, to booking hotels in a higher one, e.g., luxury segment.
- *The user behavior changes depending on the context he is in.* Several contextual factors, such as the location, the weather, and the intent of the trip, highly influence travelers' decisions. The same user can behave differently when choosing a hotel to spend the summer vacation with his family or to attend business meetings.

Classical Content-Based Filtering (CBF) and Collaborative Filtering (CF) approaches cannot address the continuous cold-start problem since they do not take into account the volatility of user preferences and relevant contextual factors. While the continuous cold-start problem has not been specifically addressed in the literature, several approaches are able to cope with each of the described aspects. Hybrid approaches [Burke, 2002], combining CBF and CF, deal with the classical cold-start problem. Time-aware RS [Campos et al., 2014] are able to model the evolution of user preferences by considering the chronologically ordered history of user transactions. In particular, Context-Aware RS (CARS) [Adomavicius and Tuzhilin, 2015] offer a promising way to address the continuous cold-start problem [Bernardi et al., 2015]. Sparsity is addressed by introducing information about the user's environment and recommendations are driven by the actual context of the user. Preferences are also modeled according to the context and each behavior is associated with a specific situation. An appropriate solution requires the identification of the contextual factors influencing users, the collection of related data, and the design of models integrating these factors.

3.5 Related Work

Several solutions have been proposed to assist tourists and travelers in their planning process [Felfernig et al., 2007; Gretzel, 2011; Kabassi, 2010]. These solutions aim to recommend one item or a package of items including destinations, attractions, accommodations, and activities, among others. However, the particular problem of hotel recommendation has had relatively little research devoted to it. In this section, we review existing work related to this problem. We first present existing approaches according to the exploited data source, and then discuss the problem of bounded resources and the related problem of destination recommendation.

Explicit feedback for hotel recommendation. Explicit feedback in the form of textual reviews and ratings has been leveraged for hotel recommendation. In [Levi et al., 2012], a cold-start RS relying on textual reviews is developed. The assumption made is that users favor reviews written by users with the same trip intent, e.g., business trip or leisure trip, of similar backgrounds, i.e., nationality, and with similar preferences for hotel aspects, e.g., location, room, and service. These elements are used to measure similarities between users and define context groups. Hotels are modeled as feature vectors by exploiting the words used in the corresponding reviews. Features are additionally extracted for each travel intent, nationality, and hotel aspect. Recommendation is then performed by combining preferences of users who are similar to the target user with the importance assigned to each feature by users. The dataset used was extracted from two travel search engines, *Tripadvisor.com* and *Venere.com*. A crowdsourcing experiment based on Amazon Mechanical Turk showed that common traits for visitors of any hotel can be identified by mining reviews.

Another framework for hotel recommendation was proposed in [Zhang et al., 2015] and leverages textual reviews and ratings. The authors propose a hybrid approach where users and hotels are first modeled in latent topic spaces based on the reviews given by users for hotels. Similarities between users and hotels are then derived from these models. Rating prediction is performed by using Matrix Factorization (MF) and by adding a constraint that enforces the learned models for each pair of users and hotels to be close if their computed similarity is high. The predicted ratings are then modified according to the user's travel intent that is explicitly provided through the proposed framework. Finally, diversity techniques are used to optimize the ranking of the recommended list by removing redundancies while maintaining relevance. To validate the proposed approach, experiments were conducted on a dataset from the travel search engine *Ctrip.com*, and errors in rating prediction were reported. On another note, explicit feedback in the form of ratings assigned to various hotel aspects, e.g., location, cleanliness, services, and rooms, were also exploited for hotel recommendation. In [Nilashi et al., 2015], a 3-dimensional tensor is created by associating the first dimension with users, the second one with items, and the third one with hotel aspects. The tensor is used to cluster users and a dimensionality reduction technique is applied within each cluster. Neural networks integrating fuzzy logic principles are then trained to predict overall ratings in each cluster. Experiments were conducted on datasets from *Tripadvisor.com* and proved the effectiveness of the proposed method and its several components for improving the accuracy of multi-criteria prediction.

Implicit feedback for hotel recommendation. Analyzing the user behavior can indirectly reflect his opinion, and implicit feedback, which is normally more abundant than explicit feedback, can be leveraged to infer user preferences [Hu et al., 2008]. Booking transactions, i.e., hotel visits, are an example of implicit feedback collected in the hotel domain. In [Saga et al., 2008], a preference transition network is built based on user bookings. It is represented by a graph with hotels as nodes. Edges are established between pairs of hotels depending on the likelihood of users who booked one hotel to book the other one. The recommendation phase consists of two steps. First, the user selects an initial hotel, generating a list of candidate hotels. Second, candidate hotels are scored by considering the indegree and outdegree in the graph, in addition to a factor accounting for the number of times the item was recommended and attempting to promote novelty in recommendations. The system was validated using bookings from a real hotel reservation service.

Contextual factors for hotel recommendation. Like in any other domain, hotel recommendation can benefit from incorporating relevant contextual factors affecting users [Adomavicius and Tuzhilin, 2015]. Performing contextual recommendations requires the definition of the notion of context and assessing the importance and relevance of each factor. When addressing the related problem of lodging recommendation in [Sanchez-Vazquez et al., 2017], the authors designed a context-aware approach inspired by socio-economic analyses of user behavior in sharing economy marketplaces. The proposed approach integrates features capturing five aspects of user's consumption behavior: the *perceived value* or the trade-off between benefits and costs, the *perceived risk* in choosing an option rather than the others, the *price sensitivity* or the extent to which the price affects the behavior, the *perceived authenticity* of the experience it could provide, and the *electronic word-of-mouth*. The evaluation was done using a publicly available dataset from *Airbnb*, showed improvements compared to other recommendation techniques, and enabled the study of the discriminative power of the different features.

Bounded resources in hotel recommendation. As mentioned before, recommendation in the tourism domain in general and in the hotel domain in particular exhibits specific characteristics that differentiate it from other domains. In particular, each item, i.e., hotel, has a maximum capacity that cannot be exceeded and hotels cannot be consumed by an unlimited number of users. Recommending with bounded resources has first been studied in [Cremonesi et al., 2013, 2014]. Offline and online experiments were performed to evaluate personalized and non-personalized algorithms in the *low season* setting, i.e., season where all the items are available, and in the *high season* setting, i.e., season where most popular items are unavailable. Results showed that popularity-based methods outperform personalized methods in the low season setting: Popular hotels, i.e., those that are rated the highest by most people, are the best choices for most users. This can be due to the fact that the opinion of the crowd has a strong influence effect which is sometimes stronger than individual preferences. Personalized approaches perform better in the high season setting: When the most popular hotels become unavailable, user preferences drive the decision. Considering hotel availability is therefore essential to determine the recommendation approach to use.

Destination recommendation. The selection of an option for accommodation depends generally on the choice of the destination to visit, and thus, the problem of hotel recommendation is related to the one of destination recommendation. [Kiseleva et al., 2015] investigate basic recommendation strategies for suggesting destinations to users in the context of *Booking.com*, the travel search engine. In the setting they define, the user provides a list of activities that he wants to do during the trip and the system is expected to propose destinations matching these activities. The set of possible activities is defined and fixed, and users who have visited hotels of a specific destination are asked to endorse it with representative activities. Strategies for ranking destinations include the one performing it randomly, the one based on the popularity of destinations, and the one based on Naive Bayes. These strategies are compared with the baseline used at *Booking.com* and an online A/B test showed that Naive Bayes significantly increases user engagement. Further advances in this direction extended the recommendation approach by considering contextual factors, in order to address the continuous cold-start problem [Bernardi et al., 2015]. The work in [Kiseleva et al., 2016] investigated how to leverage context information for recommendation in cases where user history is not available, e.g., users not logged in or new to the system. The main assumption made is that users give similar endorsements in similar situations, a situation

being defined by a set of contextual factors, e.g., time, location, device type, and browser. Reviews are contextualized with respect to the defined factors. Typical user situations are detected and constitute what is called *contextual user profiles*. Recommendation models are then derived for each contextual user profile. When a user gets access to the system, the contextual information is used to map him to a contextual user profile and the corresponding model is used for recommendation. Experiments were conducted within a production A/B testing environment at *Booking.com*, comparing the contextual approach proposed with a non-contextual one. Results showed that the contextual approach substantially increases user engagement.

Relation to our work. The work presented in this thesis and related to hotel recommendation differs from previously proposed approaches in multiple ways. While previous work has privileged the use of explicit feedback, e.g., ratings and reviews, we rely on implicit feedback, i.e., bookings, since it is much more available and does not require additional efforts from the user side [Amatriain and Basilico, 2016]. We address the problem as it is delimited by real-world settings which has been rarely reported in previous work. However, we do not specifically address the problem of bounded resources. Simple solutions can be implemented to cope with this problem, alternating between different recommendation approaches depending on the availability of hotels [Cremonesi et al., 2014]. In an attempt to address the continuous cold-start problem [Bernardi et al., 2015], we design novel approaches integrating multiple contextual factors that affect the decision-making process of users. Context-aware techniques have been previously exploited for recommendation in the tourism sector [Van Setten et al., 2004] but not in the hotel sector. In addition, the focus has been mostly on context in mobile applications which definition does not cover all the factors we exploit in our work. Defining context is an important question itself since the quality of recommendation depends on the relevance of the contextual factors considered. We present in the next section the notion of context as it appears in the hotel domain and we discuss how it affects the user behavior.

3.6 Context in the Hotel Domain

Following the definition introduced by [Dey, 2001], context is *"any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application."*. A concrete framing of the notion of context in a specific recommendation domain requires understanding the decision-making process carried out during the task of selecting an option.

Decision-making process. The nature of tourism products purchased by customers forces a complex decision-making process in which customers are highly involved and committed [Horner and Swarbrooke, 2016]. Customers can have high level of insecurities during purchase since they cannot try the products before the definite selection. In addition, traveling is often an important event in people's lives and decisions have a considerable emotional significance for the people involved. In general, the decision is driven by a number of factors related to customers and to external parameters heavily influencing them and which are beyond their control. Figure 3.4 illustrates some of these internal and external factors [Horner and Swarbrooke, 2016].

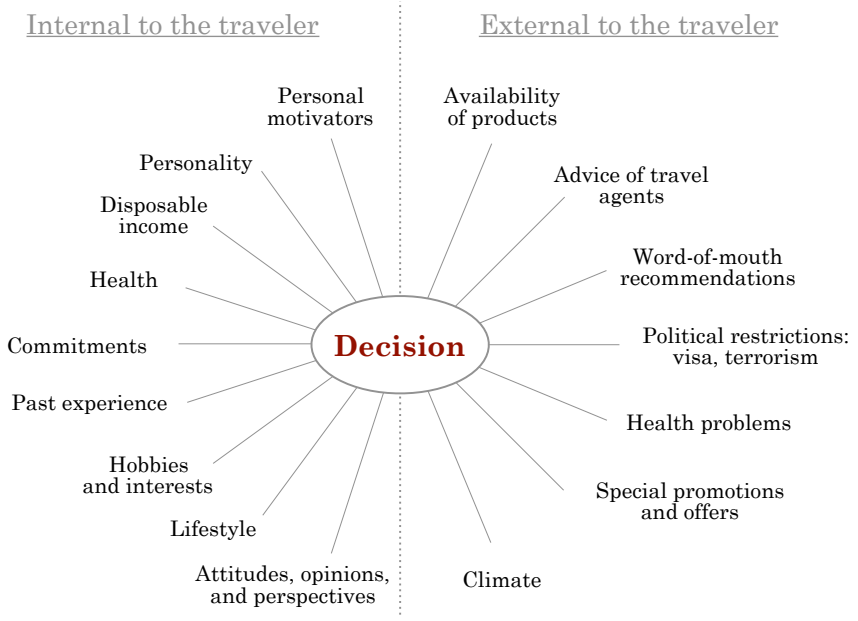


FIGURE 3.4: Internal and external factors influencing the traveler's decision. Figure adapted from [Horner and Swarbrooke, 2016].

Categories of context. A more refined classification of contextual factors is presented in [Adomavicius and Tuzhilin, 2015; Fling, 2009] and introduces four broad categories of context: physical context, social context, interaction media context, and modal context. While this classification was initially meant to frame context in mobile applications [Fling, 2009], these categories can also be adapted to other types of applications. We present them in the following and discuss how they are observed in the hotel domain.

- *Physical context.* The physical or environmental context includes the user location which greatly influences decisions and actions. It also covers the time, activity, weather, and temperature, in applications where these factors are relevant. Choosing a hotel to book depends on the destination the traveler is planning to visit which selection is itself affected by the traveler's country of residence, the season, and the weather, for example.

- *Social context.* The social context represents the presence and role of other people around the user. Users can be strongly affected by other people during the decision-making process. This occurs in cases where a user is booking a vacation for a group of people including himself. It also occurs when the decision is guided by other people's past experiences and by the word-of-mouth or word-of-mouse including reviews posted on social media.

- *Interaction media context.* This category of context describes the device used to access the system and the type of channel used to deliver personalized recommendations. Users may have different behaviors when browsing on their mobile or on their personal computer. Different strategies of recommendations may then be adopted to address the different user attitudes. This problem is not addressed in our work, mainly because this type of information is missing or not exhaustive across the data we handle.

- *Modal context.* The modal context covers the state of mind of the user, his goals, and his mood. It plays an important role in the selection process. As an example, a single user behaves differently when organizing a business or a leisure trip.

Context at purchase vs. context at consumption. In applications where the actual consumption of the item is close in time to its purchase, e.g., movie and music streaming applications, the context is expected to remain stable between the two events. Therefore, contextual factors are evaluated at the moment of purchase and used to generate recommendations adapted to the context of consumption. This is obviously not the case when purchasing tourism products in general and hotels in particular. The context of the user when searching and booking a hotel is significantly different from his context during the trip and during his visit to the hotel. However, elements from both contexts are relevant to the final decision. The location of the user during the purchase, e.g., his country of residence, affects the chosen destination and the subset of candidate hotels that is considered. In addition, the location of the user during consumption, i.e., the destination, affects the choice of the hotel since hotels do not have the same characteristics in all regions. Therefore, the environments of the user when choosing the hotel and when visiting it guide the decision-making process.

3.7 Conclusion

We present in this chapter the problem of hotel recommendation with its characteristics as emerging in real-world applications and in the industrial context we consider in this thesis. We discuss how this particular problem differs from other recommendation problems addressed in the literature, making the use of classical recommendation approaches insufficient. The decision-making process carried out when choosing a hotel is complex: Users can have high level of insecurities and organizing trips involves a considerable emotional significance. Hotel recommendation also suffers from the continuous cold-start problem. In fact, users recurrently fall into the cold-start problem due to the rarity of the traveling activity, the volatility of preferences, and the multifaceted behavior. We discuss these limitations and their implications, and we review existing work related to hotel recommendation. Given that contextual information can be exploited to address the problem, we provide a framing of the broad notion of context as it appears in the hotel domain.

Conventional context-aware approaches adopt the *representational* view of context [Dourish, 2004] which assumes that context is represented by a predefined set of observable static attributes where all possible values are fixed, known, and atomic. In our work, we argue that the context variable emerging in the hotel domain is *partially observable*. This notion of partial observability is related to context accessibility, relevance, acquisition, and modeling, as defined in Chapter 1. It requires the development of appropriate methods accounting for the different categories of context, which we provide in Chapters 4, 5, and 6, alongside a characterization of the contextual factors considered.

Chapter 4

Leveraging Explicit Context

This chapter presents the industrial solution we designed for hotel recommendation [Al-Ghossein et al., 2018d], combining several context-aware recommendation models in order to alleviate the shortcomings of using traditional approaches as discussed in Chapter 3. The *physical*, *social*, and *modal* contexts of travelers are leveraged to improve the performance of the RS, and information related to the geographical and temporal dimensions, textual reviews, and the trips' intents is integrated into the system. Contextual information considered in this chapter is *explicitly* provided by users and is directly related to at least one of the RS entities, i.e., users, hotels, or user interactions with hotels. On the other hand, it is *partially observable* since the trips' intents are not observed at the moment of recommendation but are only recorded for past interactions. The relevance of each contextual factor with regards to target users is also unknown while affecting the overall performance of the RS. The proposed RS addresses these challenges and is validated on real-world datasets.

4.1 Introduction

Choosing the most appropriate option for accommodation when making travel plans is an effortful and time-consuming task for most people. RS rely on personalization and deploying them in the hotel domain facilitates trip planning. As discussed in Chapter 3, the hotel recommendation problem is different from other recommendation problems studied in the literature and faces, in particular, the continuous cold-start issue [Bernardi et al., 2015].

A promising way to address this issue is to leverage Context-Aware RS (CARS) [Adomavicius et al., 2011]. These systems take into account the contextual factors that affect travelers' choices. Recommendations are driven by the context and by the behavior of other users when they were in similar contexts. Building a robust CARS requires identifying the relevant contextual factors and designing models that integrate them. A survey conducted among domain experts reveals that the traveler's decision-making is mainly sensitive to different types of context: the physical, social, and modal contexts, introduced in Section 3.6.

In this chapter, we propose context-aware models for hotel recommendation that take into account the multiple types of context in order to cope with the limitations of only

relying on booking data. These models outperform state of the art methods and integrate geographical and temporal dimensions, textual reviews extracted from social media, and the trips' intents. We present an industrial solution combining the proposed models and handling each user segment differently. Experiments prove the interest of integrating contextual factors to improve the quality of recommendation and show how the performance differs based on the targeted user segment. We also evaluate traditional recommendation methods on a real-world dataset and show how they compare when applied to the particular problem of hotel recommendation.

Notations. The notations adopted throughout the chapter are those introduced in Section 2.1.4. To generate recommendations for a target user $u \in \mathcal{U}$, we compute the *relevance scores*, denoted by $\hat{r}_{ui}$, for each hotel $i \in \mathcal{I}$. Hotels are then ranked by decreasing order of $\hat{r}_{ui}$, and the top-N hotels of the list, i.e., the N hotels that score the highest, are selected for recommendation. The key difference between the methods presented in this chapter resides in the way $\hat{r}_{ui}$ is computed.

The remainder of the chapter is organized as follows. In Sections 4.2, 4.3, and 4.4, we present the recommendation models designed, taking into account the physical, social, and modal contexts of users, respectively. Section 4.5 gives an overview of the RS developed. Experiments and results are presented and discussed in Section 4.6. Finally, Section 4.7 concludes the chapter.

4.2 Influence of the Physical Context

Motivation. The first factor we consider in the physical context is the geographical location of hotels and the users' country of residence. Users we handle come from all over the world and hotels are spread in more than 90 countries. Choosing a hotel is highly correlated with the choice of the destination to visit. In addition, the user's country of residence is important in the sense that residents of one country share the same culture and tend to have similar tastes when it comes to traveling. Furthermore, as noticed in our datasets, the large majority of users' bookings are in hotels located near their country of residence.

The second factor we consider is temporality: The timing of a trip has also an impact on the users' final destinations. Leisure trips, usually organized during holidays, are meant to discover new trendy places while business trips occur almost all year long. The seasonality factor is also relevant. Some users are used to organize trips to the beach in the summer and to ski resorts in the winter. Another example of seasonality influence is observed when residents of cold countries tend to choose warm destinations in an attempt to escape the freezing weather.

To illustrate these insights, Table 4.1 and Table 4.2 show the percentage of bookings in selected destinations and during specific periods of one month made by Australian and French residents respectively. In our datasets, we notice that the majority of Australian residents' trips involve close destinations. However, the percentage of visits to Europe is significantly higher in June than in December, thus highlighting the temporal influence. We note that the number of overall bookings in all periods is comparable. With respect to French residents, we notice that the number of visits to the Paris region in February is

TABLE 4.1: Proportion of bookings made by Australian residents in selected destinations during periods of one month

Destinations	**March**	**June**	**September**	**December**
Oceania	76%	69%	71%	78%
Europe	4%	12%	12%	5%
Asia	9%	9%	8%	9%

TABLE 4.2: Proportion of bookings made by French residents in selected destinations during periods of one month

Destinations	**February**	**May**	**August**	**November**
Paris Region	34%	30%	22%	37%
South of France	22%	24%	33%	23%
Spain	0.5%	1%	2%	0.7%

greater than in the month of August. In addition, they tend to visit the south of France and Spain more frequently during summer. Given these observations, we conclude that geography and temporality have an influence on the global behavior of residents of the same country with respect to the visited destinations.

Approach. To integrate these ideas, we introduce `localRS` where we propose to cluster countries of residence, and consequently users, based on the destinations visited by its residents each period of the year. The main idea is to build a local recommendation model for each cluster instead of building a single global model for all users. Local models better reflect the users' interests since they cluster together those having similar tastes regarding visited destinations. They are also able to capture new travel trends that start in a given region. The concept of building multiple local models instead of one global model has been proposed in previous work [Christakopoulou and Karypis, 2016; Lee et al., 2014]. We omit details about related work on local models in this chapter, only to mention that the approaches previously proposed cluster users and items based on the observed behavior, i.e., user interactions, and are therefore subject to limitations in sparse conditions.

In the process of clustering, a country of residence is represented by a set of features including a feature for each destination country each month of the year. Destination countries considered are countries where hotels are available. The value of a feature is equal to the proportion of bookings made by residents in the destination country in the defined period. We apply the k-means clustering technique [Hartigan and Wong, 1979] and create k clusters of countries where residents have similar behaviors with respect to visited destinations and periods of visits.

The k-means clustering is a popular technique for cluster analysis in data mining. It aims to partition observations into k clusters where each observation is assigned to the cluster with the nearest mean. It assumes a Euclidean space and that the number of clusters, k, is known in advance. While the problem is computationally difficult, heuristic algorithms are usually employed and converge to a local optimum. In the basic version of the k-means algorithm, we initially choose k points that are likely to be in different clusters and designate each one of them as the centroid of a cluster — there are several ways to do this initial selection that

TABLE 4.3: Examples of clusters of countries which residents have similar behaviors regarding visited destinations and periods of visit (non-exhaustive list per cluster)

Cluster 1	Cluster 2	Cluster 3	Cluster 4
France	Italy	United States	UAE
Belgium	Germany	Canada	Bahrain
Andorra	Switzerland	Peru	Kuwait
Guinea	Netherlands	Portugal	Qatar
Guyana	Greece	India	Morocco
Algeria	Romania	China	Spain
Tunisia	Russia	South Korea	Tanzania
Senegal	Turkey	Vietnam	Cuba

will not be discussed here. Then, for each remaining point, we find the the closest centroid and add the point to the corresponding cluster. After iterating over all points, the clusters' centroids are adjusted. This process is repeated until convergence.

Choosing the value of k is important. If k is too small, the recommendation algorithm is sensitive to noise and if k is too large, we may be missing out on users who could be relevant for the generation of recommendations. In our setting, the value of k is finally set following extensive experiments.

Table 4.3 shows examples of computed clusters and a list of some of the countries included in each cluster. The clustering technique as described here is dependent on two factors. First, it relies on the number of users present in each country and whether they constitute a representative sample of the whole population when it comes to the traveling behavior. Second, the technique depends on the worldwide availability of hotels, considering that the density of hotels is not comparable in all countries.

4.3 Influence of the Social Context

With the evolution of social platforms, most people rely on other people's experiences and reviews to choose a hotel. Studies in the tourism sector have indicated that travelers perceive online reviews as more trustworthy than information provided by official websites [Filieri et al., 2015; Litvin et al., 2008]. Consequently, we integrate into the RS textual reviews posted online as an additional source of information to describe hotels. The reviews are anonymous as we are not able to match the users leaving reviews on websites and our target users. Introducing content description is a way to avoid overfitting given the sparse data. It also enables recommending hotels that are newly introduced to the system [1] when reviews are available. The proposed approach, which we refer to as `CTRk+`, is based on Collaborative Topic Regression (CTR) [Wang and Blei, 2011], also known as Collaborative Topic Modeling, which is a hybrid recommendation method. While several other works have exploited content for recommendation [Agarwal and Chen, 2009, 2010; Wang et al., 2015], CTR offers one of the most prominent framework that is presented in the following.

[1] http://pressroom.accorhotels-group.com/accorhotels-officially-welcomes-fairmont-raffles-and-swissotel

4.3.1 Collaborative Topic Modeling

CTR [Wang and Blei, 2011], initially introduced to recommend scientific articles, is a hybrid approach that combines Collaborative Filtering (CF) based on latent factor models [Salakhutdinov and Mnih, 2008a] and content analysis based on probabilistic topic modeling [Blei et al., 2003; Chang et al., 2009]. Latent factor models leverage user interactions and topic models rely on articles' content or textual descriptions of items. Introducing a content analysis component with CF improves the recommendation performance and allows to generalize to previously unseen items.

Topic models. The goal in topic modeling approaches [Blei and Lafferty, 2009] is to associate a document seen as an unordered list of words with a vector of topics, i.e., of word distributions. Documents are thus represented in a low-dimensional space and models have the characteristic of being interpretable [Chang et al., 2009]. CTR integrates Latent Dirichlet Allocation (LDA) [Blei et al., 2003] which is one of the most influential topic modeling approaches.

Latent Dirichlet Allocation. LDA [Blei et al., 2003] is a generative model describing text documents and corpora where the key notion involved in the description of a document is the notion of *topic*. A topic corresponds to a word distribution (for instance in a music-oriented corpus, the word "concert" would have a higher probability to be drawn than in a sport-oriented corpus) and a document is described as a mixture of topics. Topics are learned in an unsupervised fashion and, thus, do not necessarily correspond to human-understandable concepts [Chang et al., 2009]. The popularity of LDA is due to its simplicity, modularity, and interpretability. Assuming there are K topics, the generative process for each document in the corpus is presented in Figure 4.1 and is described as follows:

1. Choose topic proportions $\boldsymbol{\theta} \sim Dirichlet(\boldsymbol{\alpha})$
2. For each word w_n in document:
   (a) Choose a topic $z_n \sim Multinomial(\boldsymbol{\theta})$
   (b) Choose a word w_n from the multinomial $p(w_n|z_n, \boldsymbol{\beta})$

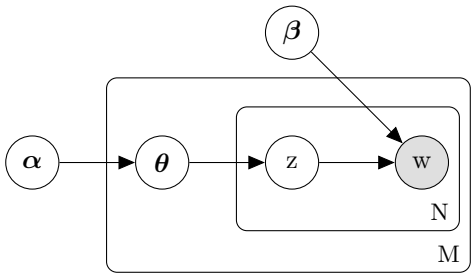


FIGURE 4.1: Generative model for Latent Dirichlet Allocation (LDA)

In terms of document analysis, the parameters of LDA can be understood in the following way:

- The vector $\boldsymbol{\alpha}$ represents the global topic trend. For instance, a parameter $\boldsymbol{\alpha} = (1, \ldots, 1)$ corresponds to a uniform choice over all topics, on average. The higher the component α_x is, the more frequent topic x will be in the whole corpus.
- The matrix $\boldsymbol{\beta}$ stores the probability of words inside topics. If a word w is set to belong to topic z, then it will be chosen with probability $\beta_{z,w}$.
- The vector $\boldsymbol{\theta}$ corresponds to the topic distribution inside one document.

Compared to clustering approaches where each observation is assigned to only one cluster, LDA enables documents to exhibit several topics. A training of LDA model is possible based on maximum likelihood principle [Blei et al., 2003]. In practice, it is suggested to use either Gibbs sampling or variational inference for this task.

Collaborative Topic Modeling. CTR assumes that documents describing the items are generated by a topic model and represents users with topic interests. It adds a latent variable that offsets the topic proportions when modeling user latent factors. This offset is derived from the rating data and represents the fact that two similar items, i.e., having similar topic proportions, can be interesting to different types of users. As more users interact with the item, the offset value becomes clearer. The latent factor model used in CTR is Probabilistic Matrix Factorization (PMF) [Salakhutdinov and Mnih, 2008a] (Section 2.5.2.5) and the topic model is LDA [Blei et al., 2003].

We assume that each hotel i is described by a document d_i containing n_{d_i} words and having a multinomial distribution $\boldsymbol{\theta}_i$ over K topics. Users and items are modeled in the low-dimensional space of dimension K. Each user u is represented by the latent vector $\mathbf{p}_u \in \mathbb{R}^K$ and each hotel i by the latent vector $\mathbf{q}_i \in \mathbb{R}^K$. We define $\mathbf{P} = (\mathbf{p}_u)_1^n$ and $\mathbf{Q} = (\mathbf{q}_i)_1^m$, and we denote by λ_* the regularization parameters. The generative process of CTR is given as follows:

1. For each user u, choose user latent vector $\mathbf{p}_u \sim \mathcal{N}(0, \lambda_P^{-1}\mathbf{I}_K)$
2. For each hotel $\boldsymbol{i}$,
   (a) Choose topic proportions $\boldsymbol{\theta}_i \sim Dirichlet(\boldsymbol{\alpha})$
   (b) Choose hotel latent offset $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \lambda_Q^{-1}\mathbf{I}_K)$ and set the hotel latent vector as $\mathbf{q}_i = \boldsymbol{\epsilon}_i + \boldsymbol{\theta}_i$
   (c) For each word w_{in} in d_i,
      i. Choose topic assignment $z_{in} \sim Multinomial(\boldsymbol{\theta}_i)$
      ii. Choose word $w_{in} \sim Multinomial(\boldsymbol{\beta}_{z_{in}})$
3. For each user-hotel pair (u, i), choose the rating $r_{ui} \sim \mathcal{N}(\mathbf{p}_u\mathbf{q}_i^\top, c_{ui}^{-1})$

We note that $\mathbf{I}_K$ is a K-dimensional identity matrix and c_{ui} is the precision parameter for rating r_{ui} defined in Section 2.5.2.4 and indicating the confidence we have in the observation r_{ui}.

Learning the parameters. Computing the full posterior of the parameters is intractable and the authors in [Wang and Blei, 2011] develop an algorithm in the style of the expectation-maximization algorithm to learn the maximum a posteriori estimates. The maximization of the posterior is equivalent to maximizing the log-likelihood of $\mathbf{P}$, $\mathbf{Q}$, $(\boldsymbol{\theta}_i)_1^m$, and $\mathbf{R}$, given λ_P, λ_Q, and $\boldsymbol{\beta}$,

$$\mathcal{L} = -\frac{\lambda_P}{2}\sum_u \|\mathbf{p}_u\|_2^2 - \frac{\lambda_Q}{2}\sum_i \|\mathbf{q}_i - \boldsymbol{\theta}_i\|_2^2 + \sum_i \sum_n \log\left(\sum_m \theta_{im}\beta_{m,w_{in}}\right) - \sum_{u,i}\frac{c_{ui}}{2}(r_{ui} - \mathbf{p}_u\mathbf{q}_i^\top)^2 \tag{4.1}$$

The function is optimized by coordinate ascent, iteratively optimizing the CF variables $\{\mathbf{p}_u, \mathbf{q}_i\}$ and the topic proportions $\boldsymbol{\theta}_i$. The authors in [Wang and Blei, 2011] state that fixing $\boldsymbol{\theta}_i$ as the estimate from vanilla LDA gives a comparable performance and reduces the complexity of computations. The final relevance scores, $\hat{r}_{ui}$, are computed as follows: $\hat{r}_{ui} = \mathbf{p}_u\mathbf{q}_i^\top$.

4.3.2 Handling Positive and Negative Reviews

Our approach integrating textual reviews as hotel descriptions into the recommendation model, `CTRk+`, is based on CTR. The PMF is replaced by a ranking model and topics are extracted separately from positive and negative reviews and finally combined in the model.

Pairwise approach. Motivated by Bayesian Personalized Ranking (BPR) [Rendle et al., 2009] (Section 2.5.2.4) and its superiority in implicit feedback settings, we introduce a pairwise method for the CF part of CTR. Model parameters are learned by considering the pairwise ranking between hotels. The user u is assumed to prefer hotel i over hotel j if $r_{ui} = 1$ and $r_{uj} = 0$ and the pairwise preference probability is given by:

$$p(i >_u j \mid \Theta) = \sigma(\mathbf{p}_u\mathbf{q}_i^\top - \mathbf{p}_u\mathbf{q}_j^\top) \tag{4.2}$$

where σ is the logistic sigmoid and Θ represents the model parameters. The set of training data, D_{bpr}, is defined as follows: $D_{bpr} := \{(u,i,j) \mid r_{ui} = 1 \wedge r_{uj} = 0\}$. This leads to replacing, in Equation 4.1, $-\sum_{u,i}\frac{c_{ui}}{2}(r_{ui} - \mathbf{p}_u\mathbf{q}_i^\top)^2$ by $\sum_{(u,i,j)\in D_{bpr}} \ln\sigma(\mathbf{p}_u\mathbf{q}_i^\top - \mathbf{p}_u\mathbf{q}_j^\top)$, since the elements of D_{bpr} are drawn from the probability $\sigma(\mathbf{p}_u\mathbf{q}_i^\top - \mathbf{p}_u\mathbf{q}_j^\top)$ (item 3 in the generative process of CTR). Learning is performed following the same algorithm as CTR and the update equations are modified accordingly. Experiments on our datasets showed that fixing $\boldsymbol{\theta}_i$ as the estimate from LDA leads to a comparable performance, as in the original work of CTR, which was adopted in our setting.

Handling review polarity. We rely on reviews extracted from TripAdvisor [2] in order to provide a textual description for hotels. TripAdvisor reviews are graded from 1 to 5, representing the level of user satisfaction. They can be divided into a set of positive reviews

[2] http://www.tripadvisor.com

mentioning available and appreciated characteristics of the hotel and a set of negative reviews pointing out the missing or unsatisfying features. A large majority of users check the reviews online before booking a hotel and are influenced by what is shared. While the objective description of hotels is available, we consider that the reviews are a richer source because they derive from the real user experience.

In the latent space in which users and hotels are represented, the elements of $\mathbf{p}_u$ and $\mathbf{q}_i$ measure respectively how much the user is attracted to a latent factor and to what extent the hotel possesses it, positively or negatively. A first approach to applying CTR to our problem is to concatenate all available reviews for a hotel i in one document d_i and use it to extract the topic proportions θ_i. The values of $\mathbf{q}_i$ that we get indicate the probability of a topic emerging in d_i. However, we do not distinguish between the cases where this topic denotes a positive feature of i or a negative one.

We propose to extract topics separately from positive and negative reviews and combine them to model items. Therefore, we model the hotel latent vector as follows:

$$\mathbf{q}_i = \boldsymbol{\epsilon}_i + \boldsymbol{\theta}_{i^+} - \boldsymbol{\theta}_{i^-} \tag{4.3}$$

where $\boldsymbol{\theta}_{i^+}$ denotes the topic distribution extracted from positive reviews and $\boldsymbol{\theta}_{i^-}$ the one extracted from negative reviews.

4.4 Influence of the Modal Context

The modal context covers the users' state of mind and mood. We consider here two aspects of the influence of modal context on users. First, the user's interest may change over time and the preference shift should be considered during the learning process by giving more weight to recent interactions. Then, choosing accommodations is strongly related to the trip's intent which is not provided explicitly but can be inferred from features related to the stay. These features include information related to the *lead time*, the *staying days of the week*, and the *company of other people*. The *lead time* is the time separating the reservation date and the check-in date and is usually higher for leisure trips than for business trips, leisure trips being usually planned in advance. Analyzing the *staying days of the week* is insightful since a stay only involving weekdays and organized in a non-holiday period is most likely a business one. Other information concerning the *company of other people*, i.e., the number of adults and the number of children, is also meaningful.

Since the trip's intent is not provided at the moment of recommendation, we build a classical recommendation model without explicitly modeling the context variable like in most CARS (Section 2.7). Information about the trip's intent is leveraged during the learning process where we propose to modify the probability of sampling observations depending on the associated context. The approach we propose, which we refer to as `BPRx3`, is based on `BPR` [Rendle et al., 2009] (Section 2.5.2.4) where we rely on a biased sampling to learn the model.

BPR. As mentioned before, BPR assumes that user u prefers hotel i over hotel j if the pair (u, i) is observed and the pair (u, j) is not. The learning data is represented by the set

$$D_{bpr} := \{(u, i, j) \mid r_{ui} = 1 \wedge r_{uj} = 0\},$$

the training triples being uniformly sampled from the data due to the very large number of pairs (u, i, j).

BPR_{rec} and the *new interest* aspect. We assume that hotels chosen recently are preferred over the ones corresponding to older choices. BPR++ [Lerche and Jannach, 2014] is an extension of BPR and addresses, among other points, the recency and temporal aspect of events. The authors proposed a biased sampling that would sample alternatively from D_{bpr} or from another set that we denote by D^{rec}_{bpr} and defined as follows:

$$D^{rec}_{bpr} := \{(u, i, j) \mid r_{ui} = 1 \wedge r_{uj} = 1 \wedge rec(u, i, j) = 1\},$$

where the function $rec(u, i, j)$ returns 1 if the booking done by u in i is more recent than the booking done in j, and 0 otherwise.

BPR_{int} and the *new situation* aspect. We assume that a hotel observed in a certain context revealing the intent of the user's trip is preferred over the ones usually chosen in this same context. We denote by $\mathcal{F}_{ui}$ the set of features related to the intent of the trip and associated to r_{ui}, when the pair (u, i) is observed. $\mathcal{F}_{ui}$ includes the lead time, the number of adults and children accompanying the user, the week in which the check-in occurs, and the fact that the stay spans over a weekend or not. The learning data D^{int}_{bpr} is defined similarly to D_{bpr} but the probability of sampling j is as follows:

$$p(j \mid \mathcal{F}_{ui}) \propto |\{r_{aj} = 1 \mid \mathcal{F}_{aj} = \mathcal{F}_{ui}, \forall a \in \mathcal{U}\}|.$$

Some hotels are more targeted when traveling in a specific context and have therefore more chances of being sampled as a negative item for that particular context, if they are not observed.

Combining the three assumptions in `BPRx3`. We propose to use all three sets D_{bpr}, D^{rec}_{bpr}, and D^{int}_{bpr} to learn the model, and we introduce two parameters γ_{rec} and γ_{int} indicating the probability of sampling from each set. The learning procedure is detailed in Algorithm 4.

Algorithm 4 `BPRx3`: `BPR` based on a biased sampling

Input: D_{bpr}, D^{rec}_{bpr}, D^{int}_{bpr}

initialize Θ ▷ parameters of the model
repeat
  p = random(0, 1)
  if $p \leq \gamma_{rec}$ **then** draw (u, i, j) from D^{rec}_{bpr}
  else
    if $p \leq \gamma_{int}$ **then** draw (u, i, j) from D^{int}_{bpr}
    else draw (u, i, j) from D_{bpr}
    end if
  end if
  update Θ ▷ similar to [Rendle et al., 2009]
until convergence

4.5 Overview of the System

This section describes the architecture of the RS designed in industry. It is based on the previously proposed methods and operates in two phases.

Phase 1. We cluster users in several buckets with respect to their country of residence and to the visited destinations in different periods, taking into account the geographical and temporal influences (Section 4.2). Grouping users affected similarly by both of these dimensions before building the recommendation model helps to uncover users' preferences. The models are then built locally for each cluster of users.

Phase 2. In this phase, we build local recommendation models taking into account the several types of context. The performance of the recommendation methods depends on the sparsity level of data for each user. While some approaches are adapted for new users, others are more suited for frequent users. The user category also defines which type of context is more relevant when selecting accommodations.

After testing multiple recommendation approaches (refer to Section 4.6), we select the most accurate one for each segment of users. As the user makes more and more bookings, we collect additional information about his preferences and feed it to a more appropriate model thus improving the overall quality of recommendation. To this end, with the help of domain experts, we define a segmentation of users based on the number of bookings made. The user segments are defined as follows: *inactive users*, *rare users*, *occasional users*, and *frequent users*. We use the number of bookings as a threshold to separate users into segments. The thresholds are derived from the data and we rely on extensive experiments to determine them. We note that the definition of these segments is proper to our recommendation problem and serves the purpose of building an accurate system. Section 4.6.3 details how we form the segments and which method we apply for each one of them.

4.6 Experimental Results

Extensive experiments were conducted on several datasets to demonstrate the models proposed in the previous sections and to justify the architecture of the developed system. This section attempts to answer the following questions:

- **Q1.** How does the performance of local models built with respect to geography and temporality compare to the performance obtained with one global model? (Section 4.2)
- **Q2.** Does introducing positive and negative reviews as a description of hotels improve the quality of recommendation? (Section 4.3)
- **Q3.** Does introducing the recency of bookings and features related to the trip's intent improve the quality of recommendation? (Section 4.4)
- **Q4.** How do classical recommendation approaches compare when applied on a real-world dataset extracted from the hotel industry?

TABLE 4.4: Statistics of the booking datasets used in this chapter

Dataset	#users	#hotels	#bookings	#reviews
AH	7,802,637	4,574	34,709,006	-
AH-MAXI	338,259	4,111	1,120,508	-
AH-MINI	58,959	3,317	228,155	-
AH-TRIP	56,909	2,718	210,974	201,922

- **Q5.** How does the performance of the proposed approaches vary according to the user segment? (Section 4.5)

Metrics. Recommendation models are evaluated on a set of held-out bookings. We consider that we recommend N hotels to each user, and note which of these hotels were actually visited. We use recall@N and NDCG@N for measuring the performance (Section 2.2.3). Since large values of N are not interesting for top-N recommendation, we set N to 5 and 10 for the two metrics.

Parameters. We perform a grid search over the parameter space of each evaluated method in order to find the parameters that give the best performance that is reported.

4.6.1 Contribution of the Physical Context

In order to prove the effectiveness of the `localRS` component, we compare the performance of the recommendation models under two different settings. In the first one, `globalRS`, we build one model for all users. In the second one, $\texttt{localRS}_\texttt{k}$, we build one model per cluster of users, for a total of k clusters (Section 4.2).

Datasets. AH is the original dataset described in Section 3.2. We associate to each cluster $i \in [1, k]$ a dataset AH-I derived from AH.

Experimental setup. We split the dataset AH into a training and a test set. We sort the bookings of each user in a chronological order and select the first 80% to constitute the training set and the rest to constitute the test set. For users that appear only once in the booking data, we randomly select 20% of them and add their booking to the test set in order to evaluate the performance of the system on inactive users. We use the data from the training set of AH in order to generate k clusters of users.

For the `globalRS` setting, we use the training and test sets of AH in order to learn the recommendation model and test it. For the $\texttt{localRS}_\texttt{k}$ setting, we build one model for each dataset AH-I using its training set and we generate recommendations for users appearing in the corresponding test set. The training and test sets of each dataset AH-I are derived respectively from the training and test sets of AH by filtering the bookings made by the users contained in the cluster i.

Compared methods. In order to compare `globalRS` and $\texttt{localRS}_\texttt{k}$ and highlight the contribution of the clustering component, we show the results for two recommendation methods:

TABLE 4.5: Answering Q1. Recall@N and NDCG@N of the `Knni` and `BPR` methods under the `globalRS` and $\texttt{localRS}_\texttt{k}$ settings for the dataset AH

Method	Metric	globalRS	$\texttt{localRS}_5$	$\texttt{localRS}_{10}$	$\texttt{localRS}_{15}$
Knni	Recall@5	0.0846	0.0861	0.08590	**0.0863**
	Recall@10	0.1284	**0.1306**	0.1301	**0.1306**
	NDCG@5	0.0667	**0.069**	0.0689	**0.069**
	NDCG@10	0.0823	0.0848	0.0847	**0.085**
BPR	Recall@5	0.3212	0.3253	**0.3261**	0.3258
	Recall@10	0.3704	0.3740	**0.3741**	**0.3741**
	NDCG@5	0.2873	0.302	**0.303**	0.3028
	NDCG@10	0.3048	0.3194	**0.3201**	**0.3201**

- `Knni` is the item-based neighborhood method 2.5.1.2. We use the Jaccard similarity to measure items' similarity and set the number of neighbors to 2000.
- `BPR` relies on pairwise preferences when learning the latent model [Rendle et al., 2009] (Section 2.5.2.4). We set the number of latent factors to 100, and the regularization parameters $\lambda_P = \lambda_Q = 0.0025$.

Answering Q1. Table 4.5 shows the recall@N and NDCG@N for $N = \{5, 10\}$ of the `Knni` and `BPR` methods under the `globalRS` and $\texttt{localRS}_\texttt{k}$ settings. The quality of recommendation is improved under the $\texttt{localRS}_\texttt{k}$ setting — for all the represented values of k — underlining the importance of clustering users before generating recommendations. The optimal value of k varies with each tested model. After leading extensive experiments, we set $k = 10$, maximizing the overall gain we get on a representative set of methods when compared to the `globalRS` setting. Further experiments prove that the performance improves when in the $\texttt{localRS}_\texttt{k}$ setting regardless of the recommendation method.

4.6.2 Contribution of the Social and Modal Contexts

We evaluate here `CTRk+` and `BPRx3` proposed in Sections 4.3 and 4.4 respectively, highlighting the benefits of integrating textual reviews and features related to the trips' intent into the recommendation model.

Datasets. In the developed system, recommendation models are built for each cluster separately ($\texttt{localRS}_\texttt{k}$ setting). We show the results for two clusters of different sizes which datasets are denoted by AH-MINI and AH-MAXI. AH-TRIP is derived from AH-MINI and is used to validate the model introducing textual reviews. We extract 201k reviews from TripAdvisor, written in the French language, and we keep in AH-TRIP the hotels with available reviews and their related bookings. Reviews have been preprocessed by mainly removing stop words, removing words occurring once, and stemming remaining words. The details concerning the composition of the datasets are in Table 4.4.

TABLE 4.6: Answering Q2. Recall@N and NDCG@N of CTRk+ and other variants for the dataset AH-TRIP

Metric	WRMF	CTR	CTRk	CTRk+
Recall@5	0.3642	0.3721	0.3687	**0.3732**
Recall@10	0.4166	0.4371	0.4467	**0.4542**
NDCG@5	0.3358	0.343	0.3411	**0.3447**
NDCG@10	0.3584	0.3658	0.3701	**0.3769**

Methods compared. The models included in our comparison, as well as the corresponding parameters giving the best performance on held-out recommendations, are listed in the following:

- WRMF is a Matrix Factorization (MF) technique handling implicit feedback datasets [Hu et al., 2008] (Section 2.5.2.4). We set the number of latent factors to 100, $\lambda_P = \lambda_Q = 0.001$, and $c_{ui} = 1.0$ for positive observations and $c_{ui} = 0.01$ for negative observations.
- CTR [Wang and Blei, 2011] is a model using topic modeling and CF simultaneously (Section 4.3). Each hotel is described by one document combining all of its reviews. We set the number of latent factors to 100, $\lambda_P = 0.001$, $\lambda_Q = 10$, and $c_{ui} = 1.0$ for positive observations and $c_{ui} = 0.01$ for negative observations.
- CTRk is based on CTR and uses a pairwise method for performing the CF part (Section 4.3). We set the number of latent factors to 100, $\lambda_P = 0.0025$, $\lambda_Q = 0.025$, and $c_{ui} = 1.0$ for positive observations and $c_{ui} = 0.01$ for negative observations.
- CTRk+ is derived from CTRk where we distinguish between topics extracted from positive and negative reviews (Section 4.3). We set the number of latent factors to 100, $\lambda_P = 0.0025$, $\lambda_Q = 0.025$, and $c_{ui} = 1.0$ for positive observations and $c_{ui} = 0.01$ for negative observations.
- BPR [Rendle et al., 2009] is introduced in Section 2.5.2.4. We set the number of latent factors to 100, and $\lambda_P = \lambda_Q = 0.0025$.
- $\text{BPR}_{\texttt{rec}}$ [Lerche and Jannach, 2014] expands BPR using the recency of bookings to learn preferences (Section 4.4). We set the number of latent factors to 100, $\gamma_{rec} = 0.1$, and $\lambda_P = \lambda_Q = 0.0025$.
- $\text{BPR}_{\texttt{int}}$ uses features related to the trip's intent when sampling triplets to learn the model (Section 4.4). We set the number of latent factors to 100, $\gamma_{int} = 0.5$, and $\lambda_P = \lambda_Q = 0.0025$.
- BPRx3 relies on a biased sampling guided by the recency of bookings and by features related to the intent of the user's trip (Section 4.4). We set the number of latent factors to 100, $\gamma_{rec} = 0.1$, $\gamma_{int} = 0.6$, and $\lambda_P = \lambda_Q = 0.0025$.

Answering Q2. We measure recall@N and NDCG@N for $N = \{5, 10\}$ for the following methods: WRMF, CTR, CTRk, and CTRk+ for the dataset AH-TRIP and we report the results

TABLE 4.7: Answering Q3. Recall@N and NDCG@N of BPRx3 and other variants for the dataset AH-MINI

Metric	BPR	BPR_{rec}	BPR_{int}	BPRx3
Recall@5	0.3396	0.3703	0.3497	**0.3734**
Recall@10	0.4328	0.4457	0.4376	**0.4477**
NDCG@5	0.3146	0.3548	0.3245	**0.3577**
NDCG@10	0.351	0.3847	0.3588	**0.3871**

in Table 4.6. We make the following observations: 1. CTR performs better than WRMF, which highlights the contribution of the textual reviews to the CF model; 2. As we increase the number of recommended hotels N, the ranking model CTRk becomes more adapted than CTR; 3. CTRk+ outperforms the other models and proves the importance of handling positive and negative reviews separately when extracting the topics.

Answering Q3. Table 4.7 shows the recall@N and NDCG@N for $N = \{5, 10\}$ for the variants of BPR that we propose and test for the AH-MINI dataset. BPR_{rec} and BPR_{int} use a biased sampling taking into account respectively the recency of bookings and features related to the intent of the trip, and perform better than the classical BPR. The hypotheses we make about the preferences' learning improve the quality of recommendation. BPRx3 relies on the three different assumptions and outperforms the other models. The best performance is obtained for $\gamma_{rec} = 0.1$ and $\gamma_{int} = 0.6$, proving the interest of using all of the three learning sets: D_{bpr}, D^{rec}_{bpr}, and D^{int}_{bpr}.

4.6.3 User Segmentation and Performance

In favor of addressing the questions Q.4 and Q.5, we report the results separately for each category of users, a category being defined by the number of bookings previously made by the user. These categories will later constitute the segments of users we rely on in our work.

Methods. The methods evaluated in this subsection include WRMF, BPR, CTRk+, and BPRx3, in addition to the following methods:

- MostPop recommends the most popular hotels with respect to the user's profile. The user profile includes a set of features related to the gender, the age category, and the country of residence. Note that this information is often missing in our setting, except for the country of residence. MostPop recommends to a target user the most popular hotels chosen by those having the same or similar profiles.
- CB is a content-based method where hotels and users are represented in the space of hotels' features using vector space models and TF-IDF weighting [Pazzani and Billsus, 2007] (Section 2.4.1). Hotel features cover the location, the brand, the segment category, and offered services such as Wi-Fi connection, parking, meeting facilities, and children playground.

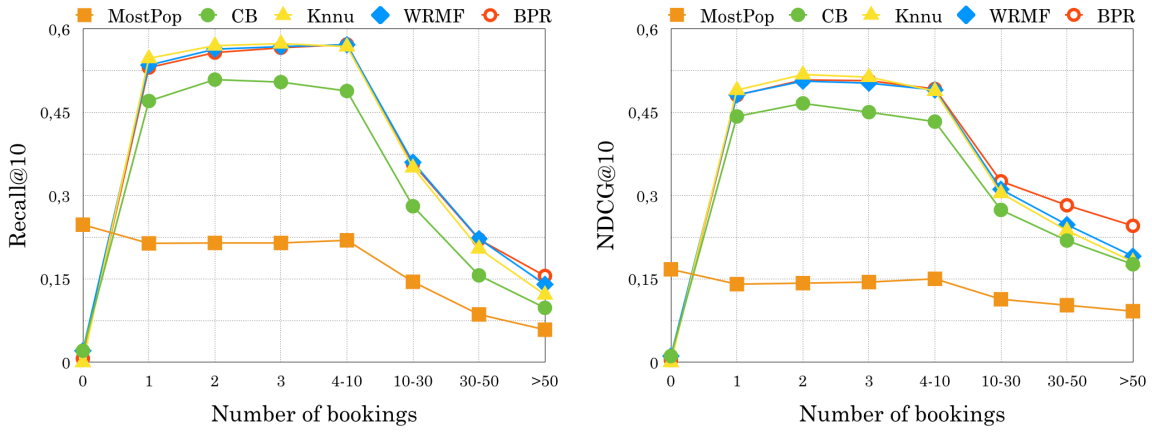


FIGURE 4.2: Answering Q4. Recall@10 and NDCG@10 of traditional recommendation approaches for the dataset AH-MAXI

TABLE 4.8: Answering Q5. Recall@10 of the proposed recommendation methods for the dataset AH-TRIP represented per category of users

#bookings	Recall@10			
	MostPop	Knnu	CTRk+	BPRx3
0	**0.3881**	0.0000	0.0174	0.0159
1	0.3276	**0.6383**	0.6222	0.6193
2	0.3079	**0.6236**	0.6179	0.6138
3	0.3054	**0.6221**	0.6142	0.6171
4-10	0.2745	0.5671	**0.5839**	0.5796
10-30	0.1801	0.3501	**0.3855**	0.3791
30-50	0.1209	0.2165	0.2483	**0.2515**
>50	0.0831	0.1527	0.1802	**0.1831**

- `Knnu` is the user-based neighborhood method, described in Section 2.5.1.1. We rely on the Jaccard similarity metric to measure similarities between users. We set the number of neighbors to 2000.

Answering Q4. Figure 4.2 shows the performance of common approaches to recommendation for the dataset AH-MAXI. By definition, and for a fixed value of N, the metrics we use decrease when the number of bookings made by users increases. The performance of the methods depends on the category of users. `MostPop` is the only method able to recommend hotels to *inactive users*, i.e., users who have not done any booking. `Knnu` performs best for users who have done few bookings while `BPR` outperforms the other methods when the number of bookings increases significantly. Since we have at our disposal a large number of features describing the hotels, we may think that `CB` is a robust method for recommendation. Experiments show that users attribute great importance only for a small set of features, including the location of the hotel. Their decision is rather driven by dynamic features. The results show the differences in performances with respect to the user category and highlight several types of behaviors where each one is better addressed by a specific model.

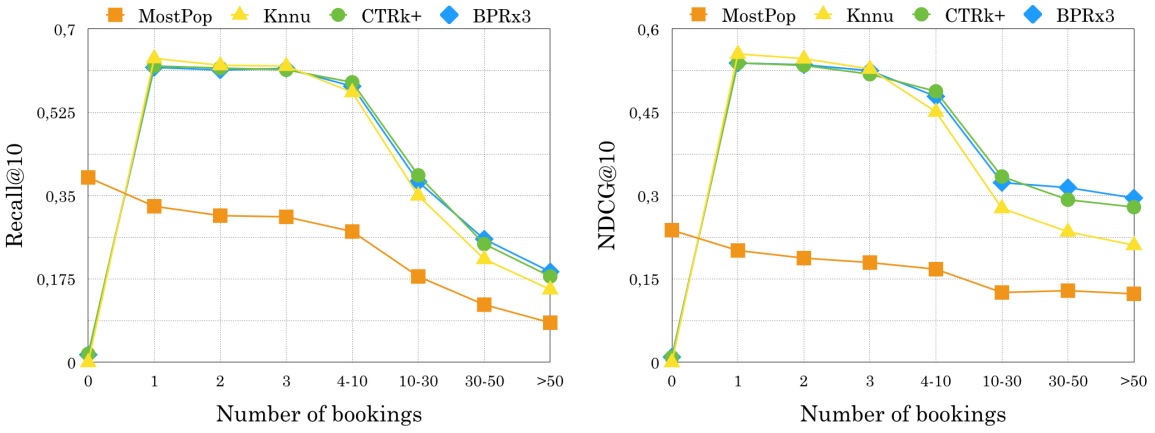


FIGURE 4.3: Answering Q5. Recall@10 and NDCG@10 of context-aware approaches per category of user for the dataset AH-TRIP

TABLE 4.9: Answering Q5. NDCG@10 of the proposed recommendation methods for the dataset AH-TRIP represented per category of users

#bookings	NDCG@10			
	MostPop	Knnu	CTRk+	BPRx3
0	**0.2378**	0.0000	0.0088	0.0095
1	0.2012	**0.555**	0.5388	0.5386
2	0.1875	**0.5466**	0.5344	0.5356
3	0.1795	**0.5281**	0.5186	0.5249
4-10	0.1675	0.4509	**0.4877**	0.4786
10-30	0.1256	0.2772	**0.3343**	0.3234
30-50	0.1289	0.2349	0.2926	**0.3144**
>50	0.1232	0.2108	0.2793	**0.2957**

Answering Q5. We compare here the methods that perform best on different segments of users. Figure 4.3, Table 4.8, and Table 4.9 show the metrics for the dataset AH-TRIP. Based on these results, we define below the user segmentation we rely on to improve the overall quality of recommendation:

- **Segment 1.** ***Inactive users.*** These users have not done any booking, and we can only use `MostPop` to generate recommendations.
- **Segment 2.** ***Rare users.*** When users start to make bookings, `Knnu` outperforms the other methods. At this stage, we start gathering information about users but not enough to apply latent factor models. This segment includes users having done between 1 and 3 bookings.
- **Segment 3.** ***Occasional users.*** We rely on `CTRk+` to recommend hotels for users that have previously done between 4 and 30 bookings. The contribution of the textual reviews is perceived on this segment where the introduction of auxiliary information avoids overfitting the relatively small number of available bookings.

- **Segment 4.** ***Frequent users.*** The benefits of using the recency of bookings and features related to the trips' intent (`BPRx3`) appear for users having done more than 30 bookings, where the behavior becomes complex to decode. It is then essential to distinguish between the context of each booking for preference learning. The results show that `BPRx3` outperforms the other methods for this segment of users.

4.7 Conclusion

The increased amount of information related to tourism has made trip planning a challenging task and personalized recommendations have thus become essential to guide users to the best choices. One of the main challenges in the hotel industry is related to the cold-start problem. In particular, users tend to return regularly to the cold status for multiple reasons, making it harder for the RS to provide the best recommendations. To address this challenge, we developed a RS that incorporates the contextual factors that influence the users' decisions. The system operates in two phases. First, users are clustered based on their physical context, grouping those having similar patterns with respect to the visited destinations and the time of the visits. Second, local models are built in each cluster and a different method is used to target each segment of users. We proposed context-aware models that consider the social context, i.e., textual reviews posted on a social platform, and the modal context, i.e., the trips' intents.

Further extensions of this work should consider the influence of contextual factors on properties other than the accuracy of the RS. In particular, serendipity, evaluating how surprising the relevant recommendations are, is a desirable characteristic (Section 2.2.2). The items suggested should be interesting and useful but not yet discovered by the user. Serendipitous recommendations lead to a more pleasant and engaging experience. Additional efforts should be made to balance between the accuracy and the serendipity of the RS.

Partially observable contextual factors considered in this chapter are all directly related to users, hotels, or user interactions with hotels. This may not always be the case, especially in situations where the user's context is observed in a domain other than the hotel domain. Mappings between auxiliary domains and the target domain have thus to be established. This idea is developed in Chapters 5 and 6 that exploit respectively event and mobility information to boost hotel recommendation.

Chapter 5

Leveraging Implicit Context

This chapter presents our proposed approach to integrating information related to planned events into hotel RS [Al-Ghossein et al., 2018a], given the importance of context for hotel recommendation as discussed in Chapter 3. Events are well-known to influence the travelers' behaviors by motivating them to organize trips. In our setting, as in most real-world applications, events and hotels belong to two different domains and feedback about events is not provided by users visiting hotels. Events are then considered to be part of the context variable affecting users while related information is *implicitly* collected by the system. There are no direct links between events on one side and users and hotels on the other. In addition, due to the fact that events are unique and short-lived, users face a new context for each organized event. The absence of direct links and the ephemeral nature of events qualify the event dimension of context as *partially observable*. We propose a framework addressing these challenges and evaluate it on real-world datasets.

5.1 Introduction

Planned events constitute a major motivator of tourism and have been playing an essential role in drawing attention to specific regions [Getz, 2008]. They are spatiotemporal and ephemeral phenomenon characterized by the people involved, the setting, and the program, among others, making each one of them a unique occasion. This character of unicity often motivates people to attend events as they will not reoccur in the exact same way in the future. Events can be classified into several categories [Getz, 2008]. Some are organized for *fun* and *entertainment* such as festivals and concerts while others are planned for *business* and *educational* purposes such as meetings, conventions, and conferences. Events can also promote *competition* like sports games. They are usually held in venues that are known to host specific types of events.

Guided by their preferences, people are usually interested in attending an event based on its intrinsic features like its type, the performers involved, and the venue where it is organized, but also based on external social factors, e.g., company of other people. In all cases, the experience of attending an event in one's hometown is significantly different from

the one of attending an event far from it. Travel becomes then a necessary condition for attendance and incurs additional costs and the need to organize a whole different experience. The process of organizing a trip requires, among other things, selecting the best option for accommodation: When planning to attend an event abroad, people usually search for a hotel close to the event's venue. Therefore, event planning affects the hotel selection process.

As discussed in Chapter 3, hotel RS need to consider the contextual factors influencing the travelers. Taking into account the schedule of events takes one step forward into modeling the real decision-making process of travelers. However, it is not a trivial task. While travelers' bookings may be associated with events organized in the neighborhood of the visited hotels, this association is not available in an explicit form. Observations about users' bookings on one side and about the schedule of events on the other are totally decoupled: Travelers do not provide the reason of their visit or whether they took the trip to attend a specific event. Moreover, only important and impactful events that are able to attract travelers from abroad are relevant to this task. Criteria and methods to filter these events need to be clearly defined. On some social networks, e.g., Facebook [1] and Eventful [2], users have the possibility to express their interest in attending an event. However, it is not easy to collect this information and it is thus not possible to learn user preferences for important events, even for an independent set of users, i.e., users of the social network. The ephemeral and unique character of events makes it also more challenging to generalize user preferences based on past events.

In this chapter, we explore the benefits of introducing open data related to events into hotel RS. We show how we can infer user preferences to events based on their history of hotel bookings and on information related to events, and how we can use these preferences and the schedule of future events to improve the quality of hotel recommendations. By exploiting information about hotels' environments and by recommending packages of hotels and events, we aim to facilitate tourism planning and enhance the user experience. We formulate our problem as two subproblems: the *hotel-centric* and *event-centric* problems, occurring in two different use cases, and we develop a novel framework addressing them. In our setting, direct feedback about events is not available since hotel organizations, who have access to users' bookings, do not have any information about the events these users attended. Our experiments on a real-world dataset underline the potential of using open data related to events in order to boost the performance of hotel RS.

The remainder of the chapter is organized as follows. In Section 5.2, we give a short overview of the related work on event recommendation and in Section 5.3, we formally introduce the problem we address. Section 5.4 details the process of collecting and cleaning data related to events. Section 5.5 presents our framework that is developed to learn events' preferences and generate hotel recommendations. In Section 5.6, we perform an evaluation of the framework and in Section 5.7, we discuss the results. Finally, Section 5.8 concludes the chapter.

[1] http://www.facebook.com
[2] http://www.eventful.com

5.2 Event Recommendation

Event-Based Social Networks (EBSN) such as Eventful and Meetup [3] allow users to create, share, and promote upcoming events. The vast amount of events available on EBSN has raised the problem of event recommendation. Recommending events in EBSN is subject to the new item cold-start problem (Section 2.1.3) that appears naturally in this setting: All the events to be recommended are new, ephemeral, and expected to occur in the future, and information about the historical attendance is not available. To overcome this limitation, event recommenders rely on additional relevant information related to users and events.

Hybrid recommendation. Hybrid approaches for event recommendation consider content information describing events. Preferences for upcoming events are then inferred based on preferences for past events with similar contents. [Minkov et al., 2010] leverage topics inferred from announcements and other textual resources related to events. Event descriptions are mapped to a low-dimensional space and the user parameters are then associated with these coordinates rather than the original descriptions of events. In addition, content dimensions shared collaboratively among users are distinguished from those that are unique to individual users. The proposed hybrid approach is proven to be more effective than a pure content-based approach. Another hybrid approach for recommending events from the music domain is presented in [Khrouf and Troncy, 2013]. Category information about the artists is extracted from DBpedia [Lehmann et al., 2015] and used to enrich the description of events. The cohesiveness of the user's content-based profile is also modeled to highlight his core preferences. The content-based model is combined with a model based on social interactions between users in order to integrate the collaborative dimension. Experiments on a dataset collected from three event web directories, Eventful, Last.FM [4], and Upcoming, demonstrate the usefulness of each component.

Social dimension. Social interactions on EBSN have also been used in other works to enhance the preference learning process. Users belonging to the same group or community are expected to be attracted by related events. [Liu et al., 2012] exploit online and offline social interactions for event recommendation. Online social interactions are directly observed on the EBSN while offline social interactions cover the interactions that are expected to happen between users when they actually get together physically to attend events. Information flow models are then developed to infer the user interest in an event based on the interests of other users. Offline and online social interactions were also used in [Qiao et al., 2014] where the authors propose a Bayesian Matrix Factorization (MF) approach based on social regularization factors.

Contextual recommendation. Contextual signals influence users' decisions and are thus leveraged for event recommendation. In addition to content and social features, [Macedo et al., 2015] rely on location signals, i.e., users' home distance to the events, and time signals, i.e., users' time preferences. Mobility patterns of users are modeled to distinguish between users who prefer to attend events in their neighborhood and those who are likely to attend events far away. User preferences for temporal factors, e.g., day of the week and time-slots, may also affect the decision and are also considered. A single recommendation model is

[3] http://www.meetup.com
[4] http://www.last.fm

tailored to each signal and the different models are used as features for learning to rank events. Experiments on data from Meetup demonstrate that this approach outperforms a Context-Aware RS (CARS) based on MF with social regularization.

Relation to our work. While event recommendation is the closest topic related to our work in this chapter, the problem we address remains significantly different than the formulation adopted in event recommendation. In EBSN, recommendations are made to users who are actually interacting on the social network and expressing their interest towards events shared on the platform. In the problem we are addressing, users' feedback about events is not available. In addition, event recommendation is used to enhance tourism planning and hotel recommendation, and not for the only purpose of recommending events.

5.3 Problem Formulation

Given that events have an important attractive power in the tourism sector, our idea is to investigate the benefits of introducing open data related to events to generate hotel recommendations. We consider the cases where the travelers' visits to hotels are motivated by events organized nearby that they plan on attending. Direct feedback about events is not explicitly provided but is rather inferred from another source of information: the history of bookings.

Motivation. Exploiting event data in the context of hotel recommendation can be beneficial to suggest packages of hotels and events. Interesting hotels can be proposed to users when an event they are likely to attend is organized nearby. On one hand, this facilitates trip organization for users. On the other hand, it offers a solution for hotel organizations to refine their marketing campaigns by specifically targeting interested users when a specific event is scheduled in the hotels' neighborhood. Pushing hotel recommendations coupled with events is also a way of explaining the suggestions and justifying the proposed selection of hotels (Section 2.2.2).

Formal definition. We formulate our problem as two subproblems, occurring in two different use cases. Given a hotel booking dataset gathering the set of hotels a traveler visited and given an event dataset underlining the characteristics and scheduling of events, the two subproblems are defined as follows:

- **The *hotel-centric* problem.** The main goal is to suggest hotels matching the travelers' preferences. Hotel recommendations are accompanied by suggestions of activities to do in the surroundings of the hotels such as attending a specific event. An example of recommendation in the *hotel-centric* setting is provided in Figure 5.1a.
- **The *event-centric* problem.** When tracking a major event, the set of potentially interested travelers are targeted in order to draw their attention to the event. To facilitate tourism planning, close hotels where they could stay are then proposed. An example of recommendation in the *event-centric* setting is provided in Figure 5.1b.

(A) Hotel-centric recommendation (B) Event-centric recommendation

FIGURE 5.1: Examples of hotel-centric and event-centric recommendations

Impact on the recommendation performance. Understanding the motivation behind the traveler's trip is important and helps improve the quality of recommendation. Generating better recommendations facilitates tourism planning. In addition, pushing events coupled with hotel recommendations is a way of explaining the choice of the selected hotels, and this is in particular valid in the *event-centric* problem. The impact of this work goes beyond improving the RS accuracy and affects other aspects of the RS like explaining and diversifying recommendations.

5.4 Data Collection and Analysis

This section presents the datasets used in this chapter, highlighting some challenges of the studied problem.

5.4.1 Event Dataset

Dataset collection. Data related to events were collected via the API of Eventful [5] from January 2015 to March 2017. Eventful is an open platform and gathers a large collection of events including concerts, sports, and conventions, posted and promoted by users of the social network. An event is described by the following features: title, venue, performers, start time, duration, entry price, and textual description. The content published on the

[5] http://api.eventful.com

Table 5.1: Examples of crawled events from Eventful, considered as major events susceptible of attracting travelers

Title - *Venue*
Rihanna - *Wembley Stadium, Wembley, Brent, UK*
New York Knicks vs. Chicago Bulls - *Madison Square Garden, New York, NY, US*
Final Euro: Portugal vs France - *Stade de France, Saint-Denis, FR*
Salon Livre Paris 2016 - *Paris Expo Porte de Versailles, Paris, Île-de-France, FR*
Ed Sheeran - *Sydney Entertainment Centre, Sydney, NSW, AU*
Volleyball Men's Gold Medal Olympic Games - *Maracanãzinho, Rio de Janeiro, Rio de Janeiro, BR*

platform being generated by users, it is essential to filter, clean, and enrich the crawled data before injecting it into the RS.

Filtering major events. From our perspective, some events posted on Eventful are considered irrelevant as they do not play any role in attracting travelers from abroad. They represent a gathering of people sharing the same interest on a local scale. It is important to discard these events by filtering the crawled data, thus reducing the noise added to the RS. The task of identifying attractive events is not trivial and requires a good understanding of event tourism. We base our reasoning on the idea that major events are usually organized in a limited set of famous and well-known venues. Therefore, we filter events based on the places where they are organized. We rely on knowledge bases in order to get background information about venues. In particular, we use DBpedia [Lehmann et al., 2015] which extracts structured information from Wikipedia. We retrieve the entities belonging to the classes of *venues* and *sports facilities*, gathering locations where events are most likely to take place.

In order to be considered as famous, we suppose that a venue must verify one of the two criteria defined in the following. First, the Wikipedia page describing the venue must be translated in a minimal number of languages, attesting that the venue is world-famous or at least famous in a part of the world and that it is attracting travelers from outside the region. Second, the venue's capacity, e.g., seating capacity, should be large enough proving that it can welcome a great number of travelers. Table 5.1 and Table 5.2 show examples of filtered and non-filtered events respectively, where events from Table 5.1 are considered as susceptible of attracting travelers.

Annotating events with featured performers. Events are available within a rather short time span. The attention they draw and the interest they create are generally due to two factors: the venue where they are organized and the performers they feature. Linking future events of a performer to its past occurrences is essential to be able to measure similarities between events and to model travelers' preferences for events.

TABLE 5.2: Examples of crawled events from Eventful, considered as local events that do not have a major impact on travelers

Title - *Venue*
Dine & Dance - *Cafe Taste, Toronto, Ontario, CA*
Yoga for a Stressful World - *Manor Woods Elementary School, Ellicott City, MD, US*
Learn to Swim - *Curie Park, Chicago, IL, US*
Drinks at the Refinery - *The Refinery, Hackney, Hackney, UK*
Les Femmes Savantes - *L'Aqueduc, Dardilly, Rhône-Alpes, FR*
Atelier Créativité et Bien-Être - *Dorémifasoleil, Muret, Picardie, FR*

TABLE 5.3: Statistics of the crawled dataset, EVENT-FULL, and the filtered dataset, EVENT

Dataset	EVENT-FULL
# events	2,447,088
# venues	410,217
# performers	34,787
# categories	30
# represented countries	122
# languages used	14
average # of events per day	3,042
Dataset	EVENT
# events	38,377
# venues	802
# performers	17,951

A small number of the crawled events from Eventful is annotated with performers (less than 7%) as the majority of them mentions in their title almost everything the attendees need to know. We use several methods in order to identify the events' performers based on their title. Targeting performers of music events, we extract from DBpedia the names of *artists* and *groups* and we try to find matchings in the events' titles. Concerning sports events, we identify tokens, e.g., *versus* and *vs*, allowing to parse the titles and select the multiple entities participating in the event.

Dataset size. Initially, around 2.5 million events were collected and constitute the dataset denoted by EVENT-FULL. These events are organized in 410,217 venues and belong to 30 different categories including concerts, sports, food, education, and science. After filtering the crawled venues and the corresponding events, we are left with approximatively 1.56% of the events and 0.2% of the venues. The filtered dataset is denoted by EVENT and is used in the rest of this chapter. Further information about the event datasets is provided in Table 5.3.

TABLE 5.4: Statistics of the booking dataset AH-EVENT used for recommending packages of hotels and events

Dataset	# users	# items	# transactions
AH-EVENT	2,096,622	4,574	10,213,627

5.4.2 Booking Dataset

The booking dataset used in this chapter is created the same way the dataset AH, described in Section 3.2, was created, with two distinctions. First, we only select bookings made during the period covered by the event dataset. This is mainly because events organized before the date at which we started collecting event data could not be retrieved. Second, we only consider users that can be associated with at least three events in order to have enough observations to model preferences. The procedure of associating users with events is detailed in the next section. Further information about the resulting dataset, denoted by AH-EVENT, is provided in Table 5.4.

5.5 Proposed Framework

In this section, we describe the main modules of our framework which is designed to tackle the *hotel-centric* and *event-centric* problems defined in Section 5.3. The final goal is to propose to each user a personalized list of pairs of hotels and events where the event is organized in the surroundings of the corresponding hotel and can be attended by the hotel's visitors. In the *hotel-centric* problem, we want to suggest hotels that match the user's preferences. Events are then proposed to entertain users and enhance the tourism experience. In the *event-centric* problem, we draw the user's attention to a future event and propose a hotel as part of the trip planning.

5.5.1 Overview

The proposed framework is constituted of five modules and its architecture is presented in Figure 5.2. It is flexible in the sense that the methods used in each module can be modified, improved, or even discarded, as long as the output of the module is provided in one way or another. We base our work on the assumption that each booking made by a traveler in any of the hotels was motivated by an event organized nearby in the same period. This assumption is essential in order to simplify our work, knowing that the real motivation behind each booking is unknown and that direct feedback about events is not provided. The bookings that do not respect this assumption in practice are not expected to drastically affect the performance of the RS since we are also capturing and leveraging preferences for hotels derived from real interactions with hotels.

In order to associate each booking with its motivating event, we proceed in two phases. In the first phase, we select all the events that a user could have physically attended during his stay, based on location and time parameters (module 1, Section 5.5.3.1). In the second

phase, we filter one of these events by taking into account the whole user profile and by ensuring a degree of cohesiveness across the profile (module 3, Section 5.5.3.3). This step requires the computation of similarities between events, measuring to which extent two events may interest the same person (module 2, Section 5.5.3.2). The set of interactions user-hotel and user-event are used to learn user preferences for events and hotels (module 4, Section 5.5.3.4). Recommendations in the *hotel-centric* and *event-centric* approaches are generated based on these preferences and on the schedule of future events (module 5, Section 5.5.3.5).

5.5.2 Notations and Definitions

Following the notations defined in Section 2.1.4 (refer to Table 2.1), we denote by $\mathcal{U}$ the set of users, $\mathcal{I}$ the set of hotels, and we introduce the notation $\mathcal{E}$ for the set of events. We extend the original notation of the set of items rated by a user u, $\mathcal{I}_u$, to include triplets (i, d_{in}, d_{out}) where i designates the hotel visited by u, d_{in} the check-in date, and d_{out} the check-out date. An event $e \in \mathcal{E}$ is organized in the venue v_e and features a set of performers denoted by $\mathcal{F}_e = \{f_1, f_2, ..., f_{m_e}\}$, where m_e is the number of performers of e. One part of the work consists in associating each booking with the corresponding event, principal motivator of the trip. To this end, we introduce for each user u two types of profiles described in the following:

- The user's **all-inclusive profile.** In the all-inclusive profile, denoted by $\mathcal{P}_u^{all}$ for user u, each booking $b \in \mathcal{I}_u$ is associated with the events that could be the potential motives of the trip based on location and time parameters. For each $b = (i, d_{in}, d_{out})$, $\mathcal{P}_u^{all}$ includes a tuple defined as $< i, d_{in}, d_{out}, e_1, e_2, ..., e_l >$, where $e_1, e_2, ..., e_l$ are the events that occurred in the neighborhood of the hotel i in a period close to the one bounded by (d_{in}, d_{out}). Each user u is assigned one profile $\mathcal{P}_u^{all}$ gathering information about the bookings previously made.
- The user's **limited profile**. The limited profile, denoted by $\mathcal{P}_u^{lim}$ for user u, is derived from the all-inclusive profile $\mathcal{P}_u^{all}$. Following the assumption made in Section 5.5.1, we select exactly one event for each booking $b \in \mathcal{I}_u$ and rely on a similarity measure between events to do the selection. Tuples in $\mathcal{P}_u^{lim}$ are in the form of $< i, d_{in}, d_{out}, e >$. $\mathcal{P}_u^{lim}$ is then used to learn preferences of u for hotels and events.

5.5.3 Modules

The whole process is carried out through five modules detailed in the next subsections and represented in Figure 5.2. We explain the functionalities of each module pointing out the required inputs, the generated output, and the motivation and intuitions that lead to the proposed methods.

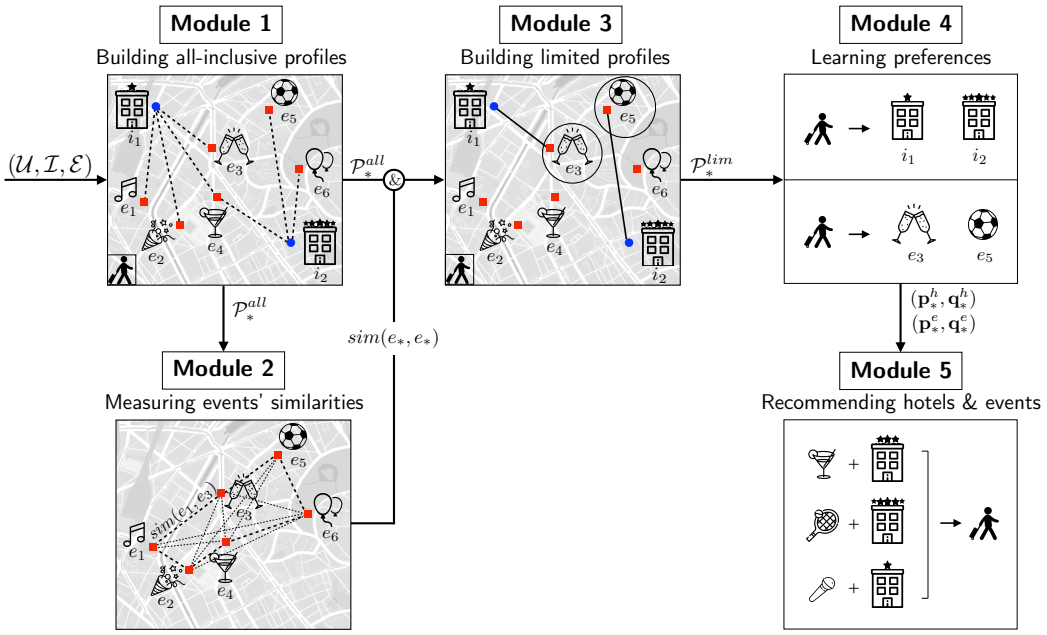


FIGURE 5.2: Architecture of the proposed framework for recommending hotels and events

5.5.3.1 Building All-Inclusive Profiles Based on Location and Time

Input. Booking data, event data, and hotels' locations.

Output. Users' all-inclusive profiles.

Functionality. This module builds users' all-inclusive profiles by linking bookings and events based on the venue's location, the hotel's location, the event's date, and the period of the stay. More formally, we associate each booking $b \in \mathcal{I}_u$, where $b = (i, d_{in}, d_{out})$, with the set of events $\{e_1, e_2, ..., e_l\}$.

Motivation. When travelers organize their trips and plan on attending an event, they search in general for hotels located in the neighborhood of the event's venue. From a hotel organization perspective, experts track the set of venues that impact the hotel demand and performance in order to better handle unusual activities.

We determine for each venue the set of hotels affected by the major events organized in the venue. These hotels are specifically the ones targeted by travelers when attending the venue's events. In the simplest case, the distance separating venues and hotels can be used to model the fact that the hotel is impacted or not by the venues' activities. We define a threshold distance, denoted by $dist(v_e)$ for the venue v_e, such that each hotel located within this distance from v_e is affected by it. Several factors influence $dist(v_e)$. In particular, $dist(v_e)$ depends on the density of hotels located in the surroundings of the venue. For venues located in big cities where hundreds of hotels are concentrated in a relatively small zone, the impacted region is smaller than the one for venues located in less crowded cities or isolated regions. Travelers are more likely to stay in a hotel far from the event's venue in a region where hotels are less implanted.

Method. Following these intuitions, we introduce some notations before defining $dist(v_e)$:

- $dist_{max}$ is the maximal distance separating v_e from a potentially impacted hotel. It is assigned a default value, i.e., 30 kilometers, with the help of domain experts, that is independent of v_e;
- $n_I(v_e)$ is the number of hotels located within a distance equal to $dist_{max}$ from v_e;
- $dist_{min}(v_e)$ is the minimal distance separating v_e from any hotel, i.e., the closest hotel to v_e.

The threshold distance $dist(v_e)$, defined in the following, ranges between $dist_{min}(v_e)$ and $dist_{max}$ and decreases when the density of hotels increases:

$$dist(v_e) = dist_{min}(v_e) + (dist_{max} - dist_{min}(v_e)).e^{(-\lambda.n_I(v_e))} \tag{5.1}$$

where λ is a positive constant set to 0.01 in our work.

Going back to the initial problem, let $\mathcal{V}_i$ refer to the set of venues impacting the hotel i where the booking b has been made. The events associated with b are those organized in any $v \in \mathcal{V}_i$ during the period $(d_{in} - (1\ day), d_{out} + (1\ day))$. It is worth mentioning that further developments in this direction could consider the automatic identification of the impact of events on hotels by tracking the change of the occupancy rate and the room prices based on the occurrence of events.

5.5.3.2 Measuring Events' Similarities

Input. Users' all-inclusive profiles.

Output. Similarities between pairs of events.

Functionality. The main functionality of this module is to compute similarities between events, i.e., compute $sim(e_a, e_b)$ for each couple of events $(e_a, e_b) \in \mathcal{E} \times \mathcal{E}$. This measure is later used to derive limited profiles $\mathcal{P}_*^{lim}$ from all-inclusive profiles $\mathcal{P}_*^{all}$.

Motivation. Intuitively, two events are similar if they share the same or similar features. A sporting event is more similar to another event from the sports category than to a concert or a convention belonging to other categories. Two events happening in the same venue or featuring the same performers are also supposed to be related. Given the limited availability of features, another way to assess similarities is to rely on people's feedback. Two events are similar if they are attended by the same people. Evaluating the similarity between two entities requires representing them in a common space where it is possible to compare them. This common space can be composed of events' features or latent features learned from the data. The similarity between events is modeled as a weighted aggregation of the similarity between venues and the similarity between performers.

Method. Assuming we have the venues' and performers' representations, we use the cosine similarity (Section 2.5.1.1) to measure the similarity between venues of two events e_a and e_b, i.e., $sim_v(e_a, e_b)$, and between the corresponding performers, i.e., $sim_f(e_a, e_b)$. $sim(e_a, e_b)$

is then given as follows:

$$sim(e_a, e_b) = \alpha.sim_v(e_a, e_b) + (1 - \alpha).sim_f(e_a, e_b) \tag{5.2}$$

where the parameter $\alpha \in [0, 1]$ determines the contribution of each type of similarity in the final measure. We rely on the all-inclusive profiles of users $\mathcal{P}_*^{all}$ to learn the representations of venues and performers, leveraging the wisdom of the crowd rather than the explicit set of relatively limited features. The problem is similar to the one we face in RS and, in particular, in MF approaches (Section 2.5.2), where we map users and items to a joint latent factor space. We apply a MF technique on the set of user-venue interactions and user-performer interactions respectively to model venues and performers in latent factor spaces.

5.5.3.3 Building Limited Profiles Based on Cohesiveness

Input. Users' all-inclusive profiles and events' similarities.

Output. Users' limited profiles.

Functionality. This module generates the limited profiles $\mathcal{P}_*^{lim}$. We identify for each booking exactly one event that could have been the main motivation of the trip. We take into account the whole set of bookings $\mathcal{I}_u$ for a user $u \in \mathcal{U}$ and the similarities between them.

Motivation. The idea is to select one event for each booking — the event that motivated the trip — such that the events selected for all the bookings of one user are the most similar. We suppose that a user is more likely to be interested in similar events (the similarity being defined in Section 5.5.3.2). Starting from the user's all-inclusive profile $\mathcal{P}_u^{all}$, the aim is to pick one event for each tuple from $\mathcal{P}_u^{all}$ such that the probability of this event belonging to $\mathcal{P}_u^{lim}$ is maximal given the other elements of $\mathcal{P}_u^{lim}$. This probability depends on the events picked from the other tuples of $\mathcal{P}_u^{all}$. The selected events should maximize the cohesiveness of elements occurring in $\mathcal{P}_u^{lim}$.

Method. The ideas formulated above are resumed in the following assumption we make. The events selected from $\mathcal{P}_u^{all}$ and included in $\mathcal{P}_u^{lim}$ maximize the sum of similarities between all pairs of events. Due to the complexity of the optimization problem, we turn to graph theory to solve it.

We build a graph for each user $u \in \mathcal{U}$ and develop an algorithm that selects one event per stay with respect to our motivation. Let G_u be a weighted complete graph associated with user u where $G_u = (N_u, E_u)$, N_u is the set of nodes, and E_u is the set of edges. When building the graph, we iterate through the tuples of $\mathcal{P}_u^{all}$ and add to G_u one node for each event appearing in each tuple. Each node is attributed a label and nodes derived from the same tuple are assigned the same label. After iterating through all the tuples of $\mathcal{P}_u^{all}$ and adding the nodes to G_u, we weight each edge connecting two nodes corresponding to the events e_a and e_b with the value $sim(e_a, e_b)$.

The problem we are trying to solve in this module is nearly equivalent to the Heaviest-k-Subgraph (HkS) problem with a few distinctions. Originally, the HkS problem can be seen

as the weighted version of the well-known densest subgraph problem [Lee et al., 2010] with restrictions on the size of the subgraph. In the HkS problem, k being an integer and $k > 1$, the task is to find a subgraph containing k nodes such that the sum of the weights of its edges is maximal. Our setting imposes one additional constraint: each node of the subgraph has to carry a unique label. k is also equal to the number of tuples of $\mathcal{P}_u^{all}$, knowing that we want to select one event per tuple, i.e., one node for each label. The nodes of the HkS selected by the algorithm correspond to the events that will be included in $\mathcal{P}_u^{lim}$. Inspired by the work in [Letsios et al., 2016], we develop a branch and bound algorithm in order to solve the problem.

The branch and bound algorithm is used to solve combinatorial optimization problems. The search space is divided into several branches of computation. The set of candidate solutions is considered to be forming a tree with the full set at the root and smaller sets of solutions as children. The algorithm explores branches of the tree and check each one of them against upper and lower estimated bounds of the optimal solution. If the upper bound of a node is lower than the global lower bound, i.e., the maximum lower bound over the set of candidate solutions considered, the branch is pruned and its children are not explored.

In our method, the branching phase consists of deciding whether to add a node to the optimal solution or not. While iterating over the existing labels, we decide which of the corresponding nodes could be considered in the optimal solution and add the corresponding branch to the tree. The lower bound is given by the sum of the weights of the edges of the subgraph included in the candidate solution. The upper bound is the sum of the weights of the edges connected to the examined node when considering the heaviest edges connecting the node to another one having a different label.

5.5.3.4 Learning Preferences for Hotels and Events

Input. Users' limited profiles.

Output. Latent representations of users, hotels, and events.

Functionality. We use user-hotel and user-event interactions derived from $\mathcal{P}_*^{lim}$ to learn user preferences for hotels and events.

Motivation. This module handles a classical recommendation problem and we count on user interactions to learn preferences for items and make recommendations.

Method. We rely on MF (Section 2.5.2) to learn the latent representations of users, hotels, and events, and to evaluate the score of a hotel and an event for a user. We build two models based on the sets of user-hotel interactions and user-event interactions respectively, modeling the preferences for hotels and events.

5.5.3.5 Recommending Hotels and Events

Input. Latent representations of users, hotels, and events, and schedule of events.

Output. Pairs of hotels and events recommendations.

Functionality. This module generates personalized recommendations of pairs of hotels and events for each user. It handles both the *hotel-centric* and *event-centric* problems, addressing each one of them differently.

Motivation. In the *hotel-centric* problem, we focus on recommending hotels that match the user preferences. We anticipate the need for looking for events to attend for entertainment purposes and we also propose an event organized nearby. In the *event-centric* approach, we create the need for a user to take part in an event and subsequently search for a hotel by drawing his attention to a specific event.

Method. The computed scores are used to rank the list of items for one user and the top-N items of the list, i.e., items that score the highest, are selected for recommendation. In the *hotel-centric* problem, we first generate hotel recommendations for one user. We use these recommendations to filter the ranked list of events where we discard events that are not linked to any of the chosen hotels and that cannot be proposed together (Section 5.5.3.1). Filtering the ranked list of events allows the selection of one event per hotel, forming the pairs to recommend. In the *event-centric* problem, we proceed inversely: We first generate events' recommendations that are then used to filter the ranked list of hotels.

5.6 Experimental Results

In order to demonstrate the proposed framework, we proceed in two steps. We first illustrate the functioning of the framework using concrete examples. We introduce simulated users and check the output produced by a number of modules. A quantitative evaluation is then performed: Offline experiments are conducted where we assess the accuracy of the RS as a whole and point out the interest of our work. It is worth mentioning that the results depend on the set of events covered and on the worldwide availability of hotels included in the datasets. Some interesting events may not be posted on Eventful and some bookings motivated by events may not be made in the hotels considered.

5.6.1 Qualitative Evaluation Through Concrete Examples

To gain a better insight into the presented framework, we simulate users with their history of bookings and evaluate the results. These results concern the computation of similarities between different entities, i.e., venues, performers, and events. We also generate recommendations for the users and highlight the content of the intermediate structures and profiles that lead to the final solution.

Similarities between venues. Table 5.5 contains nearest neighbors of two examples of venues sampled from the set of 800 venues. The first venue designates a stadium located in France and its nearest neighbors are stadiums mainly located in Europe. The second venue is a concert hall located in Luxembourg and its nearest neighbors cover European concert halls, theaters, and music venues.

TABLE 5.5: Nearest neighbors of examples of venues

Stade de France - *Saint-Denis, FR*
Stadio Dino Manuzzi - *Cesena, IT*
Cornaredo Stadium - *Lugano, CH*
Estadio José Rico Pérez - *Alicante, ES*
Ostseestadion - *Rostock, DE*
KeyArena - *Seattle, WA, US*
Philharmonie de Luxembourg - *Luxembourg, LU*
Courtyard Theatre - *London, GB*
Theater des Westens - *Berlin, DE*
Laeiszhalle - *Hamburg, DE*
Royal Opera House - *London, GB*
La Monnaie - *Brussels, BE*

TABLE 5.6: Nearest neighbors of examples of performers

Paris Saint-Germain	**Manchester United FC**
Girondins de Bordeaux	Crystal Palace FC
FC Lorient	Southampton FC
Lille LOSC	West Ham United FC
OGC Nice	Manchester City FC
Angers SCO	Tottenham Hotspur FC
Taylor Swift	**Skillet**
Chris Brown	Rammstein
Sam Smith	Last Train
Neil Diamond	Baby Metal
Elton John	Mass Hysteria
Robbie Williams	Jane's Addiction

Similarities between performers. Table 5.6 shows the nearest neighbors of some performers sampled from the set of 18k performers. Computing similarities between performers allows gathering French football clubs (first example), English football clubs (second example), pop singers (third example), and rock/metal bands (fourth example). A traveler interested in a specific performer is most likely to enjoy events featuring one of its neighbors.

Similarities between events. We combine venues' and performers' similarities to measure similarities between 38k events and we illustrate examples in Table 5.7. In a setting where we use events' features, e.g., category and performers, to compare events, we are not able to evaluate the resemblance between events that do not share the same features. However, the proposed method allows estimating if a traveler that has previously attended a football match is likely to attend a concert featuring a specific performer or any other event.

Recommendations of hotels and events. Table 5.8 shows examples of recommendations for two simulated users based on their history of bookings. The all-inclusive profile of a user gathers all the events listed for all of his bookings. Only one event per booking, marked in bold, is included in the limited profile. *User 1* is expected to have mainly attended

TABLE 5.7: Nearest neighbors of examples of events

OM - Paris Saint-Germain - *Stade Velodrome, FR*
Olympique Marseille - Rennes - *Stade Velodrome, FR*
Auxerre - Paris Saint-Germain - *Stade de France, FR*
Nimes Olympique / Paris FC - *Stade des Costières, FR*
Monaco vs Paris Saint-Germain - *Stade Louis-II, MC*
Montpellier - OM - *Stade de la Mosson, FR*
Ed Sheeran - *Sydney Entertainment Centre, AU*
Sam Smith - *Hodern Pavilion, AU*
Kylie Minogue - *Sydney Entertainment Centre, AU*
5 seconds of summer - *Hodern Pavilion, AU*
Coldplay - *Allianz Stadium, AU*
The Pink Floyd Experience - *Enmore Theatre, AU*
Swan Lake - *Edinburgh Playhouse, GB*
The Nutcracker - *Edinburgh Playhouse, GB*
Giselle - *Edinburgh Playhouse, GB*
Nicola plays Beethoven - *Usher Hall, GB*
To kill a mockingbird - *Kings Theatre, GB*
Rameau & Charpentier, Orchestra - *Usher Hall, GB*
4th eScience Symposium - *Amsterdam Arena, NL*
MaaS Meetup #2 - *Amsterdam Arena, NL*
Amazing d̈atam̈onday - *Amsterdam Arena, NL*
Inspiration 360 - *Ziggo Dome, NL*
Big Data Small World - *Amsterdam Arena, NL*
SolarEdge Advanced Training - *DSB Stadion, NL*

football events in European venues. He is recommended to go to the *FA Cup* event held in Manchester, England, among other events. *User 2* is expected to be rather interested in concerts featuring a specific genre of artists and is therefore recommended to attend the concert of *Chris Brown*, for example.

5.6.2 Quantitative Evaluation Through Offline Experiments

The proposed framework is evaluated by measuring the accuracy of the recommendations generated by the RS. The aim is to assess the influence of using event data on the quality of hotel recommendation. The evaluation focuses on the *event-centric* problem described in Section 5.5.3.5.

Evaluation scheme. The booking dataset, AH-EVENT, and the event dataset, EVENT, are used for evaluation (Section 5.4). AH-EVENT is split into a training set used as input to the framework and a test set to which we compare recommendations. We perform a chronological split (Section 2.2.1.1): The test set consists of the bookings recorded during the last 6 months since the most recent booking and the training set gathers the older bookings. Events considered for recommendation are those planned in the same time period.

User 1: History of Bookings	
Bookings	**Event - *Venue***
Jan. 12-17, 2015, Mercure Sydney	Roger Federer, Lleyton Hewitt, Sydney Exhibition Match - *Sydney Entertainment Centre*
	The 1975 Sydney - *Hordern Pavilion*
	Sydney Festival 2015: Atomic Bomb! The Music of William Onyeabor - *Enmore Theatre*
Aug. 21-24, 2015, Ibis Paris Saint-Denis	**France vs. England - *Stade de France***
Jan. 21-24, 2016, Ibis Styles London Kensington	'Responding to the new landscape' Delivering Better Outcomes for All - *Barbican Centre*
	Looking Past Degrees: How to find the Best Young Talent - *Barbican Centre*
	Nahko and Medicine for the People - *Shepherd's Bush Empire*
	Big Sing Fridays - *Royal Opera House*
	Who's Next - *The 100 Club*
	A Postcard to Bill Evans - *Southbank Centre*
	Tales from Hollywood - *Donmar Warehouse*
	Kevin Hart: What Now? Tour - *SSE Arena Wembley*
	The Complete Stone Roses - *The 100 Club*
	Crystal Palace vs. Tottenham - *Selhurst Park*
	RSC Shakespeare on Screen: King Lear (1971) - *Barbican Centre*
Mar. 17-20, 2016, Ibis Manchester Portland	**UEFA Europa League: Manchester United FC vs. Liverpool FC - *Old Trafford***
	Cosi Fan Tutte Manchester - *The Lowry*
	Michael Morpurgo: Where My Wellies Take Me - *The Lowry*
	Manchester City vs. Manchester United - *Etihad Stadium*
	The Edgeley Park Stadium Antiques & Collectors Fair - *Edgeley Park*

User 1: Recommendations	
Event Date	**Event - *Venue* - Hotel**
Dec. 2, 2016	Rod Stewart - *Barclaycard Arena* - Ibis Rotherham
Dec. 13, 2016	Coldplay Sydney - *Allianz Stadium* - Ibis Sydney Darling Harbour
Mar. 1, 2017	FA Cup 2016/17 - Manchester City v Huddersfield - *Etihad Stadium* - Ibis Manchester Princess Street
Mar. 11, 2017	Melbourne Victory vs Central Coast Mariners - *Aami Park* - Ibis Little Bourke Street

User 2: History of Bookings	
Bookings	**Event - *Venue***
May 28-31, 2015, Novotel Amsterdam City	**Ariana Grande - *Ziggo Dome***
	Toto - *Ziggo Dome*
	Armin Van Buuren & Nicky Romero, Amsterdam festivals - *Olympic Stadium*
Oct. 17-18, 2015, Ibis Styles Stuttgart	**Imagine Dragons - *Porsche-Arena***
	VfB Stuttgart vs. FC Ingolstadt 04 - *Mercedes-Benz Arena*
Feb. 12-15, 2016, Mercure Bad Duerkheim	**WWE Live - *SAP Arena***
	Mario Barth: Männer sind bekloppt, aber sexy! MANNHEIM - *SAP Arena*
Mar. 08-09, 2016, Mercure Amsterdam Slo.	Debat "Talent naar de regio NHN"+BNR Gangmakers live uitzending - *DSB Stadion*
	Muse - *Ziggo Dome*

User 2: Recommendations	
Event Date	**Event - *Venue* - Hotel**
Oct. 10, 2016	The Legend of Zelda - *Porsche-Arena* - Mercure Stuttgart City Cent
Nov. 6, 2016	Chris Brown - *Ziggo Dome* - Novotel Amsterdam Schiphol
Nov. 9, 2016	WWE Live - *Mercedes-Benz Arena* - Mercure Stuttgart Apt Messe
Nov. 10, 2016	The Cure Cologne - *Lanxess Arena* - Mercure Duesseldorf Hafen

Table 5.8: Two examples of users' history of bookings, the association with events, and generated recommendations

Table 5.9: Recall@N and NDCG@N for `MFev` and other recommendation approaches

Metric	`MostPop`	`Knnu`	`WRMF`	`BPR`	`MFev`
Recall@5	0.0121	0.0542	0.0634	0.0722	**0.0865**
Recall@10	0.0218	0.0685	0.0895	0.1042	**0.1197**
NDCG@5	0.0188	0.0318	0.0473	0.0582	**0.0708**
NDCG@10	0.0243	0.0407	0.0512	0.0696	**0.0892**

Metrics. We use recall@N and NDCG@N, defined in Section 2.2.3, to measure the recommendation accuracy, where N is the number of recommended items. Higher values for both metrics indicate a better recommendation quality. Since large values of N are not interesting for recommendation, we set N to 5 and 10.

Parameters. A grid search over the parameter space of each evaluated method is performed in order to find the parameters that give the best performance which we report.

Methods compared. The following methods are included in the comparison:

- `MostPop`. The N most popular hotels are recommended to users.

- `Knnu`. `Knnu` is the user-based neighborhood method, described in Section 2.5.1.1. We use the Jaccard similarity metric to measure similarities between users. We set the number of neighbors to 1000.
- `WRMF`. Weighted Regularized Matrix Factorization [Hu et al., 2008] is a MF technique specifically designed to handle implicit feedback and presented in Section 2.5.2.4. We set the number of latent factors to 100 and the regularization parameters to 0.001.
- `BPR`. Bayesian Personalized Ranking [Rendle et al., 2009] is a MF technique presented in Section 2.5.2.4. We set the number of latent factors to 100 and the regularization parameters to 0.0025.
- `MFev`. This is the approach we propose to solve the *event-centric* problem, leveraging event data for hotel recommendation. `MFev` (in particular, modules 4 and 5, Section 5.5) is based on `BPR` since it outperforms the other methods for hotel recommendation on our dataset. The difference in performance we get by comparing `BPR` and `MFev` shows the benefits of incorporating event data using our framework.

Results. Table 5.9 shows the recall@5, recall@10, NDCG@5, and NDCG@10 for the approaches we compare. `BPR` performs better than the other recommendation methods that are only based on hotel bookings. `MFev` outperforms `BPR` which highlights the contribution of event data in contextualizing hotel bookings and generating better recommendations. Furthermore, `MFev` offers the advantage of explaining hotel recommendations by associating an event with each proposition. Providing explanations allow travelers to check the validity of the suggestions and helps them in their trip planning. While the usage of event data can be beneficial for hotel recommendation, a discussion related to the obtained results and the limitations they are subject to is proposed in the next section.

5.7 Discussion

People can be motivated by events to organize trips and are then required to choose an accommodation which is usually selected as the mean to achieve the goal. Our experiments have shown the interest of using event data when performing hotel recommendations. In this section, we summarize the contributions and we discuss how this solution can be used in real-world environments, its limitations, and possible future improvements.

Contributions. Even though it is well-known that events affect the hotel industry, the work presented in this chapter is the first attempt to model the impact of events on user behavior with respect to the choice of hotels and to benefit from this relation to improve recommendations. The problem addressed spans over two different domains. In the hotel domain, we have information about hotels and users' bookings while in the event domain, we only have information about events. The event domain is exploited in an attempt to improve recommendations in the hotel domain. We first formulate the problem addressed and provide instructions on how to collect, preprocess, and enrich event data to this purpose. We then propose a framework that is able to infer user preferences for events based on their history of bookings and leverage these preferences for recommending packages of hotels and events. Overall, the aim is to improve the tourism planning and the user experience.

Real-world applications. The designed approach is not meant to be the single one used to generate recommendations in a real setting. It can be interesting to rely on a comparable approach in one of the two use cases we defined and that occur in the hotel industry. First, when detecting an important future event, potentially interested users can be targeted. The proposed solution allows suggesting a hotel where they could stay, helping them in the trip planning. Second, for each hotel suggested, it is possible to propose an event that could be interesting for the user, improving the attractiveness of hotels being recommended and enhancing the user experience.

Limitations. We consider situations where people travel for the main purpose of attending an event, e.g., a concert or a conference, which explains why each booking is linked to exactly one event. Since only impactful events are filtered and considered, it is likely that travelers who stayed in the hotel near the event's venue during the corresponding period made the trip for the event. It is also worth mentioning that these travelers were willing to pay the high prices for the rooms compared to the regular periods. From a hotel industry perspective, we do not have explicit information concerning the event that motivated the trip, leading us to make assumptions about the potential link between bookings and events. Assuming that this information becomes available, e.g., when arriving at the hotel, the traveler declares whether he is attending an event or not, and which one it is, the performance of the system could be improved since the booking-event associations will be less noisy.

Improvements. Improvements in this direction should consider analyzing the behavior of subsets of users and detecting those who are particularly sensible to events. While events' type and hotels' category may have an impact on recommendation, it is hard to evaluate it in an offline setting as more feedback is required from the user. Therefore, online evaluation is a must to be able to draw more conclusions and generalize the results.

5.8 Conclusion

This chapter proposes the integration of open data related to events into hotel RS. We address two problems occurring in two different scenarios where the common task is to recommend packages of hotels and events in order to facilitate tourism planning. On one hand, we anticipate the need for looking for activities to do during the trip by proposing events organized near the hotels. On the other hand, we create the need for a traveler to attend an event and organize a trip by recommending interesting events in addition to hotels located near the events' venue. Experiments have shown the interest of using event data for hotel recommendation. Further advances should consider adopting online evaluation to measure the influence of the RS on the user behavior.

This work is also an attempt to combine insights from two different domains, i.e., the hotel domain and the event domain, to benefit one of them. Leveraging knowledge from several domains can empower RS, especially given that users are not only driven by the item characteristics and features which are included in the target domain. The next chapter explores the transfer of knowledge acquired in the domain of Location-Based Social Networks (LBSN) to the hotel domain, in order to enhance the recommendation quality.

Chapter 6

Transferring Context Knowledge Across Domains

This chapter presents our approach for exploiting knowledge about mobility extracted from Location-Based Social Networks (LBSN) to benefit the hotel RS that is in particular sensible to the geography dimension [Al-Ghossein et al., 2018c]. Two distinct domains are involved: the source domain, i.e., the LBSN where a set of users are sharing their check-in activity, and the target domain, i.e., the hotel domain where another set of users are booking and visiting hotels. Knowledge about the *physical* context is acquired in the source domain where the feedback is expected to be dense and is transferred to the target domain where it is thus considered to be *partially observable*. Experiments on real-world datasets highlight the implications of transferring knowledge between these domains on the recommendation quality.

6.1 Introduction

Traveling is still considered to be a rare activity that is carried out by individuals only a few times each year. As a result, the feedback collected in the hotel domain is sometimes not sufficient to learn user preferences. Sparsity constitutes a major limitation for Collaborative Filtering (CF) methods that are well-known for outperforming other recommendation approaches. One way to address the sparsity problem is to leverage knowledge from other related domains where it is easier to get information regarding the user behavior. Cross-domain RS [Cantador et al., 2015] take advantage of the abundance of heterogeneous data providing multiple views of user preferences. They aim to improve recommendations in a target domain by exploiting preferences uncovered in source domains. When applied in the tourism domain, cross-domain RS can suggest, for example, hotels based on flight bookings or events to attend based on hotel bookings.

When organizing a trip, travelers usually select the destination to visit before choosing the hotel where they will stay and the choice of accommodation highly depends on its location. Choosing a destination to visit is in turn related to several factors. First, the

majority of trips are meant to explore destinations that are close to the place of residence of travelers. Then, users tend to follow the actual trends running locally which are also likely to change with time. In addition, the timing of the trip has an impact on the chosen destination. Some destinations are more popular during summer than during winter, and leisure trips are more frequent during holidays.

Since hotel bookings are collected by organizations managing a subset of hotels and accommodations, hotel booking datasets do not cover all the trips done and the destinations visited by users, which participates in aggravating the sparsity issue. On the other hand, recent years have witnessed the emergence of Location-Based Social Networks (LBSN), e.g., Flickr and Foursquare, where users can connect with friends, upload photos, and share their locations via check-ins for Points Of Interest (POIs). LBSN constitute a rich data source to analyze travel experiences [Liu et al., 2017]. In particular, it is possible to capture the mobility of users through their check-in history.

In this chapter, we address the problem of hotel recommendation that is suffering from sparsity by leveraging check-in data from LBSN. We learn mobility patterns from the check-ins which are easily shared by users on LBSN and combine them with hotel preferences to boost hotel recommendations. We first map check-ins and hotels to a common space of geographical regions based on the density of hotels spread worldwide. We learn preferences for these regions, map users from both domains, and combine preferences for regions and for hotels to generate recommendations. Experiments using a dataset from the hotel industry and a dataset from a LBSN show the interest of using check-in data for hotel recommendation.

The rest of the chapter is organized as follows. In Section 6.2, we discuss related work on cross-domain recommendation. In Section 6.3, we present our approach for hotel recommendation leveraging mobility data from LBSN. Experiments, results, and a discussion are presented in Section 6.4. Finally, Section 6.5 concludes the chapter.

6.2 Cross-Domain Recommendation

The traditional recommendation problem focuses on recommending items within a single domain to users who have expressed their interest for a subset of these items. Given that users generally provide feedback for items in various domains, it may be beneficial to exploit all user data to enhance the recommendation quality. Cross-domain recommendation is based on this idea and represents a potential solution for the cold-start and sparsity issues [Li et al., 2009]. It also constitutes a practical application of transfer learning [Pan et al., 2010]. We provide in this section a short overview of existing work on cross-domain RS based on the categorization proposed in [Cantador et al., 2015] where further details can be found.

Domain definition. Different notions of *domains* were adopted in the literature. Overall, a single domain can be defined at the following four levels:

- *Attribute level.* Items are of the same type and have the same attributes. Items from different domains have different values for certain attributes, e.g., action movies and comedy movies.

- *Type level.* Items are of similar types and share some attributes. Items from different domains have different attribute subsets, e.g., movies and TV shows.
- *Item level.* Items are not of the same type, e.g, movies and books.
- *System level.* Items belong to separate systems considered as different domains, e.g., movies rated on the MovieLens RS [1] and movies watched on the Netflix streaming service [2].

Cross-domain recommendation tasks. Without loss of generality, cross-domain recommendation exploits knowledge from a source domain, denoted as $\mathrm{D_S}$, to benefit a target domain, denoted as $\mathrm{D_T}$. We denote by $\mathcal{U}_\mathrm{S}$ and $\mathcal{U}_\mathrm{T}$ the respective sets of users and by $\mathcal{I}_\mathrm{S}$ and $\mathcal{I}_\mathrm{T}$ the respective sets of items. Utilizing user data across different domains may particularly serve three recommendation tasks that are cited in the following:

- *Multi-domain recommendation.* The task consists in recommending items in both the source and target domains, i.e., recommending packages of items in $\mathcal{I}_\mathrm{S} \cup \mathcal{I}_\mathrm{T}$ to users in $\mathcal{U}_\mathrm{S}$ or users in $\mathcal{U}_\mathrm{S} \cup \mathcal{U}_\mathrm{T}$. This problem is strongly related to bundle recommendation where the joint suggestion of items from different domains can be beneficial for users.
- *Cross-selling.* The task consists in recommending items in the target domain by exploiting knowledge from the source domain, i.e., recommending items in $\mathcal{I}_\mathrm{T}$ to users in $\mathcal{U}_\mathrm{S}$ by exploiting knowledge about $\mathcal{U}_\mathrm{S}$ and $\mathcal{I}_\mathrm{S}$. Item cross-selling is interesting in the case of recommending items that require accessories, for example.
- *Linked domains exploitation.* The task consists in recommending items in the target domain by leveraging knowledge from the source and target domains, i.e., recommending items in $\mathcal{I}_\mathrm{T}$ to users in $\mathcal{U}_\mathrm{S}$ by exploiting knowledge about $\mathcal{U}_\mathrm{S} \cup \mathcal{U}_\mathrm{T}$ and $\mathcal{I}_\mathrm{S} \cup \mathcal{I}_\mathrm{T}$. Linking domains is beneficial to improve recommendation and to address the data sparsity problem.

Although specific approaches were proposed to tackle each task, the formulation of the cross-domain recommendation problem is considered to cover all of them.

Data overlap. Performing cross-domain recommendation requires some relations or overlaps between the different domains. Considering the two domains $\mathrm{D_S}$ and $\mathrm{D_T}$, there are four possible scenarios of data overlap [Cremonesi et al., 2011]:

- *No overlap.* There is no overlap between users and items from both domains, i.e., $\mathcal{U}_\mathrm{S} \cap \mathcal{U}_\mathrm{T} = \varnothing$ and $\mathcal{I}_\mathrm{S} \cap \mathcal{I}_\mathrm{T} = \varnothing$.
- *User overlap.* There exist some users that have interacted with items from both domains, i.e., $\mathcal{U}_\mathrm{S} \cap \mathcal{U}_\mathrm{T} \neq \varnothing$, but every item belongs to a single domain.
- *Item overlap.* There exist some items that have been rated by users from both domains, i.e., $\mathcal{I}_\mathrm{S} \cap \mathcal{I}_\mathrm{T} \neq \varnothing$.

[1] http://movielens.org
[2] http://netflix.com

- *Full overlap.* There is an overlap between users and items from both domains, i.e., $\mathcal{U}_S \cap \mathcal{U}_T \neq \varnothing$ and $\mathcal{I}_S \cap \mathcal{I}_T \neq \varnothing$.

Cross-domain recommendation approaches. Various techniques were proposed to address the cross-domain recommendation problem. The most recent categorization of approaches was presented in [Cantador et al., 2015] and is centered around the way knowledge is exploited, resulting in a two-level categorization: *aggregating knowledge* and *linking and transferring knowledge*.

Aggregating knowledge. Recommendations in the target domain may be enhanced by aggregating knowledge from source domains. This knowledge aggregation can be performed at different levels.

First, merging user preferences, e.g., ratings and transactions, helps in generating rich user profiles and is considered the most widely used strategy for cross-domain recommendation. Several recommendation approaches can then be applied to these aggregated preferences. [Berkovsky et al., 2007] rely on a user-based neighborhood method where similarities between users are computed based on the merged preferences and nearest neighbors are used for recommendation. [Cremonesi et al., 2011] present a graph-based approach where they build a graph having the nodes associated with items and the edges revealing item similarities. Edges between items from different domains are enhanced through strategies relying on the transitive closure. CF techniques leveraging the built graph are then evaluated. [Loni et al., 2014] apply an algorithm based on Factorization Machines (FM) where user preferences are mapped to latent features. While approaches based on merging user preferences are the simplest ones, they require user overlap between the source and target domains.

Second, knowledge aggregation can be applied at an intermediate level by mediating modeling data. In CF systems where users are shared across domains, one can assume that similar users in one domain are also similar in another one. [Shapira et al., 2013] compute nearest neighbors of users in the source domain and use them to generate recommendations in the target domain. This idea was also leveraged in model-based approaches for recommendation. [Low et al., 2011] develop a hierarchical probabilistic model where global and domain-specific user latent vectors are built to perform cross-domain recommendation. It is worth mentioning that these approaches require either user or item overlap.

Finally, item relevance estimations can be combined at the recommendation level. For example, aggregating user rating predictions is applied to the movie domain in [Berkovsky et al., 2007] where domains are defined according to movie genres. Recommendations are computed in source domains and aggregated in the target domain. There is user and item overlap since movies can be associated with various genres and users watch movies from multiple genres.

Linking and transferring knowledge. While aggregating knowledge requires at least a user overlap, knowledge linkage or transfer between domains may work without any overlap and is made possible through three variants of methods.

A first variant consists in linking domains based on some common knowledge between the source and target domains. To cope with the heterogeneity of domains, it is essential to

establish correspondences between entities belonging to the multiple domains. Once domains are linked, traditional recommendation approaches can be applied. Linking domains is made possible by exploiting the overlap of user or item attributes [Chung et al., 2007], the overlap of social tags [Fernández-Tobías et al., 2013], the overlap of textual contents [Berkovsky et al., 2006], or knowledge-based rules [Cantador et al., 2013]. These methods are designed for a particular cross-domain scenario and are difficult to generalize.

A second variant consists in sharing latent features between domains. One example is illustrated in [Pan et al., 2011] where the authors propose to learn latent features simultaneously in all domains. A tri-factorization method is presented: An additional set of factors is introduced to capture domain-dependent information and shared user and item factors are expected to generate observations in all domains. While the approaches based on sharing latent features succeed in increasing accuracy, they are computationally expensive and require the overlap of users or items, depending on the method.

Finally, the third variant consists in transferring rating patterns between domains. *Rating patterns* designate the fact that, even when there is no user and item overlap between domains, latent correlations may occur between preferences of sets of users for sets of items. [Li et al., 2009] co-cluster users and items in the source domain to extract rating patterns. The rating pattern representation, referred to as *codebook*, is used to transfer knowledge and compute ratings in the target domain. Further approaches extended the codebook idea [Gao et al., 2013; Moreno et al., 2012] while [Cremonesi and Quadrana, 2014] refuted it empirically, showing that it does not transfer knowledge when domains do not overlap.

Relation to our work. In this chapter, we apply cross-domain recommendation to alleviate the sparsity problem in hotel RS by leveraging check-in data from LBSN. The source and target domains we consider belong to two separate systems: The source domain is defined within the LBSN and the target domain is the hotel domain. The goal is to recommend hotels by *linking both domains*. In our setting, there is *no overlap* between users and items since users of the LBSN are different from users booking hotels and we are not able to identify common users, assuming they exist. In addition, while the first set of users is visiting POIs, the second one is booking and visiting hotels. Linking domains is made possible with respect to the geography dimension. To the best of our knowledge, this is the first work exploiting this dimension to establish the link between entities from both domains.

6.3 Proposed Approach

In order to cope with the sparsity problem faced in hotel RS, we propose to learn mobility patterns from check-ins shared on LBSN and combine them with hotel preferences to generate recommendations. In the source domain $\mathrm{D_S}$, we have active users on LBSN, $\mathcal{U}_\mathrm{S}$, who share their check-in activity. The items $\mathcal{I}_\mathrm{S}$ are the geolocated POIs. The target domain $\mathrm{D_T}$ is the hotel domain where users $\mathcal{U}_\mathrm{T}$ are the one booking hotels and the items to recommend, $\mathcal{I}_\mathrm{T}$, are the hotels. In the problem we are considering, there is no overlap between users from both domains as we are not able to link users posting on LBSN and users booking hotels. However, a mapping can be done between check-ins from $\mathcal{I}_\mathrm{S}$ and hotels from $\mathcal{I}_\mathrm{T}$

based on the corresponding locations. Similarities between users from both domains, $\mathcal{U}_S$ and $\mathcal{U}_T$, can also be computed based on the visited locations.

Our work is motivated by a number of ideas. First, our approach is inspired by the real decision-making process of users when choosing a hotel: They first select a destination to visit and then a hotel where they want to stay. The source domain contains the users' paths through their check-in activity. We try to use the knowledge from the source domain to learn accessible destinations for users. Accessibility is encapsulated in data from LBSN and it usually relies on the distance, cost, value, and other hidden variables. We are therefore interested in the mobility patterns at a high scale. In our problem, preferences for regions are more significant than preferences for specific POIs. Based on data from LBSN, we can get the set of regions visited by a user, for example, and use this information for hotel recommendation. Once the destination is selected, the hotel choice is more likely to depend on its features. On the other hand, hotels are not spread equally worldwide. When considering regions where there is a high density of hotels, it would be relevant to learn preferences for different subregions. Since all neighborhoods in a specific region do not have the same characteristics, travelers may prefer one over the other.

We consider therefore a decomposition of the world map in several regions where the region size depends on the corresponding hotel density. Items from both domains, $\mathcal{I}_S$ and $\mathcal{I}_T$, are associated with these defined regions. Using the behavior of users on LBSN, we learn the preferences of users to the regions. To benefit from these insights, and since there is no overlap between users from both domains, we associate users from the target domain, $\mathcal{U}_T$, with users from the source domain, $\mathcal{U}_S$, that are the most similar with respect to visited regions. Preferences for geographical regions and hotels are finally combined to generate hotel recommendations.

In the following, we detail each part of our proposed approach that is illustrated in Figure 6.2. A *recommendation method* designates any latent factor model [Koren et al., 2009] that can be used for uncovering latent factors representing users and items. Generating recommendations for a target user $u \in \mathcal{U}_T$ requires the computation of relevance scores $\hat{r}_{ui}$, for each hotel $i \in \mathcal{I}_T$. Hotels are ranked by decreasing order of $\hat{r}_{ui}$ and the N hotels that score the highest are selected for recommendation.

6.3.1 Mapping Items from Both Domains

Linking items from the source domain, i.e., LBSN, and the target domain, i.e., the hotel domain, is achieved by considering an intermediate layer of geographical regions to which items from both domains are mapped based on the corresponding locations. As mentioned before, the definition of regions relies on the distribution of hotels in the whole space. In areas where there is a high density of hotels, we want to define small regions to distinguish between preferences for each subset of hotel locations whereas in areas where there is a low density of hotels, bigger regions can be considered. We introduce therefore a decomposition of the world map in several regions where the region size depends on the corresponding hotel density. Items from both domains, $\mathcal{I}_S$ and $\mathcal{I}_T$, are then associated with these regions that form the set denoted by $\mathcal{I}_R$.

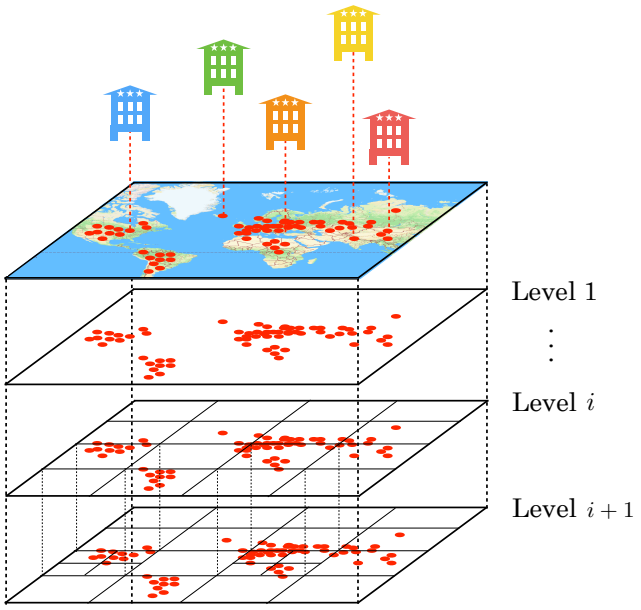


FIGURE 6.1: Hierarchical structure used to map items from the source and target domains

The region decomposition is inspired by *STING* [Wang et al., 1997], a statistical information grid-based method introduced for spatial data mining. STING is able to efficiently process region oriented queries, i.e., queries where it is required to select regions satisfying certain conditions, on a set of points. The idea is to divide the space into rectangular cells at different levels such that cells form a hierarchical structure. Statistical information about each cell is computed, stored, and then used to answer queries. STING has the advantage of being a query-independent approach and efficiently delivering answers.

Region decomposition. Inspired by STING [Wang et al., 1997], we assume the space is of two dimensions and we rely on a hierarchical division into rectangular cells represented in Figure 6.1 [Al-Ghossein and Abdessalem, 2016]. At the first level of the hierarchy, we have one cell, i.e., the root node, covering the whole region considered which is the whole map. Each cell at a higher level l is partitioned into 4 child cells at the next level $l+1$ and each child corresponds to one quadrant of the parent cell. The number of levels L is fixed and affects the size of the smallest region considered. Regions included in $\mathcal{I}_{\mathrm{R}}$ are cells selected from this hierarchy based on a density criterion and are expected to cover the whole space. The process of selecting relevant cells that will constitute the set $\mathcal{I}_{\mathrm{R}}$ is detailed in the following.

The process is comparable to clustering hotels where each cluster is represented by a cell of different size to ensure the coverage of the whole space and not only the space where hotels are located. Each cell is characterized by a density variable representing the density of hotels located in the region delimited by the cell. The density is defined as the number of hotels divided by the area of the cell. We follow a top-down approach based on the hierarchical structure of cells. Starting from the root cell, we first compute its hotel density. Moving to the next higher level, we compute again the hotel density for each child cell and compare it to the hotel density of the parent cell. If the hotel density of the child cell is

higher than the one of the parent cell, we add the child cell to $\mathcal{I}_{\mathrm{R}}$. Otherwise, we move to the next level and repeat the same process. The only difference is that we do not go through the cells that are already included in $\mathcal{I}_{\mathrm{R}}$. The procedure is maintained until we reach the maximum level L. The last step of the algorithm consists in going through the cells included in $\mathcal{I}_{\mathrm{R}}$ and merging those that are adjacent and that belong to the same hierarchy level.

Mapping items. The set of regions $\mathcal{I}_{\mathrm{R}}$ is used as an intermediate layer to map items from $\mathcal{I}_{\mathrm{S}}$ and $\mathcal{I}_{\mathrm{T}}$: Each item is associated with the region where it is located. User interactions from both domains can then be converted to interactions with regions. Using mapped data from the source domain D_{S}, i.e., mapped check-in data, we can learn preferences for users of $\mathcal{U}_{\mathrm{S}}$ to the various regions.

6.3.2 Mapping Users from Both Domains

Mapping items from both domains is the first step towards the mapping of users. In fact, check-ins and bookings can be converted to interactions with regions due to the association established between items from $\mathcal{I}_{\mathrm{S}}$ and $\mathcal{I}_{\mathrm{T}}$ with regions from $\mathcal{I}_{\mathrm{R}}$. The goal is to exploit preferences of users in $\mathcal{U}_{\mathrm{S}}$ to regions in $\mathcal{I}_{\mathrm{R}}$ to enrich the modeling of preferences of users in $\mathcal{U}_{\mathrm{T}}$.

For each user in $\mathcal{U}_{\mathrm{T}}$, we compute its nearest neighbors, i.e., most similar users, from the set $\mathcal{U}_{\mathrm{S}}$ with respect to the visited regions. A similarity measure is thus used and handles user profiles from both domains. Each profile is defined as a binary vector which dimension is equal to the cardinality of $\mathcal{I}_{\mathrm{R}}$. If the user visited a POI or a hotel located in the region, the value in the vector is set to 1, otherwise, it is set to 0. Region scores for users in the target domain, $\mathcal{U}_{\mathrm{T}}$, are then estimated by aggregating the region scores computed for each nearest neighbor.

6.3.3 Merging Preferences from Both Domains

Performing hotel recommendation for a target user $u \in \mathcal{U}_{\mathrm{T}}$ requires computing hotels' scores, denoted by $\hat{r}_{ui}$, for each hotel $i \in \mathcal{I}_{\mathrm{T}}$. We assume that the final preference for a hotel can be decomposed into a preference for the region where it is located and a preference for the hotel itself. Therefore, the score $\hat{r}_{ui}$ is the combination of two scores: one from the source domain, denoted by $\hat{r}^{\mathrm{S}}_{ur}$, which reveals the user preference for the region $r \in \mathcal{I}_{\mathrm{R}}$, and the other from the target domain, denoted by $\hat{r}^{\mathrm{T}}_{ui}$, which reveals the user preference for the hotel $i \in \mathcal{I}_{\mathrm{T}}$.

In the source domain, we build a recommendation model modeling the preferences of users in $\mathcal{U}_{\mathrm{S}}$ to regions in $\mathcal{I}_{\mathrm{R}}$ and enabling the computation of scores of regions $r \in \mathcal{I}_{\mathrm{R}}$ for each user $z \in \mathcal{U}_{\mathrm{S}}$, i.e., $\hat{r}^{\mathrm{S}}_{zr}$. In the target domain, we build a recommendation model modeling the preferences of users in $\mathcal{U}_{\mathrm{T}}$ to hotels in $\mathcal{I}_{\mathrm{T}}$ and enabling the computation of scores of hotels $i \in \mathcal{I}_{\mathrm{T}}$ for each user $u \in \mathcal{U}_{\mathrm{T}}$, i.e., $\hat{r}^{\mathrm{T}}_{ui}$.

Final recommendations are performed for users from $\mathcal{U}_{\mathrm{T}}$. The score revealing the region preference for a user $u \in \mathcal{U}_{\mathrm{T}}$ is the aggregation of scores for its nearest neighbors from $\mathcal{U}_{\mathrm{S}}$. The score revealing the hotel preference for a user in $\mathcal{U}_{\mathrm{T}}$, $\hat{r}^{\mathrm{T}}_{ui}$, is directly computed using the

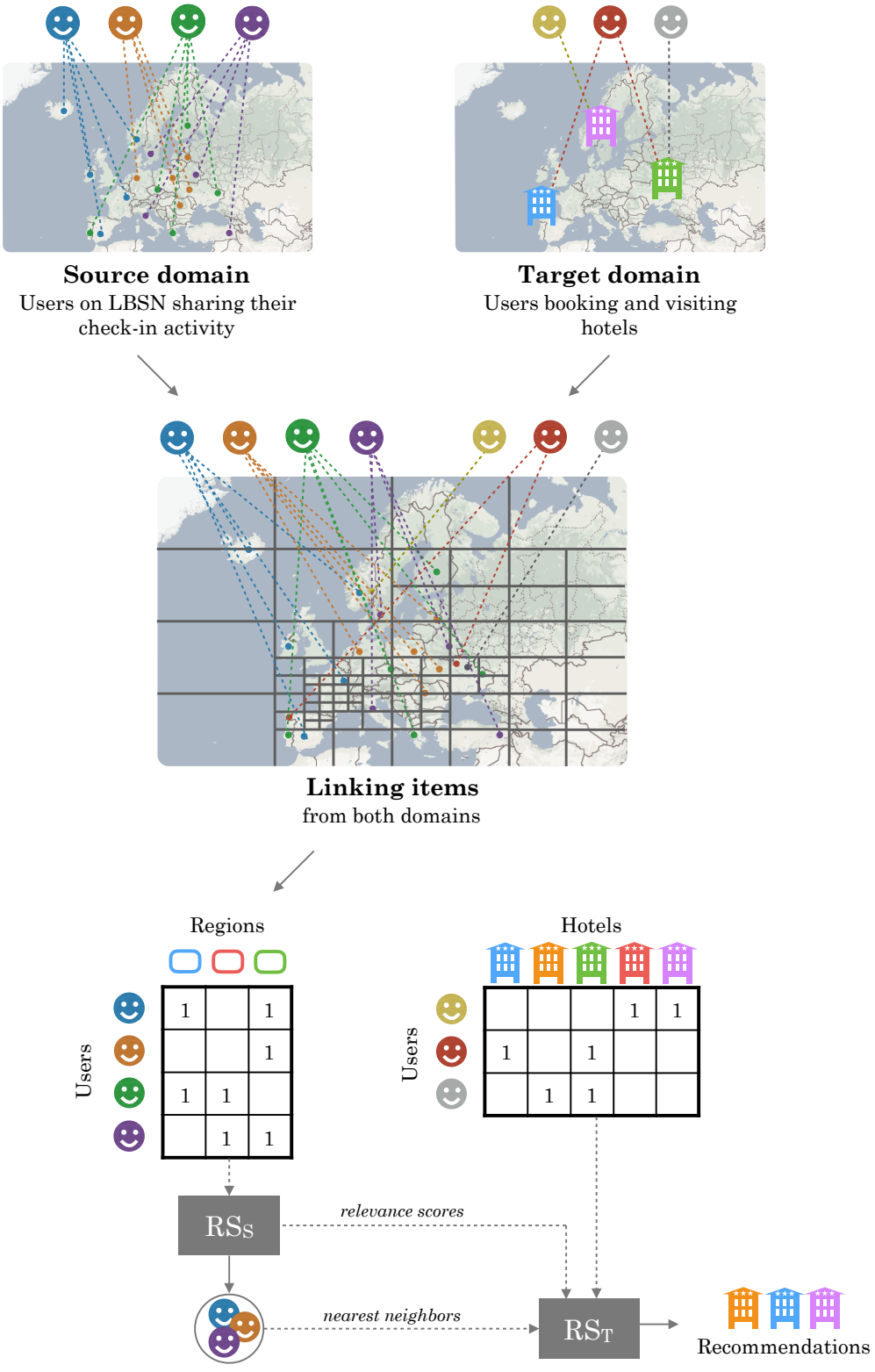


FIGURE 6.2: Proposed approach for cross-domain recommendation considering the LBSN and the hotel domains

model built. Both scores are combined and the final score for hotel i located in region r is given as follows, having a predefined weight parameter α and the number of neighbors Z_u:

$$\hat{r}_{ui} = \alpha.\hat{r}^{\mathrm{T}}_{ui} + (1-\alpha).\frac{\sum_{z \in Z_u} \hat{r}^{\mathrm{S}}_{zr}}{|Z_u|} \tag{6.1}$$

In our work, we use a Matrix Factorization (MF) method, Bayesian Personalized Ranking (BPR) [Rendle et al., 2009] (Section 2.5.2.4) to learn preferences and compute scores, since it performs well on our dataset of bookings. We note that any other recommendation method could have been used within the same cross-domain approach.

6.4 Experimental Results

In this section, we present the experiments we conducted to evaluate the benefits of using data from LBSN for hotel recommendation and to prove the interest of our approach.

Datasets. We used one dataset from each domain to test our approach. The hotel booking dataset, Ah, is extracted from the hotel industry and is presented in Section 3.2. It consists of 7.8M users, 4.5k hotels, and 34M bookings (Table 3.2). Considering the geography dimension, users come from all countries and hotels are spread in more than 90 of them. Concerning the check-in data, we rely on the Yahoo Flickr Creative Commons (YFCC) dataset [Thomee et al., 2016] which is a real-world dataset that was published recently. The original dataset gathers 100M media objects including photos and videos that have been uploaded to Flickr [3] between 2004 and 2014. Only a subset of these posts are annotated with geographic coordinates and can be handled as check-ins. Based on these posts, we consider users that have visited more than 5 regions from the one we define in Section 6.3.1 in order to have enough information to enrich the hotel domain. The filtered dataset we use, denoted by Yfcc, contains around 24M check-ins done by 32k users.

Experimental setup. We split the dataset Ah into a training and a test set. We sort the bookings of each user in a chronological order and select the first 80% to constitute the training set and the rest to constitute the test set. We also select 20% of the users who have only done one booking and add them to the test set in order to evaluate the performance on new users. We use the data from the training set of Ah in addition to the dataset Yfcc to train our recommendation method and evaluate its performance on the test set of Ah.

Evaluation metrics. We consider that we recommend N hotels to each user and note which of these hotels were actually visited based on the set of held-out bookings. We use recall@N and NDCG@N for measuring the performance (Section 2.2.3). Since large values of N are not interesting for top-N recommendation, we set N to 5 and 10 for both metrics.

Methods compared. We include in our comparison traditional recommendation methods that are listed in the following:

[3] http://www.flickr.com

- `MostPop` recommends the most popular hotels with respect to the user's profile. The user profile includes a set of features related to the gender, the age category, and the country of residence. `MostPop` recommends to a target user the most popular hotels chosen by those having the same or similar profiles.
- `CB` is a content-based method where hotels and users are represented in the space of hotels' features using vector space models and TF-IDF weighting [Pazzani and Billsus, 2007] (Section 2.4.1). Hotel features cover the location, the brand, the segment category, and offered services such as Wi-Fi connection, parking, meeting facilities, and children playground.
- `Knnu` is a user-based neighborhood method (Section 2.5.1.1) where we use the Jaccard similarity metric to measure similarities between users and set the number of neighbors to 2000.
- `WRMF` is a MF technique handling implicit feedback [Hu et al., 2008] (Section 2.5.2.4). We set the number of latent factors to 100, the regularization parameters to 0.001, and $c_{ui} = 1.0$ for positive observations and $c_{ui} = 0.01$ for negative observations.
- `BPR` is a MF technique that relies on pairwise preferences to learn the latent model [Rendle et al., 2009] (Section 2.5.2.4). We set the number of factors to 100 and the regularization parameters to 0.0025.
- `CD` is the method we propose in this chapter, leveraging data from LBSN for hotel recommendation. The maximum number of levels in the hierarchy (Section 6.3.1), L, is set to 14. We use the Jaccard similarity metric to measure similarities between users (Section 2.5.1.1) and set the number of neighbors to 500 and α to 0.7 — following a grid search process.

Results. Figure 6.3 shows the performance of the methods we consider. The results are represented for each category of users, a category being defined by the number of bookings included in the training set. By definition, the metrics we measure decrease when the number of bookings increases. `MostPop` is the only method able to recommend hotels to inactive users, i.e., users with zero bookings in the training set. The inferiority of `CB` shows that users do not attribute a great importance to all the hotels' features considered. Further investigations showed that the location of the hotel is one of the few factors that greatly affect the decision. `Knnu` performs well for users with few bookings while `BPR` outperforms the other methods when the number of bookings increases significantly.

The results obtained for `CD` show the interest of using data from LBSN to alleviate the sparsity problem. `CD` outperforms all the other methods when the number of bookings is less than or equal to 10 bookings. The interest of using cross-domain information decreases when the number of bookings increases: `BPR` outperforms `CD` when the number of bookings is greater than 30. One explanation may be due to the fact that the behavior of users actively sharing content on LBSN is not fully representative of the behavior of all travelers. In particular, people having done more than 30 bookings are more likely to be businesspeople which behavior is not necessarily similar to users from LBSN. In addition, once enough feedback about hotels is collected, it may be sufficient to rely on hotel preferences to generate appropriate recommendations. The interest of using `CD` is highlighted in the cold-start

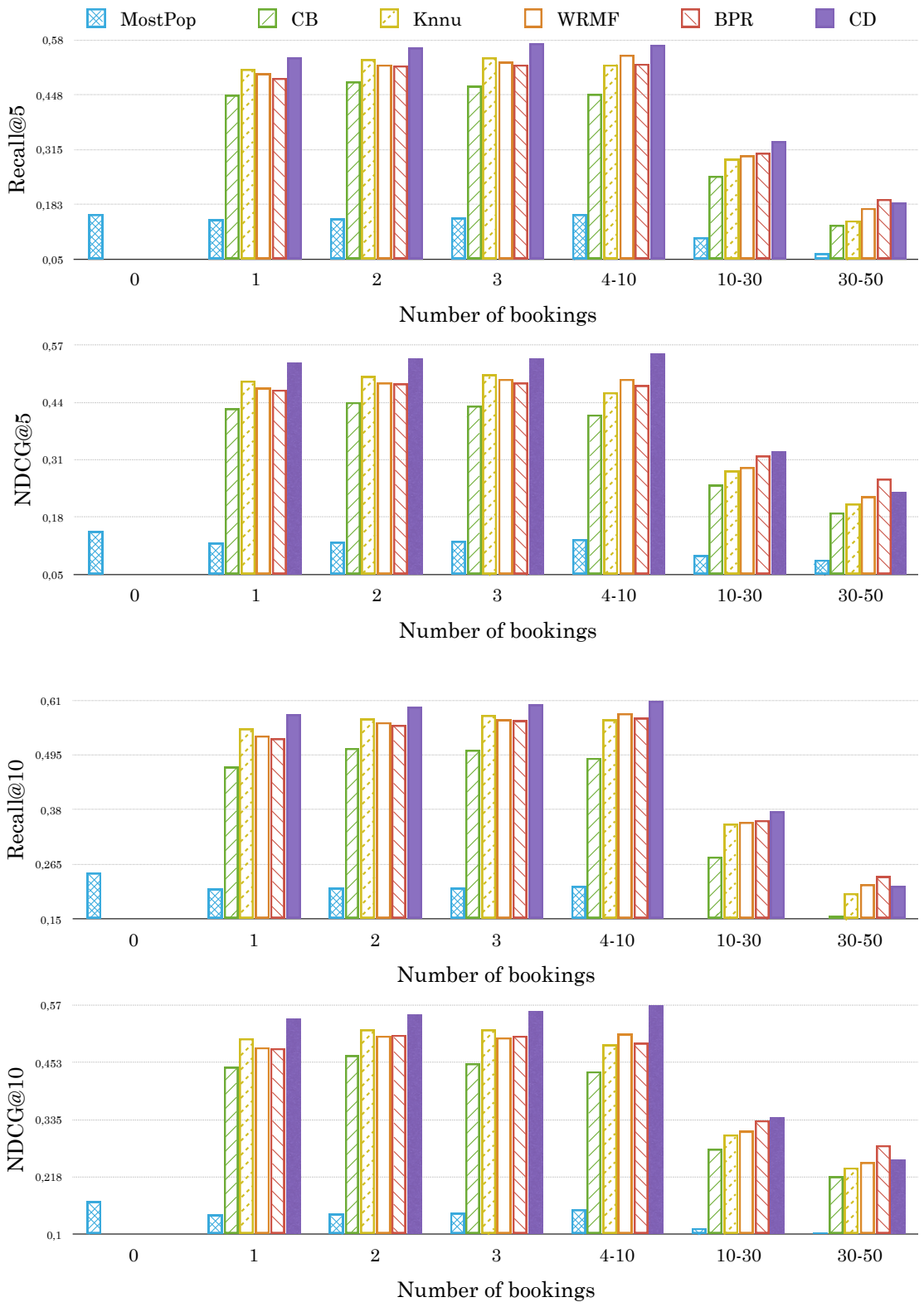


FIGURE 6.3: Recall@5, NDCG@5, recall@10, and NDCG@10 for the dataset AH, represented with respect to the number of bookings present in the training set

setting where hotel bookings alone are not enough to infer preferences. We note that the majority of users in the booking dataset have done less than 10 bookings and, therefore, `CD` improves the overall performance.

Discussion. This is the first work proposing to apply cross-domain recommendation in the hotel sector using in particular abundantly available data from LBSN. While it can be a promising approach especially in a sparse data environment, it opens several interesting challenges.

First, not all the users have their mobility behavior represented by active users posting on LBSN. These users will not directly benefit from the proposed approach. A finer analysis of users posting on LBSN, i.e., users in the source domain, and users booking hotels, i.e., users in the target domain, may help to identify relevant user segments which behavior can be similar in both domains in terms of mobility and probably underrepresented in one or in both domains. As a first basic approach, we tried addressing this issue by defining segments based on the number of bookings made, considering that this number reveals a certain aspect of the user category. Other alternatives including more advanced techniques may be applied. One possibility to benefit from cross-domain recommendation may be to learn local models per user category.

Second, further advances in this direction should consider evaluating the recommendation diversity in terms of proposed locations. It is important to generate diverse recommendations and avoid suggesting hotels located in the same area. This may occur when one region gathering several hotels is specifically promoted for a particular user.

Transferring knowledge from LBSN to the hotel sector may go beyond the mobility aspect by also considering temporality, i.e., periods during which one region is visited by a segment of users, and context of visits by analyzing meta-data associated to the posts. In addition, while we used a clustering component to map both domains, other approaches to integrating knowledge may be exploited. We may be considering to rely on a multi-task approach and to train models simultaneously in both domains.

6.5 Conclusion

In this chapter, we propose to use data from LBSN to boost hotel recommendation and to address the problem of sparsity due to the scarce feedback generated by the traveling activity. Hotel selection largely depends on the visited destination and some destinations are more accessible to users than others. Using the check-in activity from LBSN, we learn preferences for regions and use these preferences for hotel recommendation. Mapping of items from both domains is done through a space of regions which definition is based on the density of hotels. Mapping of users from both domains is done by computing similarities between users based on the visited locations. Hotel recommendation accounts for region preferences and hotel preferences. Experiments show the interest of using cross-domain information for users with few observations, i.e., in the cold-start setting.

Temporality plays an important role in the decision-making process: One destination is not considered by the same user in all periods of the year. Future work will involve adding

the time dimension and taking into account in which period of the year the check-in was made in order to distinguish between users visiting the same destinations at different periods or seasons.

While Part II of the thesis was meant to address the challenges of *partially observable* context in hotel recommendation, we focus in Part III on the implications of the presence of *unobservable* context in online recommendation.

Part III

From Unobservable Context to Online Adaptive Recommendation

Chapter 7

The Online Adaptive Recommendation Problem

This chapter presents the problem of online adaptive recommendation which is studied in Part III of this thesis. While real-world settings require the development of online RS, such systems face several challenges. They are expected, first, to continuously learn from data streams and, then, to adapt to changes in real-time. In particular, users and items rely on some *unobservable* context that is hidden from the system, and any drift in the hidden context affects the represented concepts. Recommendation models have to actively account for these drifts in order to guarantee a good performance, introducing the problem of online adaptive recommendation. We provide a clear formulation of the problem and discuss related work in both fields of RS and data stream mining, highlighting the shortcomings of existing approaches.

7.1 Introduction

While RS apply techniques from various domains such as information retrieval and human computer interaction, the system's core can be viewed as an instance of a data mining process [Amatriain and Pujol, 2015]: Data related to users and items is analyzed, a model is inferred, and recommendations are produced for each user. The model is trained with all available data and the system is then queried for recommendations. While RS are deployed online where data is continuously generated, recommendation models ensuring a good recommendation quality tend to suffer from very high latency. Training on large datasets of user interactions is computationally expensive in terms of space and time and cannot be performed each time a new observation is received. In practice, most RS build first a model from a large static dataset and then rebuild it periodically as new chunks of data arrive and are added to the original dataset.

This functioning in batch faces in particular two limitations. First, the RS cannot take into account the user feedback generated after a model update before the next one and cannot thus consider the continuous activity of users. While user preferences and item

perceptions are dynamic concepts that are constantly changing over time, the same model is used to deliver recommendations and is indifferent to these temporal dynamics. The inability to adapt to quick changes by considering recent feedback leads to poor recommendation quality and user experience [Dias et al., 2008]. Second, training a model on a continuously growing dataset is computationally expensive. This raises scalability issues in addition to the need for additional resources to store and process data. Solution options include scaling up systems which can be expensive, reducing the number of updates which can worsen recommendations, and decreasing the volume of data considered which may result in discarding valuable information for recommendation.

Advances in the RS field have focused on a part of these limitations and attempted to address each one of them using different approaches. Time-Aware RS (TARS) [Campos et al., 2014] were introduced to handle the dynamic aspect of entities modeled by RS. They respect the chronological order of observations and try to capture the evolution of users and items over time. While these systems consider the temporal dimension, they are meant to work in batch and cannot technically integrate new user feedback as soon as it is generated. Some of the computational issues were addressed by considering efficient and distributed models [Gemulla et al., 2011; Teflioudi et al., 2012] but continuously growing datasets still needed to be stored and processed.

Recent work [Siddiqui et al., 2014; Vinagre et al., 2014b] has proposed to address the recommendation problem as a data stream problem by designing *online RS*, resulting in a more realistic setting. Elements of a data stream are expected to arrive indefinitely in real-time at high uncontrolled rates. A parallel can thus be drawn with user-generated data handled by RS that share actually the same characteristics. Online RS are expected to learn from continuous data streams and to maintain recommendation models up to date, and rely on incremental learning. By adapting to changes in real-time and operating independently of the number of observations, they address the previously mentioned limitations faced by batch RS.

One of the core problems in data stream mining consists in dealing with concept drifts which occur when the definition of modeled concepts change over time [Gama et al., 2014]. Online RS have to be able to track multiple concepts changing in different ways at different moments and rates, including the preferences of each user and the perception of each item. Incremental learning is a natural way to account for a specific type of drift, i.e., incremental drift, where the concept slowly passes by intermediate states until reaching a new one. However, specific strategies and techniques need to be adopted to detect and adapt to other types of drifts such as more abrupt ones. While previous work has already proposed to design online RS, limited effort has been made to develop online adaptive RS that automatically account for drifts in real-time.

In this chapter, we present the problem of online adaptive recommendation and discuss related work in the recommendation and the data stream mining fields. Taking into account the evolution of concepts in RS can be traced back to TARS [Campos et al., 2014] with the limitation of functioning in batch and not being able to maintain models up to date. Online RS were then proposed to cope with this issue. However, drift detection techniques still needed to be integrated to ensure a strong adaptive ability facing changes, which is the main idea explored in Part III of this thesis. Overall, online adaptive recommendation is

related to TARS due to the dynamic aspect of users and items and to data stream mining due to the analysis of data streams in real-time.

The chapter is structured as follows. In Section 7.2, we discuss related work on TARS. Section 7.3 presents the problem of concept adaptation in data stream mining. In Section 7.4, we present the problem of online adaptive recommendation in addition to related work tackling different aspects of it. Finally, a summary is given in Section 7.5.

7.2 Time Dimension in Recommender Systems

Motivation. When learning user and item models based on a dataset of user interactions, traditional recommendation approaches assume that the user behavior follows a single specific pattern that is expected to reappear in the future, making prediction a feasible task. This assumption does not hold in a dynamic world where users and items are constantly evolving. Efficient RS are expected to maintain models that reflect the actual state of entities [Koren, 2009]. On the other hand, modeling temporal changes related to users and items is a challenging task. First, changes may occur on a global scale: The introduction of new items may affect the behavior of users, and seasonal patterns, holidays, and unexpected events may also influence the attractiveness of items. Many of the changes are also driven by local factors and hidden contexts related to users. A change in the user social status can result in a change of behavior and user interests tend to shift with time. While tracking all of these changes is essential to guarantee the generation of appropriate recommendations, changes happen in different ways at different moments and rates for each user and item. Therefore, simple solutions that assume having *a priori* knowledge about the way concepts are changing, e.g., solutions that discard old observations, are not sufficient. Instead, more advanced techniques that leverage signals extracted from past and recent observations need to be considered.

Historical perspective. The practical value of modeling temporal changes in RS was first highlighted within the context of the Netflix Prize (Section 2.1.1). [Koren, 2009] showed that user ratings exhibit temporal patterns that could be exploited for the recommendation task. The proposed solution allowed the distinction between long-term patterns and temporary ones and was evaluated on the dataset of the competition. Including temporal changes proved to be very useful to improve the recommendation quality and several approaches were further proposed, forming the category of RS known as *Time-Aware RS (TARS)* [Campos et al., 2014]. Then, the rise of *Context-Aware RS (CARS)* lead to investigating the possibility of integrating time as a contextual dimension. One of the first efforts following this direction [Baltrunas and Amatriain, 2009] proposed to build contextual micro-profiles representing the user behavior in different temporal contexts, e.g., morning and evening or weekend and workday. This technique was then identified as being part of the pre-filtering paradigm for CARS (Section 2.7.1). On the other hand, recent years have witnessed the emergence of a related problem which is the problem of *sequence-aware recommendation* [Quadrana et al., 2018]. The increased interest was mainly fueled by industry through the release of a new dataset of session activity from a major e-commerce platform within the context of the 2015 ACM RecSys Challenge [Ben-Shimon et al., 2015], and by the development of new deep learning techniques for sequence learning [Hidasi et al.,

2016]. Sequence-aware algorithms learn sequential patterns from user behavior and attempt to predict the user's next action within a session or to detect short-term trends. TARS and sequence-aware RS share a number of common characteristics. However, while they both rely on the order of interactions, sequence-aware RS do not focus on the exact point in time of past observations and try to detect patterns within the sequence itself. Approaching data as a chronologically ordered sequence also gave rise to online RS that handle data as a stream [Siddiqui et al., 2014; Vinagre et al., 2014b]. These RS are based on incremental learning which is a natural way to account for relatively small changes and to keep models adapted to the actual reality.

Time-Aware Recommender Systems (TARS). Several methods were proposed to integrate the temporal dimension into RS. We can mainly distinguish between three categories of approaches merging time with Collaborative Filtering (CF) methods: *recency-based time-aware* models, *contextual time-aware* models, and *continuous time-aware* models. Other categorizations can be found in [Campos et al., 2014; Vinagre et al., 2015b].

Recency-based time-aware models. Recency-based models are based on the assumption that recent observations are more relevant than older ones since they are more representative of the actual reality: They should therefore be given more importance in the learning and recommendation processes. These methods have the advantage of being simple and easy to integrate into existing recommendation models. However, the main assumption established does not always hold in real-world scenarios. In particular, it is not uniformly valid for all users and items, resulting in a limited performance. The recency aspect can be tackled with decay-based [Ding and Li, 2005; Liu et al., 2010c] or window-based [Lathia et al., 2009; Nasraoui et al., 2007] methods. Decay-based methods gradually decrease the importance and contribution of past data while window-based methods only consider observations contained in a window of a certain length. We note that window-based methods can be viewed as a particular case of decay-based methods with a binary decay function.

Contextual time-aware models. Contextual approaches integrate the time dimension by considering temporal information as context. Several contextual features can be extracted from the temporal information such as the time of day, the day of the week, the month, and the season. Once the contextual features are defined and specified, any context-aware approach can be used to address the problem (Section 2.7.1).

The various paradigms existing for CARS were used for TARS. [Baltrunas and Amatriain, 2009] proposed a pre-filtering method where they built contextual micro-profiles representing the user behavior within different time frames and used them for recommendation. Post-filtering was leveraged in [Panniello et al., 2009] for TARS: Contextual relevance weights based on temporal features were computed and used to filter recommendations. In addition, factorization methods relying on Tensor Factorization (TnF) were applied to address this problem [Gantner et al., 2010b; Liu et al., 2010b].

Continuous time-aware models. Continuous models represent ratings and user interactions as a function of time, and the model parameters are learned from the data. Some of these models rely on *time series models* and others on *factorization models*.

In the first set of models, user ratings and interactions are encoded as time series and time-series techniques are used to predict the current user interest. These techniques have

been applied when dealing with explicit feedback such as ratings in [Cao et al., 2009] and implicit feedback such as web logs in [Zimdars et al., 2001]. [Jahrer et al., 2010] proposed to build several CF models for different time periods and used blending techniques to combine the models' recommendations.

On the other hand, [Koren, 2009] was one of the first works proposing to exploit factorization models for time-aware recommendations. The proposed solution relies on SVD++ [Koren, 2008] that considers user and item biases, i.e., deviations from a global average, in addition to a factor model capturing user preferences and item perceptions. Temporal user biases are modeled using a decay function, different item biases are considered for each time window, and an additional decay function is used to weigh previous ratings for prediction. The model gained popularity due to the accuracy improvements obtained on the Netflix Prize dataset. A similar model was then presented for the task of music recommendation [Koenigstein et al., 2011]. Another way to integrate time into the Matrix Factorization (MF) framework was proposed in [Karatzoglou, 2011] where differentiated item factors are learned based on the ratings' timestamps. We mention that several approaches also considered modeling users' short-term and long-term preferences separately [Liu and Aberer, 2014; Xiang et al., 2010].

More recently, modeling the evolution of user preferences and item perceptions was considered on a more refined individual level. Instead of relying on the information provided by other users, specific approaches were developed to build personalized models and profile the evolution of individual users. [Liu, 2015] handled implicit feedback and proposed to extract topics from each user interaction observed at a certain time point. The evolution of each topic for each user is then treated as a time series to predict the user's future preference. In [Lu et al., 2016], user interests and their evolutions are learned simultaneously and collaboratively at each time point by combining MF and vector autoregression. [Gao et al., 2017] built on this idea but considered in addition the evolution of items' contents.

Relation to our work. We review in this section the main approaches proposed for TARS while omitting further details, given that all of them share one main limitation. While considering the evolution of modeled concepts over time, they are not adapted to the online setting and cannot maintain models up to date in real-time. TARS assume that the whole dataset is available at the time of training and that multiple passes can be performed on the data. We also do not review previous work on sequence-aware RS even though it is related in some way, given that we do not consider applications where relevant patterns within sequences are expected to exist. Moving to a more realistic setting, we investigate the problem of learning from a stream of user interactions with the challenge of adapting to changes in real-time. We present in the next section the more general problem of adaptive stream mining before discussing the problem of online recommendation.

7.3 Adaptive Data Stream Mining

Data stream mining. In traditional data mining, models are built using historical data in an *offline* mode: The whole dataset is available for training and the model is used for prediction once the training is completed. With the massive and growing generation of

information and signals, data is commonly available as continuous streams that can only be appropriately processed in an *online* mode. This is due to the fact that elements of a data stream are expected to arrive in real-time at high rates and to be processed one after the other in only a few passes using limited time and memory per element. In an online learning setting, data is processed sequentially and models are continuously trained when receiving new observations. Aside from the online nature of learning, handling data streams presents the challenge of detecting and adapting to concept drifts. Concept drift occurs in non-stationary environments when the data distribution changes over time and imposes the design of adaptive models.

Concept drift. The main assumption made by most predictive models is that the training data, i.e., observations used for learning, and the testing data, i.e., observations for which the model will be generating predictions, are sampled from the same probability distribution. This assumption does not hold in real-world domains where predictive models are meant to be developed and applied, due to the evolution of target concepts over time. Learning is often performed in non-stationary environments and, thus, requires adaptive models that are able to account for changes occurring unexpectedly over time. The difficulty in handling such changes is initially derived from the fact that the target concept that is being represented depends on some hidden context or on an underlying process that is unknown to the learner [Tsymbal, 2004]. Any change in the hidden context can affect the target concept and thus the performance of the learner.

The process of learning in non-stationary environments is formalized for the problem of supervised learning, resulting in a probabilistic definition of concept drift detailed in the following. In a supervised learning setting, the goal is to predict a target variable $y \in \mathbb{R}$ or the class y given a set of input features $\mathbf{X} \in \mathbb{R}^n$. Observations that are used for learning are then denoted by $(\mathbf{X}, y)$ and predictions are performed for new observations where $\mathbf{X}$ is known and y is unknown. Observations generated at time t are assumed to be sampled from the joint probability distribution $p_t(\mathbf{X}, y)$, and we denote by $p_t(y|\mathbf{X})$ the posterior distribution of class y and by $p_t(\mathbf{X})$ the evidence distribution. Concepts are expected to evolve over time and consequently distributions dynamically change. A concept drift occurring between time points t_0 and t_1 is defined as [Ditzler et al., 2015; Gama et al., 2014]:

$$\exists \mathbf{X} \ : \ p_{t_0}(\mathbf{X}, y) \neq p_{t_1}(\mathbf{X}, y) \tag{7.1}$$

Given that different natures of changes may be encountered, a terminology is adopted to designate where the change originates from in addition to its form. Concerning the distribution from which the change is originating, we distinguish two types of drifts. *Real concept drifts* occur in cases where the posterior probability $p_t(y|\mathbf{X})$ changes over time which may happen independently from changes in the evidence $p_t(\mathbf{X})$. *Virtual concept drifts* occur in cases where the evidence probability $p_t(\mathbf{X})$ changes without affecting $p_t(y|\mathbf{X})$. In other words, virtual drifts happen when the distribution of received observations changes. In both cases, the model needs to be revisited given that the error generated may no longer be tolerated.

On the other hand, concept drifts take different concrete forms. These multiple forms are illustrated in Figure 7.1, adapted from [Gama et al., 2014], where changes in the data mean are tracked. The concept drift can happen either *abruptly* resulting in a sudden

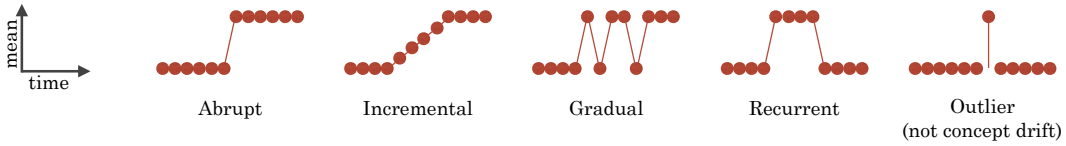


FIGURE 7.1: Concept drift forms represented over time. Figure adapted from [Gama et al., 2014].

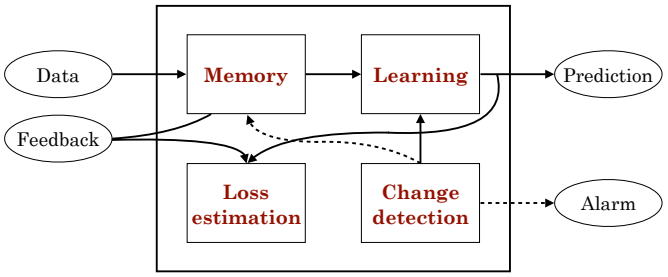


FIGURE 7.2: A generic framework for online adaptive learning. Figure adapted from [Gama et al., 2014].

change, *incrementally* consisting in the existence of intermediate states between the old concept and the new one, or *gradually* where the change is not abrupt but we keep going back to the previous concept for some period of time. Drifts can further be classified into *permanent* drifts where the effect of the change is not bounded in time and *transient* drifts which effect disappears after a certain period of time. While new concepts may be introduced at each change, previously observed concepts may reoccur resulting in *recurrent* drifts. Such settings benefit from the ability to retrieve previous knowledge related to similar observed concepts. It is worth mentioning that concept drifts should not be mixed with outliers referring to anomalies that may appear in the data and demanding no adaptivity of the models. Operating under the presence of concept drifts requires specific methods and techniques able to detect and adapt to changes.

Online adaptive learning. The online adaptive procedure consists of the following three steps, applied for each received observation: (i) *Predict.* A prediction $\hat{y}_t$ is made for the received observation $\mathbf{X}_t$ using the current model, (ii) *Evaluate.* After receiving the true label y_t, we estimate the loss $f(\hat{y}_t, y_t)$, (iii) *Update.* The observation $(\mathbf{X}_t, y_t)$ may be used to update the model.

[Gama et al., 2014] proposed a generic framework for online adaptive learning represented in Figure 7.2. It consists of four modules: memory, learning, change detection, and loss estimation, that we briefly review in the following.

Memory module. The memory module defines the strategies followed to select data for learning on one hand and to discard old data on the other hand. Adaptive learning approaches assume that recent data is the most relevant to the current prediction task. Learning is thus done based on the most recent data and can be performed using a single or multiple observations. Learning from a single observation requires storing only the corresponding observation. This is the natural setting in online learning where models are continuously updated using the received observations with no access to older ones. Adaptation to new concepts is expected to happen naturally as more learning is performed and as concepts learned from older data are gradually disappearing. Learning from multiple observations requires storing the most recent observations. Sliding windows of fixed or variable sizes are usually used to store observations which are then used to build new predictive models [Widmer and Kubat, 1996]. Discarding outdated data is a way of handling evolving streams and is achieved through forgetting mechanisms which choice depends on *a priori* knowledge related to the nature of the change. *Abrupt forgetting* may be adopted by totally discarding observations based on their relevance. This can be done using sliding windows or sampling techniques [Vitter, 1985]. A less extreme form of forgetting, i.e., *gradual forgetting*, can be performed by using all observations stored in the memory for training but weighting each one based on its age [Klinkenberg, 2004].

Learning module. The learning module defines how to build predictive models and maintain them up to date given the received observations. Predictive models can be retrained from scratch using observations stored in the memory after discarding old models. They can also be incrementally updated using received data. In *incremental learning*, observations are processed one after the other. Existing models are updated after each received observation and may access older observations. In the more restrictive setting of *online learning*, data is processed once in a sequential fashion [Littlestone, 1988]. The *streaming* setting adds limitations on the memory and time resources required for processing data given its high rate. As learning is carried on, concepts learned from recent data tend to erase those previously learned. Continuously updating models over time is a way to *passively* adapt to changes. Passive approaches face the limitation of being slow in reaction to changes: Learning is done at a constant speed independently from the changes occurring in the environment. Therefore, *active* approaches have been proposed where drifts are explicitly detected triggering a learning mechanism to adapt the models [Elwell and Polikar, 2011]. Drift detection is handled by the *Change detection* module, and once a drift is reported, it is possible to rebuild a model from scratch or adapt the part concerned with the change. Along this line, ensemble learning has been leveraged for adaptive stream mining [Gomes et al., 2017].

Change detection module. The change detection module handles techniques for the active detection of drifts. According to [Gama et al., 2014], these techniques can be classified into several categories and we mention some of them in the following. First, detectors based on sequential analysis include for example the cumulative sum. It analyzes the performance of a predictor and reports a change when the mean of the analyzed data significantly deviates from zero [Alippi and Roveri, 2006]. Then, other methods consist in monitoring two distributions. They rely on a fixed reference window representing the past observations and a sliding one containing recent observations. Distributions over the two windows are compared using statistical tests. Given that the null hypothesis states that distributions are equal, a change is detected when it is rejected. An example of these methods is ADaptive

WINdowing (ADWIN) [Bifet and Gavalda, 2007]. Additional solutions leverage statistical process control or well-defined heuristics.

Loss estimation module. The loss estimation module analyzes the performance of the learning algorithm and transfers information to the change detection module. The loss variable can be defined with respect to the model, i.e., using properties tightly related to the predictive model, or independently from the model, e.g., by comparing two sliding windows.

Relation to our work. The vast majority of work related to RS considers learning predictive models in an offline mode. As mentioned at the beginning of the chapter, this setting is not adapted to real-world environments where user-generated data arrives in the form of streams and where concepts are expected to change over time. While previous work has already proposed to learn recommendation models in an incremental or online mode, the area remains underexplored or even unexplored when it comes to actively detecting drifts and designing online adaptive algorithms for recommendation. We introduce the formulation of online adaptive recommendation relying on the framework developed for stream mining and presented in this section. We review previous work based on this formulation.

7.4 Online Adaptive Recommendation

As mentioned in previous sections, real-world settings require to formulate the recommendation problem as a data stream problem: User-generated data is continuously received and analyzed and recommendations are delivered in real-time. Online recommenders can thus rely on the generic framework presented in Section 7.3, Figure 7.2 [Gama et al., 2014]. In addition to the modules included in the framework, we consider the following two modules for the problem of online adaptive recommendation: the *retrieval* module that is responsible for the efficient computation and retrieval of recommendations and the *evaluation* module that constantly evaluates and reports the RS performance. Previous work related to online recommendation attempted to come up with solutions regarding one or several modules at once. In the following, we propose to review related work according to the module concerned by the solution. We note that two modules are not mentioned: the *change detection* module since it was not specifically addressed for online RS and the *loss estimation* module since existing solutions are included in other modules.

7.4.1 Memory Module

Assuming that recent data is more relevant than older one, predictive models have to learn from recent observations and forget old ones. This has been applied in the context of online recommendation in several ways.

Forgetting in memory-based approaches. [Nasraoui et al., 2007] use a user-based neighborhood approach that relies on a sliding window containing a fixed number of recent observations. Along this line, [Siddiqui et al., 2014] propose a recommendation algorithm that operates in two steps. Users are first clustered based on their previous interactions

and a user-based neighborhood approach is then applied in each cluster to perform recommendation. A sliding window is used over the stream: Only observations done during a certain time interval are considered when computing similarities and scores, and the others are definitely discarded. Aside from sliding windows, fading factors have also been used to gradually forget old information. [Vinagre and Jorge, 2012] propose to multiply user similarities by a positive scalar $\alpha < 1$ before updating them with information received from new observations. The value of α controls the rate at which forgetting is performed.

Forgetting in Matrix Factorization (MF) approaches. [Matuszyk et al., 2015] introduce two sets of forgetting strategies within the scope of MF: *rating-based* forgetting and *latent factor-based* forgetting. Rating-based strategies discard past ratings for each user either using sliding windows, e.g., fixed size or covering a fixed time period, or using sensitivity analysis, e.g., forget ratings that imply abnormal changes in the user profile. On the other hand, latent factor-based strategies readjust the latent factors of MF to reduce the effect of old observations. These strategies include forgetting unpopular or popular items and fading user factors.

Most online RS that only contribute on the level of the memory module face one main limitation. They require fixing a set of parameters, e.g., the size of a sliding window or a fading factor, and anticipate that the RS should forget old observations. Therefore, they assume we have *a priori* knowledge regarding the way the user behavior is changing and that preferences of all users drift at the same rate, which is obviously not the case.

7.4.2 Learning Module

The learning module defines how recommendation models are learned and maintained up to date which is usually based on incremental learning. In fact, traditional CF methods, i.e., memory-based and model-based approaches, have been adapted to the incremental setting and some of them are presented in the following.

7.4.2.1 Incremental Memory-Based Approaches

Applying memory-based algorithms in a streaming setting faces two major issues. First, similarity computation is performed in a costly offline phase where similarities between all user pairs or item pairs are evaluated given the observed interactions. Second, the whole history of user interactions cannot be stored in memory and used entirely for recommendation. Several solutions were proposed addressing one or both of these limitations.

Incremental neighborhood-based methods were first proposed in [Papagelis et al., 2005]. The authors presented an incremental user-based approach handling explicit feedback in the form of ratings. User similarities are stored and incrementally updated for each received rating. Nearest neighbors and predicted ratings are then computed at the time of recommendation. Similarities are evaluated using Pearson Correlation (Section 2.5.1.1) which is

split in the following way:

$$A = sim_{PC}(u,v),\ B = \sum_{x \in \mathcal{I}_{uv}} (r_{ux} - \overline{r}_u)(r_{vx} - \overline{r}_v),\ C = \sum_{x \in \mathcal{I}_{uv}} (r_{ux} - \overline{r}_u)^2,\ D = \sum_{x \in \mathcal{I}_{uv}} (r_{vx} - \overline{r}_v)^2 \tag{7.2}$$

where

$$A = \frac{B}{\sqrt{C}\sqrt{D}} \tag{7.3}$$

Elements A, B, C, and D are stored and incrementally updated, and new similarities can be computed in a fast manner. Incremental user-based and item-based neighborhood approaches handling implicit feedback and using the cosine similarity were then presented in [Miranda and Jorge, 2009]. The formulation of cosine similarity in an implicit feedback setting is based on user and item occurrence and co-occurrence counts. Two users co-occur for every item they both interact with and two items co-occur for each user that interacts with both of them. Given that counts are stored, maintaining similarities up to date for each received observation boils down to incrementing counts of occurrences and co-occurrences. Similarly to [Papagelis et al., 2005], nearest neighbors and scores are computed at the moment of recommendation. In an attempt to adapt to user and item changes in the scope of a neighborhood-based method, [Liu et al., 2010c] introduced the online evolutionary CF framework. It relies on an incremental version of the item-based neighborhood method and proposes instance weighting techniques to reduce the weight of old observations when computing similarities and scores.

A probabilistic neighborhood-based algorithm based on a min-hash technique was developed in [Subbian et al., 2016]. Similarities between users are approximately computed by tracking the relevant users for each item in a min-hash index which is implemented using hash functions. These hash functions are stored in memory instead of storing the full history of user interactions. Other approaches for incremental CF built on co-clustering methods for recommendation [George and Merugu, 2005; Khoshneshin and Street, 2010]. In its most simple form, [George and Merugu, 2005] proposed incremental and parallel versions of the co-clustering algorithm that simultaneously clusters users and items. Predicted ratings are estimated based on the average rating of co-clusters, users' bias, and items' bias.

On the other hand, efficient frameworks and architectures for online recommendation were presented based on neighborhood-based approaches. StreamRec [Chandramouli et al., 2011] implements a scalable item-based CF approach based on a stream processing system and handles explicit ratings. While also relying on an item-based CF approach, TencentRec [Huang et al., 2015] deals with implicit feedback and is designed to serve recommendations for a wide range of applications having different requirements. It is built on Storm with a data access component and a data storage component, and proposes real-time pruning to reduce the computation cost.

7.4.2.2 Incremental Matrix Factorization and Other Model-Based Approaches

Due to the great success of Matrix Factorization (MF) and model-based approaches in the context of recommendation [Koren et al., 2009], several efforts have been made to adapt them

to the incremental and online settings. These efforts historically followed the evolution and advances of MF approaches. While some of them focused on integrating newly observed users and items into an existing model, the others managed to update parts of learned models based on current observations.

Singular Value Decomposition (SVD). Early effort on Incremental Matrix Factorization (IMF) for RS was introduced in [Sarwar et al., 2002]. Incremental updates of SVD are made through the *fold-in* projection technique: Latent vectors corresponding to new users or new items are computed based on the current decomposition and are then appended to the existing matrices. [Brand, 2003] defined incremental operations allowing the addition, update, and removal of data already incorporated into the SVD model. This is made possible through the basic algebraic properties of SVD. Although these techniques lead to a high efficiency, SVD suffers from several shortcomings such as the need to have an initial matrix with no missing elements, resulting in a limited performance on sparse datasets (Section 2.5.2.2).

Point-wise approaches. With the rise of the MF framework for the problem of rating prediction, [Rendle and Schmidt-Thieme, 2008] introduced regularized kernel MF as a generalization of regularized MF. It provides a flexible method to derive new MF methods by transforming the product of factor matrices using kernel functions. Within this context, an online algorithm was proposed to allow the incremental addition of new users and items to the existing MF model. Concretely, the algorithm retrains the whole feature vector corresponding to the new user or item and keeps the other feature vectors fixed. Along this line, [Takács et al., 2009] proposed BRISMF, standing for Biased Regularized Incremental Simultaneous MF, supporting incremental updates of latent factors given an initially trained model and integrating user and item biases expected to model user and item tendencies with respect to ratings. For each received rating, user features of the corresponding user are retrained considering all the ratings he previously made, and item features are kept fixed to avoid a larger retraining process. While [Rendle and Schmidt-Thieme, 2008] and [Takács et al., 2009] rely on Stochastic Gradient Descent (SGD) to learn latent factors, other IMF approaches exploited variants of the Alternating Least Squares (ALS) method to benefit from its advantages (Section 2.5.2.3). [Yu et al., 2016] proposed one-sided least squares for updating one side, i.e, user-side or item-side, of an existing MF model. It is applied when a new user or a new item is encountered and is also proposed to update both sides of the model when it is required. We note that the complexity of the solution offered by ALS is higher than the one offered by SGD (Section 2.5.2.3).

Aside from the rating prediction problem, [Vinagre et al., 2014b] proposed an IMF framework for the top-N recommendation problem and handling positive-only feedback. When either a user or an item is observed for the first time, it is added to the model with random initialization. User and item latent factors are then incrementally updated following SGD, i.e., using the gradient of the objective function for the corresponding observation. [Vinagre et al., 2014b] allow slight updates of latent factors instead of total retraining like in previously mentioned approaches. The proposed method is detailed in Algorithm 5.

One limitation of the approach proposed in [Vinagre et al., 2014b] is related to the problem of dealing with implicit feedback and concerns the lack of negative observations or the way missing observations should be interpreted (Section 2.5.2.4). In fact, only considering

Algorithm 5 Overview of IMF proposed in [Vinagre et al., 2014b]

Data: stream of observations $\mathcal{D}$
Input: number of factors k, learning rate η, regularization parameters λ_P and λ_Q
Output: $\mathbf{P}$, $\mathbf{Q}$

for (u, i, t) in $\mathcal{D}$ **do**
  if $u \notin Rows(\mathbf{P})$ **then** ▷ new user observed
    $\mathbf{p}_u \leftarrow \texttt{Vector}(\texttt{size} : K)$
    $\mathbf{p}_u \sim \mathcal{N}(0, 0.1)$
  end if
  if $i \notin Rows(\mathbf{Q})$ **then** ▷ new item observed
    $\mathbf{q}_i \leftarrow \texttt{Vector}(\texttt{size} : K)$
    $\mathbf{q}_i \sim \mathcal{N}(0, 0.1)$
  end if
  $e_{ui} \leftarrow \mathbf{p}_u.\mathbf{q}_i^\top - 1$
  $\mathbf{p}_u \leftarrow \mathbf{p}_u - 2\eta(e_{ui}\mathbf{q}_i + \lambda_P \mathbf{p}_u)$
  $\mathbf{q}_i \leftarrow \mathbf{q}_i - 2\eta(e_{ui}\mathbf{p}_u + \lambda_Q \mathbf{q}_i)$
end for

positive feedback for learning the model may result in an accuracy degradation. To deal with this problem, [Vinagre et al., 2015a] extended the work in [Vinagre et al., 2014b] by artificially introducing negative feedback into the stream of observations. For each positive observation causing a model update, negative items are considered for the current user and result in additional updates. Negative items are selected based on the recency of item occurrences: The oldest items appearing in the stream of positive observations are considered as negative feedback. Other methods were also introduced to deal with missing observations in an incremental setting. [Devooght et al., 2015] extended several loss functions, e.g., squared and absolute losses, to take into account an explicit prior on unknown values and derived online learning algorithms to update the model. While this approach assumes that missing entries are equally likely to be negative feedback, [He et al., 2016] introduced a weighting strategy for missing observations based on item popularity. The authors also proposed a new learning algorithm based on ALS and handling variably-weighted missing data, in addition to an incremental strategy supporting online learning.

Pair-wise approaches. Ranking approaches based on pair-wise loss functions were designed for the task of top-N recommendation in the presence of implicit feedback (Section 2.5.2.4). In the scope of online recommendation, [Diaz-Aviles et al., 2012b] presented Stream Ranking MF or RMFX which relies on a pairwise approach to MF and uses a selective sampling strategy based on active learning ideas to perform incremental model updates. The approach maintains a reservoir containing a fixed number of observations sampled from the stream of observations, and the model is updated by iterating through the reservoir instead of going through the entire stream. [Diaz-Aviles et al., 2012a] is based on the same approach and further investigated additional sampling strategies.

Further advances in Incremental Matrix Factorization (IMF). The approach introduced in [Huang et al.] handles ratings batch by batch and proposes to design a linear

transformation of user and item latent factors over time, at each model update. On the other hand, and instead of only updating vectors related to the current received observation, [Wang et al., 2013] explored multi-task learning and proposed to update additional vectors according to a user interaction matrix. Compared to most previously mentioned approaches that rely on vector retraining, a space retraining model is proposed in [Song et al., 2015]. The feature space is retrained via auxiliary feature learning and matrix sketching strategies.

Applications. [Das et al., 2007] was one of the first applications to tackle the problem of online recommendation in order to deliver news recommendations in real-time. The authors proposed a scalable approach combining different recommendation methods and using parallel computation. They also relied on a time-decaying function for user interactions and on a time-based window to evaluate co-visits of news. On the other hand, IMF has been implemented to address the recommendation problem in specific domains. [Pálovics et al., 2014] proposed a method for online music recommendation in a social media service. The approach is based on MF and exploits temporal and social influences between users to improve recommendations. Users that are socially connected are expected to exhibit similar preferences in close periods of time. [Huang et al., 2016] developed a scalable MF algorithm with an adjustable updating strategy for video recommendation. Different types of implicit feedback, e.g., click, watch and comment, are considered and each one of them is attributed a different confidence level regarding the degree of user interest that it exhibits. These confidence levels affect the value of the learning rate used to update the IMF model.

Beyond Matrix Factorization (MF). Relatively few efforts have been made to adapt methods other than MF to the online setting of recommendation. We mention in particular the application of Tensor Factorization (TnF) [Zhang et al., 2014], Factorization Machines (FM) [Kitazawa, 2016], online regression [Agarwal et al., 2010], and bagging [Vinagre et al., 2018], omitting further details about these methods.

User and item dynamics. Beyond efficiently updating models and integrating new users and items, online recommendation should account for the evolution of users and items over time. Recently, [Chang et al., 2017] presented a framework that handles streams via a continuous-time random process. Markov processes are meant to model each time-varying user and item factors in an attempt to capture the dynamics occurring in such settings. Another framework introduced in [Wang et al., 2018] is based on Bayesian Personalized Ranking (BPR) and claims to handle users' interest drifts, new users and items, and the system overload occurring in a streaming setting. It also relies on samples stored in a reservoir in order to capture users' long-term interests instead of only learning from recent observations.

7.4.3 Retrieval Module

Without loss of generality, the recommendation problem can be divided into two tasks: learning a utility function and estimating the ratings or relevance scores of items for a target user. In an online context, the utility function is constantly updated and scores have to be computed on the fly. In an attempt to be compliant with a streaming setting,

specific methods have to be developed and adopted to ensure an efficient retrieval of recommendations. Retrieving recommendations requires computing the scores for each item and selecting the N items scoring the highest. We review some of the methods in the following, even though not all of them are meant to work in a fully online setting.

[Koenigstein et al., 2012] propose to use metric trees, i.e., binary space-partitioning trees, to index item vectors, and use a branch and bound algorithm to obtain an exact solution. The technique is applied for neighborhood-based methods [Koenigstein and Koren, 2013] and MF methods [Koenigstein et al., 2012]. An approximate faster solution relying on a clustering of users is also proposed [Koenigstein et al., 2012]. Recommendations are computed for the cluster centers and presented as an approximate result to all users belonging to the corresponding cluster.

[Yin et al., 2015a] propose to compute one list of items per latent factor where items are sorted according to their generative probabilities with respect to the latent factor. Top-N items are computed for each list and returned in a priority list. An extended version of the threshold algorithm [Fagin et al., 2003] is used to update the priority of lists in addition to the threshold score. A similar idea is also implemented in [Yin et al., 2015b] for POI recommendation. [Teflioudi et al., 2015] propose to group the latent vectors into buckets of similar lengths and then solve a smaller cosine similarity search problem between vectors of each bucket. We note that since these methods require an offline step, they are not adapted, as originally defined, to the online setting where models are constantly being updated.

7.4.4 Evaluation Module

Batch offline evaluation. With a new formulation for the recommendation problem comes the need to design new appropriate evaluation methodologies. While the traditional batch mode of offline evaluation involving holdout methods is widely used to evaluate RS, it faces several limitations in the streaming setting (Section 2.2.1.1) [Vinagre et al., 2014a]. By randomly sampling data for training and testing, holdout methods ignore the temporal dimension and do not take into account the natural ordering of observations. Recommendation algorithms designed to handle ordered data cannot thus be evaluated with these methods. In addition, shuffling observations may result in illogical operations such as using future interactions to predict past interactions. Learning from shuffled observations would also prevent capturing dynamics occurring within users and items. While batch offline evaluation expects models to be static during the recommendation phase, online RS continuously update models as recommendations are delivered. Ideas for evaluating RS in streaming settings appeared in [Matuszyk and Spiliopoulou, 2014; Pálovics et al., 2014; Siddiqui et al., 2014; Vinagre et al., 2014a,b]. In an attempt to address the limitations faced by batch offline evaluation, most of them rely on the prequential methodology [Gama et al., 2013], presented in the following.

Prequential evaluation. Prequential evaluation consists of a *test-then-learn* procedure repeated for each received observation [Vinagre et al., 2014a]. Given an implicit feedback setting where received observations are in the form of (u, i), i.e., user u interacted with item i, the prequential evaluation methodology iterates over the following steps for each observation:

1. Use the current recommendation model to recommend N items for user u;
2. Evaluate the recommendation list given the observed item i;
3. Update the recommendation model using the observation (u, i) *(optional step)*.

The prequential methodology offers several benefits. Aside from respecting the chronological order of events, it allows the continuous monitoring of the RS performance in addition to its evolution over time. Recommendation algorithms can also exploit real-time statistics returned by prequential evaluation for drift detection and model adaptation. However, and in addition to the shortcomings inherited from the fact of being an offline evaluation method (Section 2.2.1.1), this methodology suffers from one particular limitation. The recommendation list generated by the RS is evaluated against a single item, i.e., item i, failing to acknowledge other potentially interesting recommendations occurring in the list. Therefore, it is not possible to distinguish between cases where the list holds relevant items that will be chosen at a later time and cases where it holds totally irrelevant items. [Vinagre et al., 2014a] reported that after performing experiments where the recommendation list is evaluated against all future observations of the target user instead of the current observation only, the overall computed metrics do not improve significantly. Nevertheless, a possible solution consists in adopting hybrid evaluation methods such as in [Siddiqui et al., 2014].

Metrics. While Section 2.2.3 introduces several metrics commonly used to evaluate RS, we discuss the consequences of adopting prequential evaluation on the metrics' definitions. The main characteristic in the streaming setting is that we are evaluating against a single item, and we are thus aware of the number of relevant items arising from the recommendation process. Metrics are evaluated and reported for each received observation (u, i). We present some of them in the following as defined in the scope of prequential evaluation [Frigó et al., 2017], and we denote by $rank_u(i)$ the rank of item i in the recommendation list of size N for user u.

$$Precision@N(u,i) = \begin{cases} 0 & \text{if } rank_u(i) > N \\ \frac{1}{N} & \text{otherwise} \end{cases} \tag{7.4}$$

$$Recall@N(u,i) = \begin{cases} 0 & \text{if } rank_u(i) > N \\ 1 & \text{otherwise} \end{cases} \tag{7.5}$$

$$DCG@N(u,i) = \begin{cases} 0 & \text{if } rank_u(i) > N \\ \frac{1}{\log_2(rank_u(i)\ +\ 1)} & \text{otherwise} \end{cases} \tag{7.6}$$

$$MRR@N(u,i) = \begin{cases} 0 & \text{if } rank_u(i) > N \\ \frac{1}{rank_u(i)} & \text{otherwise} \end{cases} \tag{7.7}$$

Given that there is only a single relevant item when applying the prequential evaluation methodology, no normalization is needed for the Discounted Cumulative Gain (DCG) metric. The Normalized Discounted Cumulative Gain (NDCG) is therefore not adopted, in opposition to the batch evaluation setting.

Evaluation protocol adopted. Since online RS usually require an initial phase of training, the operating environment is not exclusively a streaming environment. The evaluation should thus consider the switch between learning an initial model in batch and moving into the streaming setting. To this end, we rely on Part III of this thesis on the evaluation protocol proposed in [Matuszyk and Spiliopoulou, 2014] and presented in the following. The dataset is sorted chronologically and then split into three subsets, with the proportions 30%-20%-50%:

1. **Batch Train** subset, used as a training set for initializing the models in batch.
2. **Batch Test - Stream Train** subset, used as a test set for the model initially learned, and also used for incremental online learning to ensure the transition between (1) and (3).
3. **Stream Test and Train** subset, used for prequential evaluation [Vinagre et al., 2014a].

Statistical significance of the results is assessed using the signed McNemar test over a sliding window [Vinagre et al., 2014a].

7.5 Conclusion

Summary. With the explosion of the volume of user-generated data, designing online RS that learn from data streams has become essential. Taking into account the dynamics and drifts occurring within users and items is also important to continuously adapt models and guarantee a good performance. Based on these two ideas, Part III of the thesis addresses the problem of online adaptive recommendation. In this chapter, we formulate the considered problem, present the framework of online adaptive learning designed for stream mining, and review existing work related to online RS based on it, even though the problem of online RS remains an underexplored subject in the recommendation field. Previous work considers, on one side, having *a priori* knowledge about the relevance of old observations and about the way and the rate at which drifts are occurring (Section 7.4.1). On the other side, it focuses on passive approaches that continuously update models over time without explicitly detecting changes, resulting in an inability to adapt to sudden drifts (Section 7.4.2).

Relation to our work. To the best of our knowledge, previous work on online RS has not leveraged active approaches for drift detection and adaptation, which we investigate in Part III of the thesis. From the point of view of the RS, different types of drifts are expected to occur. While the change of user preferences has an impact on a *local* scale and only concerns a single user, a change of item perception or description affects recommendation across all users and has a more *global* impact. Both of them should be taken into account to ensure a good recommendation quality. The following chapters explore online adaptive recommendation considering both types of changes: Chapter 8 handles drifts at the user preference level, Chapter 9 handles drifts at the item perception level, and Chapter 10 handles drifts at the item description level.

Chapter 8

Dynamic Local Models

This chapter presents DOLORES, our approach to adapt to drifts in user preferences in online RS [Al-Ghossein et al., 2018b]. We leverage local models that are known for their ability to capture diverse and opposing preferences among user subsets. We automatically detect drifts of preferences that lead a user to adopt a behavior corresponding to another user subset, and adjust the recommendation models accordingly. Experiments on real-world datasets show promising results regarding the use of local models to adapt to user drifts.

8.1 Introduction

RS are based on the idea that the observed user behavior exhibits core preferences that will be the main driver of the user's future actions. An accurate modeling of user preferences can thus ensure a good prediction capacity. In reality, user preferences rely on variables that are hidden from the system and any change in these variables affect user preferences. The unavailability of all relevant factors complicates the modeling task. Being aware of the existence of these hidden variables and their dynamicity, and in order to guarantee a good recommendation quality, RS have to account for user drifts, i.e., changes in user preferences, that can only be detected when tracking the user behavior.

Changes in user preferences are mainly resulting from changes in local parameters affecting the individual user independently from the others. One example in the hotel recommendation domain can be observed when users' standards evolve due to a change in the social status. Users typically move from booking hotels in one segment, e.g., economy segment, to booking hotels in a higher one, e.g., luxury segment. In contrast, other changes may occur on a more global scale. They usually concern an entire subpopulation of users and derive from a change in item popularity or seasonality. As an example, preferences for destinations vary with trends, seasons, and the occurrence of impactful events. These dynamics can then be modeled at the item level. As mentioned in Chapter 7, previous work related to modeling user drifts in online RS faces several limitations. The proposed methods require fixing a set of parameters and anticipate that the RS should forget old observations. They also assume we have *a priori* knowledge about the way the user behavior changes and that preferences of all users drift at the same rate.

In this chapter, we present a novel incremental approach relying on item-based local models to learn user preferences and track their evolution in online RS. Our approach maintains one global model for all users and several local models built separately for each subset of users. Local models have been exploited for batch RS and are able to capture diverse and opposing preferences [Christakopoulou and Karypis, 2016]. We propose to continuously evaluate over time user assignments to subsets. Users are moved from one subset to a more adapted one when a change in preferences is detected, and user profiles are updated accordingly. Experiments on three real datasets show promising results.

The remainder of the chapter is organized as follows. In Section 8.2, we discuss related work on local models for recommendation. In Section 8.3, we present our approach performing online recommendation and adapting to changes in user behaviors. Experiments and results are presented and discussed in Section 8.4. Finally, Section 8.5 concludes the chapter.

8.2 Local Models for Recommendation

The idea of learning multiple local models instead of one global model for recommendation emerged due to its ability to capture diversity in users' behavior. While this idea is implemented differently in existing approaches, it is mainly based on the same intuition: A single global model may not be able to capture all preferences of a set of users in cases where diverse and opposing preferences exist while local models, if designed appropriately, can perform this task. In fact, a group of users may have similar tastes for one set of items but different tastes for another set. It is therefore more interesting to consider user or item subsets when performing recommendations instead of the whole set of users and items, requiring the need to design local models. We review in the following previous work on local models for recommendation.

One of the first recommendation approaches proposing to rely on local models is presented in [O'Connor and Herlocker, 1999]. In the context of rating prediction, items are clustered based on the observed feedback matrix. Predictions are then computed separately in each cluster given the local model built for each cluster. The authors reported mixed results concerning accuracy improvements but proved that the approach increases scalability considerably. Close to the idea of exploiting local models, [Koren, 2008] proposed to combine latent factor and neighborhood models for rating prediction. This approach can be assimilated with the idea of combining global tendencies through the latent factor model and local ones through neighborhood models.

[Xu et al., 2012] introduced an approach that co-clusters users and items and develops Collaborative Filtering (CF) methods separately for each cluster. Given that users and items may belong to several clusters, rating predictions are computed in each one of them and combined using weights associated with each pair of user and cluster. By using binary weights, the estimation of ratings relies on the cluster with the largest weight for the user.

[Lee et al., 2013] proposed LLORMA, standing for Local Low-Rank Matrix Approximation, that is based on the idea that the feedback matrix is locally low-rank and can be represented by the weighted sum of low-rank matrices. In other words, the assumption is

that the entire feedback matrix is not low-rank whereas submatrices restricted to certain types of similar users and items are. Given a set of anchor points, submatrices are defined in the neighborhoods of these points based on a distance metric between pairs of users and items. A local low-rank matrix is then estimated for each neighborhood. Estimating these matrices is done iteratively by first estimating latent factors corresponding to the anchor points and then re-estimating latent factors based on the similarities between observations and anchor points. This process is maintained until convergence. Ratings are computed as a weighted combination of the output of local models where the weight designates the similarity between anchor points and the computed element. While a first approach [Lee et al., 2013] used a squared error objective function, further advances investigated the use of a pairwise ranking objective function [Lee et al., 2014].

Instead of relying only on local models to perform recommendation, recent work proposed to combine global and local models for the task of top-N recommendation [Christakopoulou and Karypis, 2016]. Building on the success of item-based models, the authors proposed to compute relevance scores by using a user-specific combination of the predictions given by a global and a local item-based models. The item-based model adopted is the Sparse LInear Method (SLIM) [Ning and Karypis, 2011] which estimates item-item relations following a regularized optimization problem. A personalized weight is defined for each user indicating the interplay between the global and local component given the user behavior. The method operates by jointly optimizing the item-based models' estimation, the user-specific combination, and the assignment of users to local models. Compared to LLORMA, the proposed approach is based on user subsets instead of anchor points of the feedback matrix, considers updating user subsets for which the local models were estimated, and uses a global model. [Christakopoulou and Karypis, 2018] is based on the same ideas but uses latent factor models instead of item-based models.

[Beutel et al., 2017] introduced the problem of *focused learning*, claiming that when learning one recommendation model and optimizing for an average metric, many items are left badly modeled. Focused learning consists in estimating models that are meant to improve the modeling of a subset of items and enhance the prediction related to this subset. The defined problem is related to local modeling but existing local models are not sufficient to address focused learning. The authors formulate the problem as a hyperparameter optimization task where the idea is to find the hyperparameters optimizing an objective function that only covers a subset of items.

Relation to our work. Independently from the specific approach implemented, previous work has shown the interest in using local models to improve the recommendation quality. However, all existing approaches are not adapted to the online setting. Local models are not updated incrementally and data partitions are not determined in real-time. In this chapter, we exploit local models to take into account drifts in user preferences. Our approach is inspired by the method developed in [Christakopoulou and Karypis, 2016] for the advantages it offers and for its flexibility.

8.3 Proposed Approach

Motivation. The goal is to develop an online RS that detects the change in user preferences and adapts to it. Our approach extends item-based methods since it has been shown that they outperform user-based methods in most cases [Deshpande and Karypis, 2004]. A single item-based model may not be enough to capture the preferences of a set of users. In particular, a single model can not detect the diversity of preferences existing in user subsets. Local models built separately for each subset of users try, for their part, to represent fine-grained patterns.

We focus on detecting the change of preferences that would push a user to adopt a behavior that is different from the one of those belonging to the same subset and closer to the one of those assigned to another subset. In the hotel recommendation problem, this could be for example the consequence of a change in the user social status. The drift is therefore handled by assigning the user to a more adapted subset and updating the models accordingly. Our approach is designed to extend any incremental item-based method and we rely here on the item-based neighborhood method.

Incremental item-based neighborhood method. Item-based neighborhood methods explore similarities between items to provide recommendations. Relying on the cosine similarity to measure similarities in an implicit feedback setting and based on the notations introduced in Section 2.1.4, the similarity between item i and item j is given by:

$$sim_{CS}(i,j) = \frac{\mathbf{r}_i \,.\, \mathbf{r}_j}{\|\mathbf{r}_i\| \, \|\mathbf{r}_j\|} = \frac{|\mathcal{U}_i \cap \mathcal{U}_j|}{\sqrt{|\mathcal{U}_i|} \times \sqrt{|\mathcal{U}_j|}} \tag{8.1}$$

An incremental version of the item-based neighborhood method based on cosine similarity is proposed in [Miranda and Jorge, 2009] (Section 7.4.2.1). The idea is to maintain one counter c_i for each item i tracking its occurrence, i.e., the number of users that have interacted with it, and an additional counter c_{ij} for each pair of items (i,j) tracking their co-occurrence, i.e., number of users that have interacted with both items. For each received observation (u,i), the counters c_i and c_{ik} for each item k that u has interacted with in the past are incremented. Similarities between items can then be updated. At the recommendation step, nearest neighbors of items that the user has interacted with are computed and are used to assess items' scores.

Dynamic local models. The approach we present is called DOLORES, standing for Dynamic Local Online RS, and is shown in Algorithm 6. We maintain p local item-based models and one global item-based model. Each model is represented by an item-item similarity matrix $\mathbf{S}_*$. We learn p local similarity matrices denoted by $\mathbf{S}_l$ where l is the index of the user subset with which the local model is associated, and one global similarity matrix $\mathbf{S}_g$. At timestamp t, every user u belongs to one subset l_u where $l_u \in \{1, \ldots, p\}$ and the rating $\hat{r}_{ui}$ is given by:

$$\hat{r}_{ui} = \alpha_g.(\hat{r}_{ui})_g + (1 - \alpha_g).(\hat{r}_{ui})_{l_u} \tag{8.2}$$

$(\hat{r}_{ui})_g$ is computed using the global model, $(\hat{r}_{ui})_{l_u}$ is computed using the local model l_u, and α_g is the weight controlling the contribution of the global and local models.

Updating the models. Online RS assume that observations, i.e., user-item pairs (u, i), are continuously generated and handled. Each observation received is used to update the models, i.e., the similarity matrices, and to update the assignments of users to subsets. The idea is to detect that u is adopting a behavior that is no longer similar to users of the subset l_u. We assume that this is the case when we find that there is a local model that performs better than l_u for u. The change of user preferences also requires forgetting old information that is no longer relevant to the current behavior of u. We compare the performance of l_u using the full profile of u, i.e., $\mathbf{r}_u^\top$, against the performances of the other local models using only recent observations of $\mathbf{r}_u^\top$.

Comparing the performances of local models. Given the online setting, estimating the error of a model l_u is only possible using a single observation (u, i). We define the metric measuring the error of the model l_u when tested for the pair (u, i) as follows:

$$err(l_u, \mathbf{r}_u^\top) = 1 - \frac{1}{rank_u(i)} \tag{8.3}$$

where $rank_u(i)$ returns the ranking of item i in the item list sorted by decreasing order of $(\hat{r}_{u*})_{l_u}$. A better recommendation model should rank the relevant item higher in the list.

Forgetting irrelevant observations. When a change in the user behavior is detected, the assignment of u to user subsets is modified and his profile, i.e., the vector $\mathbf{r}_u^\top$ of the matrix $\mathbf{R}$, is updated. The old observations corresponding to the previously adopted behavior have to be forgotten. To this end, we rely on a forgetting strategy [Matuszyk and Spiliopoulou, 2014] where we keep in $\mathbf{r}_u^\top$ the last f items observed for u and remove the older observations. We denote by $\mathbf{r}_u'^\top$ the result of applying the forgetting strategy to $\mathbf{r}_u^\top$.

Initializing item-based models. Real-world RS usually have access to part of the data before running in an online fashion. We use part of the available data to perform an initial batch training (Section 7.4.4). Users are first separated into clusters either randomly or using a clustering algorithm. The local models and the global model are learned in batch. We then fix the learned models, iterate over users, and verify that the subset they belong to generates the smallest error on the observed user-item pairs. If this is the case, the user remains in the initial cluster. Otherwise, he is moved to the cluster with the smallest error.

Running time. We note that all the models can be trained in parallel and used for recommendation independently. Our approach introduces a very low overhead to the original item-based method we extend. As mentioned in Section 7.4.2.1, specific approaches can be adopted to speed up neighborhood methods and adapt them to the streaming setting given its constraints on resources. Therefore, running performance is not the main concern in the scope of our work but rather the impact of using local models on considering the evolution of user preferences.

Algorithm 6 Overview of our approach DOLORES

Data: stream of observations $\mathcal{O}$

for $o = (u, i)$ in $\mathcal{O}$ **do**
  Compute $err(l_u, \mathbf{r}_u^\top)$
  for $l \in \{1, \ldots, p\}$ such that $l \neq l_u$ **do**
    Compute $err(l, \mathbf{r}_u^\top)$
  end for
  set $l_{opt} = l$ such that $err(l_{opt}, \mathbf{r}_u^\top) = \min_l err(l, \mathbf{r}_u^\top)$
  if $err(l_{opt}, \mathbf{r}_u^\top) < err(l_u, \mathbf{r}_u^\top)$ **then**
    Update the similarity matrix of global model $\mathbf{S}_g$ with (u, i)
    Assign user u to the local model l_{opt}
    $\mathbf{r}_u^\top \leftarrow \mathbf{r}_u'^\top$
  else
    Update the similarity matrix of global model $\mathbf{S}_g$ with (u, i)
    Update the similarity matrix of local model l_u $\mathbf{S}_{l_u}$ with (u, i)
  end if
end for

Table 8.1: Statistics of the real-world datasets used to evaluate DOLORES

Dataset	# users	# items	# transactions
Ah_Eur	98,130	3,332	704,722
Ml-1M+	6,014	3,232	226,310
Ml-10M+	67,312	8,721	1,544,812

8.4 Experimental Results

In order to evaluate our approach, we rely on the evaluation protocol introduced in Section 7.4.4. We use two measures for evaluating the quality of recommendation, recall@N and DCG@N, as defined in Section 7.4.4 [Frigó et al., 2017]. The metrics are computed individually and averaged for all observations.

Datasets. The performance of DOLORES is evaluated on three datasets which characteristics are shown in Table 8.1. The Ah_Eur is extracted from the hotel industry. It is derived from the dataset Ah (Section 3.2) by selecting a cluster of European users as defined in Section 4.2. Ah_Eur gathers bookings of these users done during a period of three consecutive years. The Ml-1M+ and Ml-10M+ datasets are respectively derived from the MovieLens 1M and MovieLens 10M datasets which gather movie ratings given by users interacting with the MovieLens RS [Harper and Konstan, 2016]. In this chapter, and in order to use the data as implicit feedback, we follow the strategy employed in [Vinagre et al., 2014b] and we keep in the MovieLens datasets the pairs for which the rating is in the 20% of the rating scale of the dataset, i.e., rating equal to 5.

Parameters. We performed a grid search over the parameter space of the methods in order to find the parameters that give the best performance which we report. The number of local

Ah_Eur				
Method	**Recall@5**	**DCG@5**	**Recall@10**	**DCG@10**
KNNi	0.1621	0.1033	0.2819	0.1415
KNNi_{w}	0.1683	0.1042	0.2831	0.1421
DOLORES	**0.1852**	**0.1179**	**0.3045**	**0.1561**
DOLORES-G	0.1816	0.1165	0.3012	0.1548
LORES	0.1694	0.1075	0.2902	0.1461

Table 8.2: Recall@N and DCG@N for DOLORES, its variants, and other methods for the dataset Ah_Eur

Ml-1M+				
Method	**Recall@5**	**DCG@5**	**Recall@10**	**DCG@10**
KNNi	0.0407	0.0253	0.0716	0.0352
KNNi_{w}	0.0409	0.0258	0.0717	0.0354
DOLORES	**0.0521**	**0.0298**	**0.0798**	**0.0386**
DOLORES-G	0.0505	0.0289	0.0788	0.0381
LORES	0.0412	0.0271	0.0729	0.0364

Table 8.3: Recall@N and DCG@N for DOLORES, its variants, and other methods for the dataset Ml-1M+

models p examined took on the values: {2, 5, 10, 15, 20}. We fixed the number of neighbors to 100.

Methods compared. In order to demonstrate our proposed method, we evaluate the performance of several recommendation models including variants of the proposed method:

- KNNi denotes the incremental item-based neighborhood method.
- KNNi_{w} denotes the incremental item-based neighborhood method that uses a sliding window per user and retains only the last f items rated by each user. We set f=20 for all Ah_Eur, Ml-1M+, and Ml-10M+.
- DOLORES stands for Dynamic Local Online RS and denotes the method we propose in this chapter. We set $p = 15$, $f = 20$, α_g=0.5 for Ah_Eur, and $p = 15$, $f = 20$, α_g=0.7 for Ml-1M+ and Ml-10M+.
- DOLORES-G stands for Dynamic Local Online RS without a Global model. This is equivalent to setting the parameter α_g to 0 in Equation 8.2 and only using local models to perform recommendation. We set $p = 15$ and $f = 20$ for Ah_Eur, Ml-1M+, and Ml-10M+.
- LORES stands for Local Online RS. In LORES, the user assignment to subsets is fixed after the initial optimal assignment of each user. We set $p = 15$, $f = 20$, $\alpha_g = 0.5$ for Ah_Eur, and $p = 15$, $f = 20$, $\alpha_g = 0.7$ for Ml-1M+ and Ml-10M+.

Performance of the methods. Experimental results are shown in Tables 8.2, 8.3 and 8.4.

ML-10M+				
Method	**Recall@5**	**DCG@5**	**Recall@10**	**DCG@10**
KNNi	0.0591	0.0389	0.0943	0.0502
KNNi_{w}	0.0594	0.0390	0.0944	0.0502
DOLORES	**0.0602**	**0.0394**	**0.0953**	**0.0511**
DOLORES-G	0.0598	0.0392	0.0949	0.0508
LORES	0.0596	0.0390	0.0945	0.0505

TABLE 8.4: Recall@N and DCG@N for DOLORES, its variants, and other methods for the dataset ML-10M+

- By comparing KNNi with DOLORES and its variants, we show the importance of building local models and considering the change in user preferences instead of relying on one global model for all users. Given that all these approaches rely on the same base model, the gain in performance is due to the ability to capture fine-grained preferences through local models.
- By comparing KNNi_{w} with DOLORES, we highlight the advantage of using our approach instead of sliding windows. KNNi_{w} is systematically forgetting information over time while DOLORES considers that preferences evolve at different rates. Comparing KNNi and KNNi_{w} shows that forgetting information results in a minor improvement in performance especially for the ML-1M+ and ML-10M+ datasets where the benefit of using a sliding window is not really clear.
- By comparing DOLORES-G with DOLORES, we show the benefit of using a global model to capture global patterns instead of only using local models. However, as mentioned before, DOLORES-G still performs better than KNNi and KNNi_{w}.
- By comparing LORES with DOLORES, we demonstrate the benefit of reevaluating user assignments to subsets as new observations arrive. This also shows that the initial local model assigned to the user does not stay the most adapted one as time goes by, which can be explained by the shifting of user preferences over time. Results prove that taking into account this change of preferences has a high impact on the quality of recommendation. We note for example that for the AH_EUR dataset, user assignments to subsets are modified for 20% of the received observations.

DOLORES outperforms the other methods for the studied datasets and we can see the relative benefit of each of its components.

Discussion: Extending DOLORES to other models. In principle, the idea of using local models in an online setting while allowing user assignments to change over time can be applied to any recommendation model. We relied in this chapter on item-based neighborhood models. These models, like any other item-based model, offer one particular advantage in the scope of deploying the idea mentioned above. In fact, the user model is independent of the space of local models: It is thus possible to evaluate the performance of each local model with regards to the target user without the need to have a specific user model for each local model. In particular, in several item-based models, the user is represented by the set of items he previously interacted with and each local model defines a relation between pairs of

items. In contrast, the same idea cannot be directly applied to latent factor models where a user model, i.e., a vector of latent factors, is strongly related to the local model it is assigned to. This statement implies that local models are learned separately and independently. As a result, users and items assigned to different local models are not represented in the same space of latent factors. As mentioned in Section 8.2, [Christakopoulou and Karypis, 2018] proposed two approaches based on latent factor models and combining global and user subset specific sets of latent factors. The base model for both approaches is Singular Value Decomposition (SVD). Moving a user from a subset to another requires projecting him to the new subset and learning his projected latent factors. This solution may not be feasible in an online environment where computations should be done fast, and other solutions need to be considered.

8.5 Conclusion

Tracking the changes in user preferences in online RS raises unique challenges. On one hand, RS have to be able to process data streams and adapt to drifts in real-time. On the other hand, these drifts happen differently for each individual and knowledge about the way they occur is not available. In this chapter, we propose DOLORES, a novel approach based on local models, to address the problem. Relevance scores are computed using a global model and a local model representing a subset to which the user is assigned. Our approach automatically detects changes of preferences by continuously reevaluating the assignment of users to subsets and updating the models.

While our approach is based on the incremental item-based neighborhood method, further advances should consider relying on other recommendation models to benefit from their advantages. In particular, latent factor models constitute a promising direction given their superiority for the problem of recommendation. Some challenges related to extending latent factor models in the scope of DOLORES are mentioned in the discussion in Section 8.4. Another interesting direction would consider distinguishing between different types of drifts in user preferences, e.g., abrupt and gradual drifts, and handle them differently. While DOLORES considers drifts in user preferences, the next chapter provides a solution to account for drifts in item perceptions.

Chapter 9

Adaptive Incremental Matrix Factorization

This chapter presents AdaIMF, our approach to adapt to drifts in item perceptions in online RS. We exploit the successful framework of Matrix Factorization (MF) and design a novel adaptive learning rate schedule for item latent factors. Learning rates are dynamically adapted based on the performance of item models, ensuring that the MF model reflects the current state of users and items. Experiments on synthetic and real-world datasets demonstrate the effectiveness of AdaIMF and showcase its behavior in the presence of item drifts.

9.1 Introduction

Modeling users and items within the scope of a RS requires understanding the underlying dynamics and interactions occurring between both types of entities. While individuals may be independently affected by some hidden personal context, they may also be subject to common influences, uniformly affecting their perception of items. In the context of hotel recommendation, seasonality, holiday periods, and important events can impact hotel perceptions for all users considering to book a hotel. In fact, one hotel can be affected by the organization of a big sporting or business event in the surrounding region [Getz, 2008], and also by the political and economic situations of the country. In contrast to the example where an individual gets a new job with a higher income and revises his behavior accordingly, changes in item perceptions concern a whole group of users and should be modeled on the item level.

From all the different recommendation approaches that have been proposed, Matrix Factorization (MF) techniques have been widely used for Collaborative Filtering (CF) and are known to deliver good accuracy and scalability (Section 2.5.2) [Hu et al., 2008; Koren et al., 2009; Lee et al., 2013]. In its most basic form, MF represents both users and items by vectors of latent factors that are inferred from observations and used to generate recommendations. The model is usually learned using Stochastic Gradient Descent (SGD)

or other gradient-based methods. With the recent emergence of online recommendation, several incremental versions of MF have been proposed, enabling the integration of new feedback in real-time (Section 7.4.2.2). These approaches have the disadvantage of learning at a constant pace, independently from how users and items are actually evolving. As an example, the Incremental MF (IMF) approach proposed in [Vinagre et al., 2014b] updates the user and item models for each received observation by performing one step of the SGD algorithm and using a fixed learning rate.

In this chapter, we propose an adaptive learning rate method for IMF that dynamically adjusts the learning rates over time and accounts for changes happening in real-time at the item level for each item individually. Since item perceptions can be affected by unknown and unexpected events at different moments, it is important to integrate these dynamics into the model to ensure an accurate modeling of users and items. Each item is attributed a different learning rate in the learning process and when the item model is considered to be inconsistent with its current perception, the corresponding learning rate is automatically increased, enabling more learning from the newly received instances. Experiments on synthetic and real-world datasets show that our method outperforms other incremental approaches for online recommendation. We also show the importance of considering changes occurring at the item level with regards to the recommendation quality.

The remainder of the chapter is organized as follows. In Section 9.2, we discuss previous work related to learning rate schedules proposed for MF. In Section 9.3, we present our approach which is an adaptive learning rate method for IMF accounting for drifts on the item level. Experiments and results validating our approach are presented in Section 9.4. Finally, Section 9.5 concludes the chapter.

9.2 Learning Rate Schedules for Matrix Factorization

SGD and other gradient-based methods are widely adopted to learn MF models (Section 2.5.2) [Koren et al., 2009]. Applying these methods requires determining a proper learning rate: While small learning rates lead to a very slow convergence, high learning rates cause the model to diverge. Several adaptive learning rate methods have been recently proposed, aiming to achieve more informative gradient updates than fixed learning rates and ensuring a better learning process with regards to generalization and convergence. They have been applied in many domains, especially in neural networks where they have been achieving a good performance [Dean et al., 2012]. Following a brief reminder of SGD, we report in this section related work on adaptive learning rate methods in general, and on their particular application for MF.

Stochastic Gradient Descent (SGD). As mentioned in Section 2.5.2.3, SGD is a numerical optimization algorithm used to estimate the parameters optimizing an objective function $\mathcal{L}$. It is a version of the gradient descent approach where the actual gradient is approximated using the gradient at a single data point. In practice, we iterate over the set of available data points and update the parameters for each point considered until convergence. For

each observation o, the model parameters $\mathbf{x} \in \mathbb{R}^d$ are updated as follows:

$$\mathbf{x} = \mathbf{x} - \eta \frac{\partial \mathcal{L}(\mathbf{x}; o)}{\partial \mathbf{x}} \tag{9.1}$$

where η is the learning rate, determining the speed at which the optimal parameters are reached. Setting the value of η affects the learning process. In particular, high learning rates lead to a faster convergence but may cause numerical instability, and small learning rates imply slow convergence. In addition to the difficulty of choosing an appropriate learning rate, traditional settings apply the same learning rate to all parameter updates which may not always be convenient.

Adaptive learning rates. To address the learning rate issue, several adaptive learning rate methods were proposed and have proven to achieve a good performance in various applications. These methods are mainly designed to deal with sparse data and are based on the idea of increasing gradient updates for infrequent parameters and decreasing them for frequent parameters. In particular, AdaGrad [Duchi et al., 2011], standing for *Adaptive Gradient*, relies on this idea and was mainly exploited to improve the robustness of SGD in the scope of training neural networks. AdaGrad assigns a different learning rate for each parameter in $\mathbf{x}$ at a time step t, and updates are performed as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\eta}{\sqrt{\mathbf{G}_t + \epsilon}} \odot \mathbf{g}_t \tag{9.2}$$

where each element of $\mathbf{g}_t$, $g_{t,k}$, denotes the gradient of $\mathcal{L}(\mathbf{x})$ with respect to parameter x_k at time step t, $\mathbf{G}_t$ is a diagonal matrix where each diagonal element $G_{t,kk}$ is the sum of the squares of the gradients $g_{*,k}$ up to step t, and ϵ is a constant avoiding division by zero. Elements of $\mathbf{G}_t$ keep on accumulating positive terms corresponding to the squared gradients. Therefore, the denominator keeps on growing causing the learning rate to drop iteration after iteration, reaching a point where it is infinitely small and learning can no longer be performed. While this is achieving the initial goal of having higher learning rates for infrequent parameters and inversely, the monotonically decreasing learning rate prevents the algorithm to keep on acquiring knowledge when convergence is not reached yet.

AdaDelta [Zeiler, 2012] was proposed as an extension of AdaGrad to cope with the problem of continuously decreasing learning rates. Instead of keeping on accumulating squared gradients over all previous observations, only those included in a restricted sliding window of fixed size are considered. To avoid storing previous values and discarding them as they no longer fit in the window, a running average of the squared gradients is adopted. Given a decay constant ρ, the running average, denoted as $E[\mathbf{g}^2]_t$ at time step t, is computed as follows:

$$E[\mathbf{g}^2]_t = \rho E[\mathbf{g}^2]_{t-1} + (1-\rho)\mathbf{g}_t^2 \tag{9.3}$$

$\sqrt{\mathbf{G}_t + \epsilon}$ in Equation 9.2 is then replaced by $\sqrt{E[\mathbf{g}^2]_t + \epsilon}$, which is equivalent to the Root Mean Squared error of the gradient, $RMS[\mathbf{g}]_t$. Given that the units of updates do not match, the authors defined an additional running average for the squared parameter updates as follows:

$$E[\Delta\mathbf{x}^2]_t = \rho E[\Delta\mathbf{x}^2]_{t-1} + (1-\rho)\Delta\mathbf{x}_t^2 \tag{9.4}$$

where $\Delta \mathbf{x}_t = \mathbf{x}_{t+1} - \mathbf{x}_t$.

Finally, the model parameters are updated as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{RMS[\Delta \mathbf{x}]_{t-1}}{RMS[\mathbf{g}]_t} \mathbf{g}_t \tag{9.5}$$

Further advances along this direction introduced the Adam method [Kingma and Ba, 2015] where learning rates are adjusted by considering an exponentially decaying average of the first moment and the second moment of gradients.

Adaptive learning rates for Matrix Factorization (MF). Inspired by the success of adaptive learning rates for SGD in the context of neural networks and other techniques, several learning rate schedules were designed to learn the parameters of a MF model. These schedules are based on different intuitions related to the learning of user and item models, in addition to the nature of observations analyzed to perform learning, e.g., ratings. As mentioned before, the potential interest of adopting adaptive learning rate schedules resides in a faster convergence and a better generalization.

Vanilla SGD used to learn a MF model adopts a fixed schedule for the learning rate which is set to the same constant value for all parameters during the whole learning process [Koren et al., 2009]. A monotonically decreasing schedule was proposed in [Yun et al., 2014] where the learning rate is updated at each iteration over the whole set of observations and decreases over time. Given constant parameters α and β, the learning rate at iteration z is defined as follows:

$$\eta^z = \frac{\alpha}{1 + \beta \ . \ z^{1.5}} \tag{9.6}$$

On the other hand, early studies on backpropagation in neural networks introduced the *bold driver* heuristic [Battiti, 1989] which consists in adjusting the learning rate according to changes in the objective function values after each iteration. This approach was then leveraged for MF [Gemulla et al., 2011] where, starting from an initial learning rate η^0, the learning rate at the next iteration is increased if the loss decreases, and inversely. This is performed as follows:

$$\eta^{z+1} = \begin{cases} \alpha\eta^z & \text{if } \Delta_z < 0 \\ \beta\eta^z & \text{otherwise} \end{cases} \tag{9.7}$$

where $\alpha > 1$, $\beta \in [0, 1]$, and Δ_z is the difference between the value of the objective function at the beginning and the end of the z-th iteration.

Ideas developed in AdaGrad and AdaDelta were then introduced for MF. When considering the direct application of AdaGrad for MF, each parameter, i.e., each user latent factor and each item latent factor, is assigned a different learning rate that is being updated at each iteration. [Chin et al., 2015] argue that the cost of implementing AdaGrad for MF cannot be disregarded. The space complexity is $O((n+m)K)$ and each analyzed observation requires $O(K)$ additional operations to update the learning rate of each latent factor. [Chin et al., 2015] proposed then to adopt the same learning rate for all factors of one user model and one item model, reducing the space complexity to $O(n + m)$. An additional strategy was introduced, combining *slow learners* and *fast learners*, to prevent the learning rate from

dropping rapidly at the first few iterations, especially due to the random initialization of user and item latent factors. A similar approach based on AdaDelta was also exploited in [Wei et al., 2016].

The learning rate schedules presented so far are based on one of the following ideas: The learning rate should gradually decrease as we are expected to reach convergence with further iterations, the learning rate should increase when the loss function decreases and inversely, and different learning rates should be assigned for each user and item and should decrease for frequent parameters. Taking another direction, recent work [Li et al., 2018] proposed to adjust the learning rate based on the noisiness of the evaluated ratings: Noisy ratings are assigned small learning rates resulting in small gradient updates that prevent the MF model to be oversensitive to noise. These noisy ratings are expected to be associated with large training errors, and a different learning rate is associated with each rating r_{ui}, defined as follows:

$$\eta_{ui}^{z} = \frac{\eta_0}{\sqrt{err_{ui}^{(z-1)} + \epsilon}} + \beta \tag{9.8}$$

where err_{ui} accumulates squared training errors in estimating r_{ui} evaluated at each iteration and up to iteration z, η_0 is the predefined learning rate, ϵ is a constant avoiding division by zero, and β is a constant preventing the learning rate to become infinitely small. The authors also derived a theoretical analysis related to the convergence rate and the generalization error bound.

Adaptive learning rate for Incremental Matrix Factorization (IMF). While learning rate schedules are optimized for a faster convergence of MF models, the problem of online recommendation faces other constraints. Online RS are continuously handling user interactions and update recommendation models by analyzing each interaction only once. In contrast to the batch setting, it is not possible to iterate over the whole dataset as much as needed until convergence. In addition, convergence is not even desirable in such settings. We are looking to continuously integrate new observations and to represent the current state of users and items. Given the non-stationarity nature of environments, we expect models to constantly evolve over time. Nevertheless, latent factors of several IMF approaches are learned and updated using SGD (Section 7.4.2.2). In particular, the approach presented in [Vinagre et al., 2014b] relies on a fixed learning rate schedule and update models for each received observation.

Few IMF approaches leverage adaptive learning rates to learn the model parameters. In the context of video recommendation, [Huang et al., 2016] proposed an adjustable incremental SGD algorithm that updates the MF model in real-time. It has the particularity of handling different types of user interactions. The main assumption made is that each type of interaction, e.g., impression, click, and play, exhibits a different degree of interest. An interaction related to user u and item i is assigned a weight w_{ui} based on its type, where a higher weight indicates a stronger expected interest. These weights influence the learning procedure by adjusting the learning rate for the observation (u, i) as follows:

$$\eta_{ui} = \eta_0 + \alpha w_{ui} \tag{9.9}$$

where η_0 is the basic learning rate of the training process and α is the constant controlling the influence of weights on learning.

Relation to our work. Most learning rate schedules proposed to enhance the learning of MF are adapted to the batch setting: Learning is performed by iterating several times over the set of interactions and the learning rate is expected to decrease over time as convergence is reached. Converging to a static model is not desirable in an online RS and in a non-stationary environment. The model is expected to evolve over time and to follow changes occurring on the user and item levels. Convergence can only happen locally with respect to time, i.e., during bounded time periods where no changes are happening, and locally with respect to the model, i.e., some users and items can be static while others are not. In this context, learning rate schedules can be explored to control the learning process by enabling the model to learn more from relevant observations. We propose a learning rate schedule that accounts for drifts in item perceptions and we present our approach in the next section.

9.3 Proposed Approach

Motivation. In this chapter, we focus on the problem of online recommendation in non-stationary environments. We consider that item models may shift over time due to diverse unexpected and unknown events. Our goal is to take these shifts into account during the continuous learning process in order to learn models that are accurately representing the current state of users and items. Since each item can drift at a different moment and at a different rate, we reason at the single item level and independently for each one of them. The problem we address consists in detecting if a change happened, and then in adapting the model.

Given the implicit feedback setting (Section 2.5.2.4), we are only observing positive interactions, e.g., clicks and transactions, in real-time, and we aim to perform top-N recommendation for each observed user. Observations received for each item concern users for which the item should be ranked high in the recommendation list, i.e., the item should have a high relevance score, since these users are interested in the item. Given an appropriate recommendation model, our approach is based on the following idea: A drift in the item model results in the deterioration of the recommendation quality for users choosing this item. Therefore, when the item model is no longer delivering a good recommendation performance, we consider that it is no longer adapted to the perception of users selecting it. One way to measure the recommendation performance is to rely on the ranking of the item in the user recommendation list. In the absence of a drift, the performance of the item model, or the series of rankings delivered for each observed user selecting the item, is expected to be stable and learning is done at a constant rate. Otherwise, the ranking quality is expected to deteriorate as more observations are generated and as no specific strategy is adopted to handle the change. In this case and in order to adapt the item model, we propose to learn more from recent observations.

Our idea is to adopt a dynamic learning rate able to adapt the item model depending on its ranking quality for received observations. We expect to have a different learning rate for each item, defining at which rate the item model should learn from new observations. This rate should be fixed if there is no change detected when inspecting the stream of interactions concerning it. In contrast, it is expected to automatically increase when the recommendation performance decreases. However, the learning rate should not decrease

under a certain threshold to maintain continuous update and adapting to incremental drifts. As mentioned before, we are not looking to converge to a single static model since we consider non-stationary environments where users and items are constantly shifting. On the contrary, we assume that each batch of observations is generated by a single model and that this model is changing from one batch to the other.

Adaptive Incremental Matrix Factorization (AdaIMF). Our approach, AdaIMF, assigns a different learning rate for each item and is based on the IMF framework proposed in [Vinagre et al., 2014b] (Section 7.4.2.2). Learning rates automatically adapt according to the ranking quality for received observations: They increase when the ranking quality deteriorates and inversely. Inspired by previously proposed learning rate schedules (Section 9.2), we define the learning rate of AdaIMF for item i at time step t_i, i.e., t-th observation (u, i) for item i, as follows:

$$\eta_i^{t_i} = \eta_0 + \alpha f_i^{t_i} \tag{9.10}$$

where η_0 is the fixed predefined learning rate ensuring minimal learning at each step, α is a constant controlling the influence of the dynamic part on the learning rate, and $f_i^{t_i}$ is a dynamic variable revealing the ranking quality delivered for item i up to time step t_i.

The value of f_i depends on the ranking of item i in recommendation lists of users that have interacted with i up to time step t_i. It should be averaged on a number of observations to avoid being sensible to noisy data points and to consider how the performance changes over time. However, it should not account for all previous observations as variations can no longer be detected when the number of observations grows indefinitely. To enable the algorithm to be sensitive to recent trends, f_i monitors the ranking of i for recent observations included in a fixed size sliding window. We avoid storing all elements included in the window which is required by this solution, and we adopt the running average of rankings defined as follows:

$$f_i^{t_i} = \rho f_i^{t_i - 1} + (1 - \rho)\frac{rank_u^{t_i}(i)}{m} \tag{9.11}$$

where ρ is the decay constant controlling how much the algorithm is sensitive to recent observations, $rank_u^{t_i}(i)$ is the ranking of item i in the recommendation list of u, and m is the total number of items. When item i is being constantly ranked at the top of the list for users that are actually interacting with it, f_i will shrink. However, when i is ranked at the end of the list, f_i will increase causing the learning rate to increase too and the model to learn more from new observations. A stable ranking quality will also result in a stable learning rate leading to learning at a constant pace. η_i is expected to vary between $\eta_0 + \frac{\alpha}{m}$ and $\eta_0 + \alpha$. While one can argue that the gradient itself captures the prediction loss for item i on one observation, the adaptive learning rate tracks the loss on the series of consecutive observations.

Details related to the algorithm. The overall approach of AdaIMF is presented in Algorithm 7. As mentioned in Section 7.4.4, recommendation models used in online RS are first initialized in batch before the operating environment switches to a streaming environment. The values of f_i are expected to be initialized during the batch phase and represent the average ranking delivered for each item i before running the adaptive learning rate schedules. In order to compute $rank_u^{t_i}(i)$, the relevance score of each item is estimated for u using the scalar product between vectors $\mathbf{p}_u$ and $\mathbf{q}_i^\top$. These scores are then ordered by decreasing order and the rank of i is retrieved. On the other hand, given that AdaIMF only accounts

for drifts at the item level, user parameters are updated with the constant learning rate η_0, assuming that users are evolving at the same rate. Extending AdaIMF to handle user drifts will need careful considerations: User interactions are usually sparse and it is not clear if the feedback obtained from the actions of one user is enough to appropriately adapt the corresponding learning rate. Nevertheless, in the setting of AdaIMF, we assume that learning user latent factors at a constant rate is sufficient as a first solution.

Algorithm 7 Overview of our approach AdaIMF

Data: stream of observations $\mathcal{D}$
Input: number of latent factors K, initial learning rate η_0, constant α, decay constant ρ, regularization parameters λ_P and λ_Q
Output: $\mathbf{P}$, $\mathbf{Q}$

Initialize f_i^0 and $t_i = 1$ **for all** $i \in \{1, \ldots, m\}$
for (u, i) in $\mathcal{D}$ **do**
  if $u \notin Rows(\mathbf{P})$ **then** ▷ new user observed
    $\mathbf{p}_u \sim \mathcal{N}(0, \lambda_P^{-1}\mathbf{I_K})$
  end if
  if $i \notin Rows(\mathbf{Q})$ **then** ▷ new item observed
    $\mathbf{q}_i \sim \mathcal{N}(0, \lambda_Q^{-1}\mathbf{I_K})$
  end if
  Compute $rank_u^{t_i}(i)$ ▷ details in Section 9.3
  $f_i^{t_i} \leftarrow \rho f_i^{t_i - 1} + (1 - \rho)\frac{rank_u^{t_i}(i)}{m}$
  $\eta_i^{t_i} \leftarrow \eta_0 + \alpha f_i^{t_i}$
  $e_{ui} \leftarrow \mathbf{p}_u . \mathbf{q}_i^\top - 1$
  $\mathbf{p}_u \leftarrow \mathbf{p}_u - 2\eta_0(e_{ui}\mathbf{q}_i + \lambda_P \mathbf{p}_u)$
  $\mathbf{q}_i \leftarrow \mathbf{q}_i - 2\eta_i^{t_i}(e_{ui}\mathbf{p}_u + \lambda_Q \mathbf{q}_i)$
  $t_i \leftarrow t_i + 1$
end for

Complexity. Compared to vanilla SGD, AdaIMF requires tuning two parameters: α and ρ, knowing that ρ could be set around 0.9. In terms of space complexity, we introduce one variable per item resulting in $O(m)$. Additional operations involve updating one learning rate for each received observation, requiring the computations of f_i and η_i. We note that the complexity is lower than the one introduced by AdaError [Li et al., 2018] and by parameter-wise adaptive learning rate schedules such as AdaGrad [Duchi et al., 2011] which requires the update of K learning rates for each received observation.

9.4 Experimental Results

In this section, we present the experiments conducted to demonstrate the effectiveness of our approach, AdaIMF. We first use synthetic datasets to show how AdaIMF performs in the presence of simulated item drifts. We then report results for real-world datasets and show the interest of considering temporal dynamics in real-world scenarios. The evaluation

protocol is common in both series of experiments and is described in Section 7.4.4. Reported results correspond to observations included in the *Stream Test and Train* set. We use the metrics MRR@N, recall@N, and DCG@N to evaluate the quality of recommendations (Section 7.4.4).

9.4.1 Performance of AdaIMF on Synthetic Datasets

In the first series of experiments, we rely on synthetic datasets where we artificially introduce item drifts in models generating the data and we examine how AdaIMF performs compared to other incremental approaches.

Datasets. Given a MF model, i.e., a matrix of user latent factors $\mathbf{P}$ and a matrix of item latent factors $\mathbf{Q}$, we describe in the following the procedure used to generate synthetic data based on the model. The synthetic dataset consists of an ordered sequence of observations. For each observation to generate, we first sample uniformly a user u from the set of n users. Then, we sample an item i with a probability of $\frac{\mathbf{p}_u \mathbf{q}_i^\top}{\sum_{j=0}^{m} \mathbf{p}_u \mathbf{q}_j^\top}$ from the set of m items. This is explained by the fact that, given an accurate MF model, items with a higher score are more likely to be selected by the user, and the score of item i for user u is given by $\mathbf{p}_u \mathbf{q}_i^\top$. We report results for two synthetic datasets, denoted by SYN_1 and SYN_2, and introduced in the following.

The first synthetic dataset, SYN_1, is the concatenation of two batches of observations generated by two different MF models, where only one specific item model varies between both models. The idea is to evaluate the effect of the shift of this single item on the whole learning process and on the recommendation performance. In practice, we randomly initialize elements of the matrices $\mathbf{P}$ and $\mathbf{Q}$ from a normal distribution $\mathcal{N}(0, 0.1)$, and generate the first batch of observations. We then resample the latent factors of a single item i represented by vector $\mathbf{q}_i^\top$ and generate the second batch of observations. SYN_1 is obtained by processing one batch after the other. We set n and m to 50, the number of latent factors K to 10, and each batch consists of a random number of observations comprised between 50k and 100k.

In the second synthetic dataset, SYN_2, we simulate the case where several items are drifting independently at different moments. SYN_2 is the concatenation of several batches of observations where one item model is resampled between two consecutive batches. The item that is drifting is randomly chosen. We set n and m to 200, K to 10, and each batch consists of a random number of observations comprised between 10k and 20k.

Experimental results for SYN_1. Experiments run on SYN_1 consisted in comparing the performance of our approach `AdaIMF` with an IMF based on a Fixed Schedule for learning rates, i.e., using the same constant learning rate for all items, denoted by `FS-IMF`. To avoid learning a model before launching the streaming setting, we assume that, before any observation is received, the MF model in both methods is the one initially used to generate the first batch of SYN_1. Then, the model is updated as observations are received. As mentioned before, SYN1 consists of interactions generated by the same MF model except for one item model that shifts at a certain point in time. Figure 9.1a shows the evolution of

(A) Evolution of the learning rate associated with the drifting item, using `AdaIMF`

(B) MRR@m of `AdaIMF` and other variants

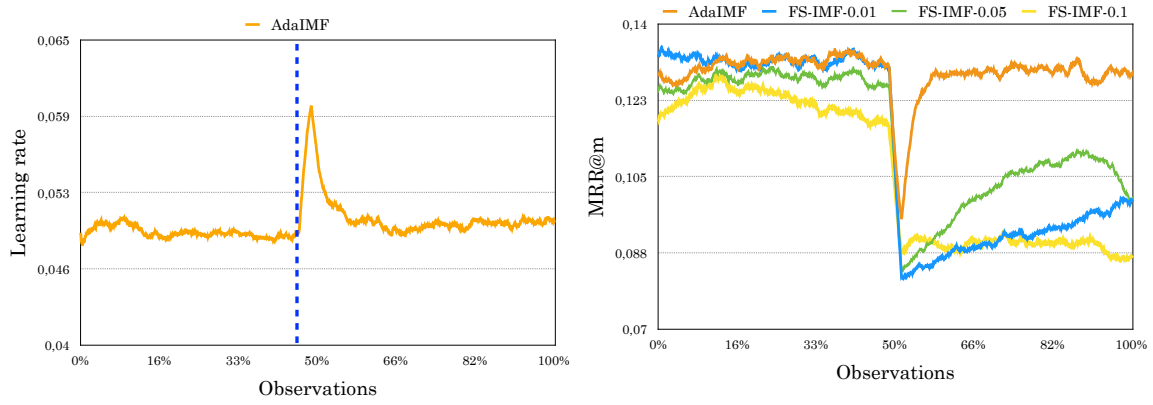


FIGURE 9.1: Experimental results for the dataset SYN_1 showing the behavior of `AdaIMF`

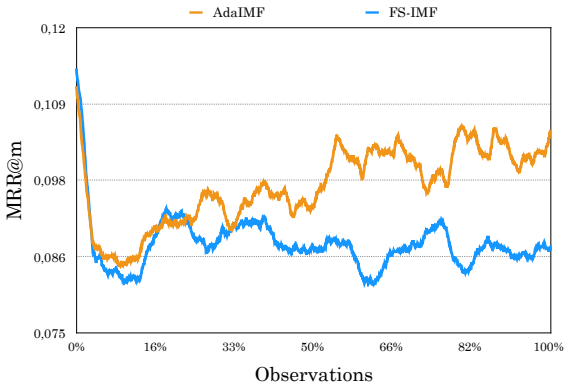


FIGURE 9.2: MRR@m of `AdaIMF` and `FS-IMF` for the dataset SYN_2

the learning rate associated with this particular item when using `AdaIMF` that is essentially designed to assign an adaptive learning rate for each item. The moment at which the drift effectively happens is marked by the blue dotted vertical line. The learning rate is first maintained almost constant as no change is detected. When the change happens, the ranking quality delivered for the item drops which causes the learning rate to increase. Learning from observations corresponding to the new model is then performed at a greater rate. As the ranking quality improves, the learning rate starts to decrease until regaining a constant value, which is slightly higher than the one adopted at the beginning of the experiment. This is due to the fact that, even though `AdaIMF` attempted to adapt to the item change, the ranking performance is still not as good as the one delivered at the beginning of the experiment, where the model adopted was specifically the one used to generate the observations.

Figure 9.1b shows MRR@m for `AdaIMF`, where the parameters $(\eta_0, \lambda_*, \alpha, \rho)$ are set to $(0.01, 0.01, 0.1, 0.9)$, and for several `FS-IMF` where the values of η_0, added as a suffix to the name of the method, vary between 0.01, 0.05, and 0.1, and where $\lambda_* = 0.01$. At the

beginning of the experiment, the performances of `AdaIMF` and `FS-IMF-0.01` are roughly equivalent. Significant differences in MRR@m for the different approaches appear when the item drift actually happens. We observe that `AdaIMF` passes through a transition phase where the performance instantly drops and regains slowly a reasonable level after adaptive learning is achieved. However, `FS-IMF` does not recover well after the item drift. Depending on the value of η_0 which needs to be tuned, the performance may slightly improve after learning is performed for some time, i.e., `FS-IMF-0.05`, or not at all, i.e., `FS-IMF-0.1`. We note that, even though only one item is drifting, its model is impacting the model of users who are interacting with it, which are in turn impacting models of other items they are interacting with, and so on. This experiment shows the importance of adopting an adaptive learning rate schedule for IMF, i.e., `AdaIMF`, even in a setting where only one item is drifting.

Experimental results for SYN$_2$**.** Experiments run on SYN$_2$ compared the performances of `AdaIMF` and `FS-IMF` in a setting where several items are drifting at different moments. The parameters $(\eta_0, \lambda_*, \alpha, \rho)$ are set to $(0.01, 0.01, 0.1, 0.9)$ for `AdaIMF` and the parameters (η_0, λ_*) to $(0.1, 0.01)$ for `FS-IMF`. Figure 9.2 shows MRR@m for both methods. Similarly to the protocol used for SYN$_1$, models are first initialized with the MF model used to generate the very first batch of observations. This explains the relatively good performance at the beginning of the experiment, which then drops as multiple drifts are occurring. Results show that `AdaIMF` outperforms `FS-IMF` and is able to maintain a good performance even though drifts are happening. `FS-IMF` witnesses a decrease in the average performance due to the fact that it is learning at a constant rate which is not sufficient to account for item drifts. Assuming that items are drifting over time, we show in these experiments that `AdaIMF` is well-designed to maintain a good recommendatin quality. Going further, experiments on real-world datasets, reported in the next section, prove that this assumption holds in real-world settings.

9.4.2 Performance of AdaIMF on Real-World Datasets

Datasets. We use several real-world datasets in order to demonstrate AdaIMF. We exploit datasets that can be used for the evaluation of online RS, that is datasets where interactions are labeled with corresponding timestamps. We also attempt to focus on datasets where item perceptions are expected to change over time. The five real-world datasets used are described below, and statistics related to these datasets are summarized in Table 9.1.

Booking dataset. The booking dataset used in this section is denoted by AH_EUR and is extracted from the hotel industry. It is derived from the dataset AH (Section 3.2) and is introduced in Section 8.4. AH_EUR gathers 704,722 bookings done by 98,130 European users in 3,332 hotels during a period of three consecutive years.

MovieLens dataset. The MovieLens 1M dataset, denoted here by ML-1M, gathers 1M movie ratings provided by 6,040 users on 3,706 movies. These ratings were originally collected through the MovieLens RS [Harper and Konstan, 2016].

Lastfm-1K dataset. The Lastfm-1K dataset [Òscar Celma, 2010], denoted here by LASTFM, gathers users' listening records between February 2005 and May 2009. We filter out users

TABLE 9.1: Statistics of the real-world datasets used to evaluate AdaIMF

Dataset	# users	# items	# transactions
AH_EUR	98,130	3,332	704,722
ML-1M	6,040	3,706	1,000,000
LASTFM	990	26,216	17,319,094
GOWALLA	18,737	32,510	1,278,274
FOURSQUARE	24,941	28,593	1,196,248

that have listened to less than 5 artists, and artists selected by less than 5 different users. The final dataset used contains 990 users, 26,216 artists, and 17,319,094 listening records.

Gowalla dataset. The Gowalla check-in dataset [1], denoted here by GOWALLA, was collected from February 2009 to October 2010 and gathers check-ins shared on Gowalla. Similar to [Liu et al., 2017], we filter out users with less than 15 check-ins and Points Of Interest (POIs) with less than 10 visitors. The final dataset used contains 18,737 users, 32,510 POIs, and 1,278,274 check-ins.

Foursquare dataset. The Foursquare dataset [2], denoted here by FOURSQUARE, was collected from April 2012 to September 2013 and gathers check-ins shared on Foursquare [Yang et al., 2016]. Similar to [Liu et al., 2017], we use check-ins done within the United States, filter out users with less than 10 check-ins and POIs with less than 10 visitors. The final dataset contains 24,941 users, 28,593 items, and 1,196,248 check-ins. We note that even though we exploit datasets related to POI recommendation, we do not consider any specific approach for this type of problem since we are interested in the more generic problem of online recommendation.

Parameters. We performed a grid search over the parameter space of the methods in order to find the parameters that give the best performance. We mainly report the performance corresponding to the parameters leading to the best results, unless stated otherwise.

Methods compared. AdaIMF is mainly compared against other incremental recommendation methods adapted to the online setting. We only consider one approach for IMF, which is the one proposed in [Vinagre et al., 2014b], given that the learning rate schedule we introduce can be applied to any other IMF approach relying on SGD. While we examine other existing learning rate schedules for IMF, we do not consider those that consist in indefinitely decreasing the learning rate over time, e.g., AdaGrad, since we want to ensure continuous learning from new observations. The methods we evaluate are listed in the following:

- `AdaIMF` is the approach we present in this chapter (Section 9.3) where we introduce an adaptive learning rate schedule attempting to dynamically adapt to item drifts. We set the parameters $(\eta_0, \lambda_*, \alpha, \rho)$ to $(0.01, 0.0001, 0.001, 0.9)$ for AH_EUR, $(0.001, 0.0001, 0.1, 0.9)$ for ML-1M, $(0.0001, 0.1, 0.1, 0.9)$ for LASTFM, $(0.001, 0.1, 1, 0.9)$ for GOWALLA, and $(0.01, 0.1, 0.1, 0.9)$ for FOURSQUARE.

[1] http://snap.stanford.edu/data/loc-gowalla.html
[2] http://sites.google.com/site/yangdingqi/home/foursquare-dataset

Metric	MF	FS-IMF	AdaDelta-IMF	AdaIMF
MRR@m	0.1026	0.1318	0.1401	**0.1543**
Recall@1	0.0624	0.0641	0.0645	**0.0751**
Recall@5	0.1502	0.1781	0.1941	**0.2147**
Recall@10	0.1811	0.2416	0.2842	**0.3171**
Recall@50	0.2342	0.2933	0.6143	**0.6439**
Recall@100	0.2621	0.3177	0.7046	**0.7306**
DCG@5	0.1083	0.1111	0.1259	**0.1459**
DCG@10	0.1184	0.1552	0.1601	**0.1788**
DCG@50	0.1303	0.1669	0.2273	**0.2512**
DCG@100	0.1348	0.1708	0.2421	**0.2654**

TABLE 9.2: Performance of `AdaIMF` and its variants for the dataset AH-EUR for $K = 20$

- `AdaDelta-IMF` is an IMF approach that adopts the AdaDelta learning rate method in an online context [Zeiler, 2012](Section 9.2). Each item is assigned a learning rate which value depends on the accumulated squared gradients over recent observations. We set the parameters $(\eta_0, \lambda_*, \rho)$ to $(0.01, 0.0001, 0.9)$ for AH_EUR, $(0.001, 0.0001, 0.9)$ for ML-1M, $(0.01, 0.001, 0.9)$ for LASTFM, $(0.01, 0.001, 0.9)$ for GOWALLA, and $(0.01, 0.1, 0.9)$ for FOURSQUARE.

- `FS-IMF` is the IMF approach proposed in [Vinagre et al., 2014b] and adopting a Fixed Schedule (FS) for learning rates, i.e., the learning rate is the same for all items and remains constant during the training process. We set the parameters (η_0, λ_*) to $(0.01, 0.0001)$ for AH_EUR, $(0.01, 0.01)$ for ML-1M, $(0.0001, 0.1)$ for LASTFM, $(0.01, 0.1)$ for GOWALLA, and $(0.01, 0.1)$ for FOURSQUARE.

- `MF` designates a MF approach that is only trained in batch on the first subset of observations. It is evaluated on the *Stream Test and Train* set (Section 7.4.4) without being updated when receiving new observations. Learning in batch is performed by iterating over the first subset of observations until convergence. We set the parameters (η_0, λ_*) to $(0.01, 0.0001)$ for AH_EUR, $(0.001, 0.0001)$ for ML-1M, $(0.0001, 0.1)$ for LASTFM, $(0.01, 0.001)$ for GOWALLA, and $(0.01, 0.1)$ for FOURSQUARE.

- `Knni` is the incremental item-based approach proposed in [Miranda and Jorge, 2009] (Section 7.4.2.1) and using the cosine similarity. We set the number of neighbors to 300 for all datasets.

- `Rand` randomly selects items for recommendation.

Results. Metrics measured for `AdaIMF` and the other methods are presented in Tables 9.2 to 9.6 for all datasets, having $K = 20$. Results show that `AdaIMF` outperforms the other methods for the studied datasets. The proposed adaptive learning rate strategy allows the model to deliver better recommendation quality in terms of MRR, recall, and DCG, by adapting to the dynamics occurring on the item level. `MF` performs worse than the other methods since the model is not being updated at all as new observations are being received, showing the interest of considering online RS where observations are integrated in real-time.

Metric	MF	FS-IMF	AdaDelta-IMF	AdaIMF
MRR@m	0.0041	0.0091	0.0087	**0.0099**
Recall@1	0.0006	0.0016	0.0014	**0.0022**
Recall@5	0.0032	0.0078	0.0071	**0.0084**
Recall@10	0.0063	0.0152	0.0137	**0.0157**
Recall@50	0.0272	0.0663	0.0635	**0.0673**
Recall@100	0.0495	0.1194	0.1191	**0.1215**
DCG@5	0.0019	0.0046	0.0041	**0.0053**
DCG@10	0.0029	0.0071	0.0063	**0.0076**
DCG@50	0.0073	0.0177	0.0168	**0.0185**
DCG@100	0.0109	0.0263	0.0257	**0.0272**

TABLE 9.3: Performance of AdaIMF and its variants for the dataset ML-1M for $K = 20$

Metric	MF	FS-IMF	AdaDelta-IMF	AdaIMF
MRR@m	0.0218	0.0319	0.0608	**0.0708**
Recall@1	0.0098	0.0151	0.0349	**0.0367**
Recall@5	0.0265	0.0373	0.0843	**0.0948**
Recall@10	0.0401	0.0568	0.1108	**0.1331**
Recall@50	0.1076	0.1582	0.1856	**0.2661**
Recall@100	0.1577	0.2361	0.2553	**0.3458**
DCG@5	0.0183	0.0262	0.0604	**0.0666**
DCG@10	0.0226	0.0324	0.0691	**0.0789**
DCG@50	0.0371	0.0541	0.0854	**0.1078**
DCG@100	0.0452	0.0667	0.0919	**0.1208**

TABLE 9.4: Performance of AdaIMF and its variants for the dataset LASTFM for $K = 20$

Metric	MF	FS-IMF	AdaDelta-IMF	AdaIMF
MRR@m	0.0114	0.0291	0.0256	**0.0448**
Recall@1	0.0059	0.0161	0.0133	**0.0315**
Recall@5	0.0142	0.0361	0.0321	**0.0573**
Recall@10	0.0205	0.0503	0.0465	**0.0682**
Recall@50	0.0445	0.1094	0.1031	**0.1182**
Recall@100	0.0631	0.1526	0.1429	**0.1791**
DCG@5	0.0101	0.0262	0.0229	**0.0452**
DCG@10	0.0121	0.0308	0.0274	**0.0487**
DCG@50	0.0172	0.0436	0.0396	**0.0557**
DCG@100	0.0203	0.0506	0.0461	**0.0587**

TABLE 9.5: Performance of AdaIMF and its variants for the dataset GOWALLA for $K = 20$

Metric	MF	FS-IMF	AdaDelta-IMF	AdaIMF
MRR@m	0.0623	0.0708	0.0771	**0.0813**
Recall@1	0.0462	0.0503	0.0546	**0.0569**
Recall@5	0.0776	0.0894	0.0974	**0.1027**
Recall@10	0.0917	0.1092	0.1191	**0.1258**
Recall@50	0.1277	0.1634	0.1798	**0.1937**
Recall@100	0.1485	0.1921	0.2123	**0.2308**
DCG@5	0.0629	0.0708	0.077	**0.0811**
DCG@10	0.0674	0.0772	0.0841	**0.0884**
DCG@50	0.0754	0.0891	0.0974	**0.1034**
DCG@100	0.0787	0.0938	0.1027	**0.1094**

TABLE 9.6: Performance of AdaIMF and its variants for the dataset FOURSQUARE for $K = 20$

When comparing the results for FS-IMF and AdaDelta-IMF, it is not possible to form a general opinion about which one performs better in all cases. While AdaDelta-IMF outperforms FS-IMF for the datasets AH-EUR, LASTFM, and FOURSQUARE, it is not the case for the datasets ML-1M and GOWALLA. Understanding the behavior of AdaDelta-IMF in an online setting is not trivial. In general, a higher absolute value of the gradient is derived from a more important loss evaluated for a specific data point. According to AdaDelta's schedule, the learning rate decreases when the squared gradients recorded from previous observations are significant. This can be related to the idea of drifts in item models. When the item model is no longer adapted to the current observations, the learning rate in AdaDelta-IMF may decrease and learning from new observations is not performed at high rates, in contrast with AdaIMF. One can imagine that this may be beneficial in settings where the change is occurring on a limited period of time. Instead of learning at a constant rate like in FS-IMF, it can be interesting to decrease the learning rate and reduce thus the sensibility of the model to the received observations. On the other hand, AdaDelta-IMF limits updates for frequent parameters, which may not be necessarily beneficial in an online dynamic environment. In all cases, AdaIMF performs better than both of these methods. The improvements brought by AdaIMF are amplified when the number of latent factors, K, increases, which can be observed when comparing these results with the ones we describe next.

Figures 9.3, 9.4, and 9.5 show the recall@N and DCG@N for the datasets LASTFM, GOWALLA, and FOURSQUARE, for different values of N and for $K = 50$. When increasing the number of factors, the superiority of the different approaches changes. In particular, FS-IMF outperforms AdaDelta-IMF for all represented datasets. In addition, for LASTFM and for small values of N, the performance of AdaDelta-IMF is comparable to, or even worse than, the one of MF. AdaDelta-IMF may then not be adapted to the setting of online recommendation. This observation may not necessarily come into sight when using the traditional batch evaluation protocol. This shows, once again, that the adoption of online RS and sequential learning and evaluation can invalidate several recommendation techniques that perform well in batch mode. We note that Knni performs better than MF, proving the need to develop incremental approaches that integrate interactions as soon as they are generated. Finally, and as shown in the previous series of experiments, AdaIMF outperforms all the other methods for the studied datasets. This observation strengthens the idea that

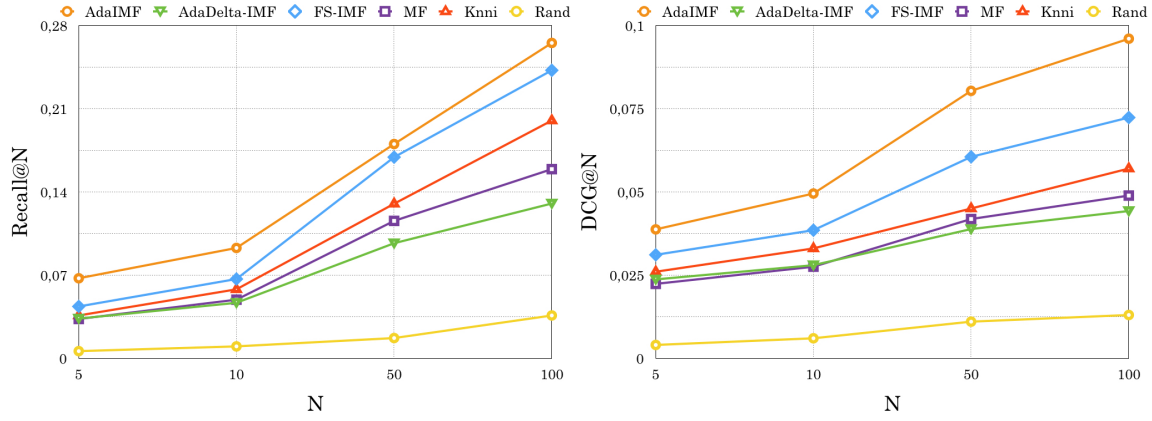


FIGURE 9.3: Recall@N and DCG@N of `AdaIMF` and other variants and incremental methods for the dataset LASTFM, using different values of N, for $K = 50$

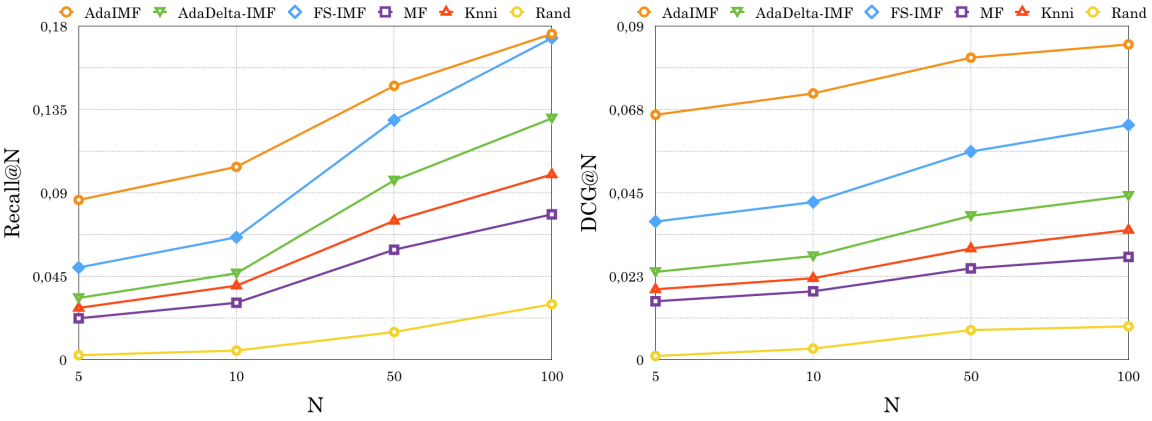


FIGURE 9.4: Recall@N and DCG@N of `AdaIMF` and other variants and incremental methods for the dataset GOWALLA, using different values of N, for $K = 50$

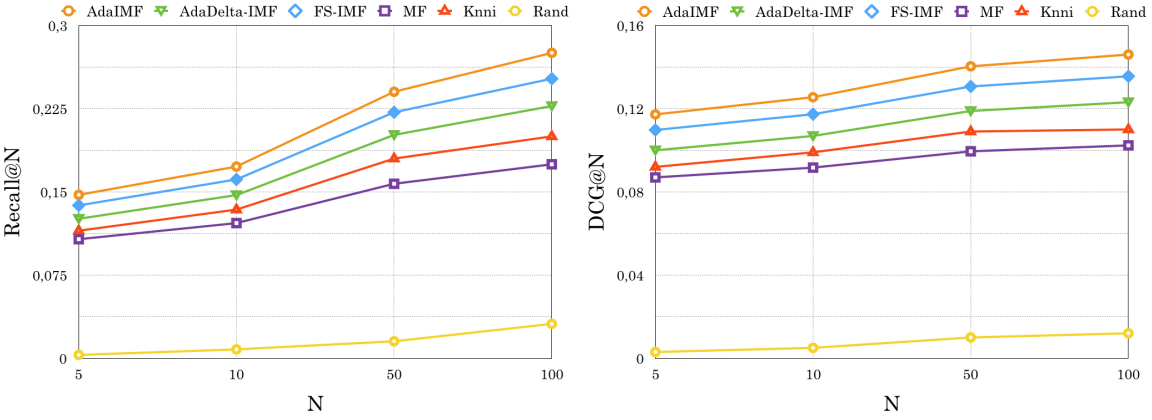


FIGURE 9.5: Recall@N and DCG@N of `AdaIMF` and other variants and incremental methods for the dataset FOURSQUARE, using different values of N, for $K = 50$

item drifts are occurring in the environment of RS and have to be taken into account to improve the recommendation quality.

9.5 Conclusion

Within the ecosystem of RS intersect multiple users and items that are shifting simultaneously while constantly influencing each others. Adapting to these changes in online RS raises several challenges, especially due to the large number of concepts that are being tracked and to the lack of knowledge regarding the dynamics that are occurring. While some types of shifts affect each user individually, others impact the item perceptions for a whole group of users and have to be modeled on the item level. In this chapter, we propose AdaIMF, an adaptive learning rate schedule for IMF that takes into account drifts in item perceptions. Each item is assigned a learning rate that automatically increases when the performance of the item model deteriorates, enabling more learning from recent observations corresponding to the new item perception. Experiments on synthetic datasets show that AdaIMF is able to maintain a good recommendation quality in the presence of item drifts. Experiments on real-world datasets prove the superiority of AdaIMF when compared to other incremental approaches and highlight the dynamics occurring in real-world settings.

While AdaIMF assumes that the whole item model should be updated at the same rate, real-world scenarios may suggest that the multiple aspects of a single item are evolving independently. These aspects may be represented by the latent factors, and future work should investigate the adoption of an adaptive learning rate for each latent factor, based on the ideas of AdaIMF. While this solution can induce a higher complexity, we may consider subgroups of items for which aspects are evolving in the same way.

While the approach proposed in this chapter inspects the dynamics in user interactions to adapt to item drifts, the next chapter presents our approach leveraging textual descriptions of items to integrate drifts.

Chapter 10

Adaptive Collaborative Topic Modeling

This chapter presents CoAWILDA, our approach to adapt to drifts in item descriptions in online RS [Al-Ghossein et al., 2018e,f; Murena et al., 2018]. We propose an online hybrid approach that handles the item cold-start scenario by supporting the integration of new items into the recommendation model in real-time. CoAWILDA combines a Collaborative Filtering (CF) component, a topic model, and a drift detection technique, and is well-designed to automatically adapt to drifts occurring in real-time. Experiments on synthetic and real-world datasets show the effectiveness of CoAWILDA and demonstrate how it reacts in the presence of drifts.

10.1 Introduction

Collaborative Filtering (CF) mainly suffers from rating sparsity and from the cold-start problem. Auxiliary information like texts and images has been leveraged to alleviate these problems, resulting in hybrid RS. Collaborative Topic Regression (CTR) [Wang and Blei, 2011] is an example of a popular hybrid approach combining probabilistic topic modeling for content analysis [Blei et al., 2003] and latent factor models for CF [Pan et al., 2008]. While hybrid approaches were proven to be a good solution to get the best of Content Based Filtering (CBF) and CF approaches, existing hybrid RS are meant to work in batch: They cannot continuously maintain models up to date and are not able to adapt to changes in user preferences and item descriptions which occur due to temporal dynamics.

In this chapter, we address the problem of online hybrid recommendation in a dynamic environment where users interact with items in real-time and where new items are expected to arrive accompanied by a textual description. This setting is common in — but not restricted to — the news and tweet recommendation domains, for example, where new articles and tweets are continuously generated and read by users.

We propose an adaptive collaborative topic modeling approach for online recommendation. Our approach combines AWILDA, an adaptive version of online Latent Dirichlet

Allocation (LDA) [Hoffman et al., 2010] that is able to analyze and model documents arriving in a stream on one side, and Incremental Matrix Factorization (IMF) [Vinagre et al., 2014b] to leverage user interactions for the learning of preferences on the other side. Our approach is adaptive as we actively detect changes in topics that may occur in the document stream and adjust accordingly. We alternate online refinement of the topic model when no drift is found with batch training when it is needed. The decision of retraining and the chunk of data on which we retrain the model are automatically determined by an adaptive sliding window technique [Bifet and Gavalda, 2007].

The purpose of introducing our approach is threefold. First, it is fully incremental and can thus be used in an online setting to generate recommendations. Second, since it is a hybrid approach relying on users' past interactions and on textual information, it addresses in particular the item cold-start problem which, to the best of our knowledge, has not been studied yet in the context of online recommendation. Third, its capacity of automatically detecting and adapting to drifts makes it suitable for real-world scenarios where changes in topics of document streams are frequently happening due to unexpected events and need to be considered for a better quality of recommendation.

The chapter is structured as follows. In Section 10.2, we review previous work on several topics related to our work. Section 10.3 presents our approach for adaptive topic modeling and Section 10.4 our approach for online hybrid recommendation. Experiments and results are reported and discussed in Section 10.5. Finally, Section 10.6 concludes the chapter.

10.2 Related Work

In this section, we review related work on hybrid RS, relevant variants of LDA, and news recommendation which shares some commonalities with the problem addressed.

Hybrid RS. While several trends for hybrid recommendation exist (Section 2.6), we consider in this chapter the one that incorporates content-based characteristics into CF approaches. Hybrid RS are able to recommend new items in the item cold-start scenario by leveraging auxiliary information. They also help alleviate the sparseness of feedback data, thus improving the quality of recommendation.

Previous work has utilized text data such as abstracts [Wang and Blei, 2011], synopses [Wang et al., 2015], and reviews [Bao et al., 2014]. Several techniques have been used to model documents like LDA [Wang and Blei, 2011], stacked denoising autoencoders [Wang et al., 2015], and convolutional neural networks [Kim et al., 2016]. Images have also been leveraged in this context and visual appearances of items can be added to the preference model [He and McAuley, 2016].

While most hybrid RS are designed to work in batch, we propose an online hybrid RS that is able to address the item cold-start problem in a dynamic environment where items are added in real-time. The model generating item descriptions is expected to change over time due to some hidden factors and affects the recommendation process globally across all users. Among the few works that considered concept drifts in the context of online recommendation, the focus has been on considering local changes occurring on the user

level. In this chapter, we consider topic modeling for handling document streams and we manage to detect drifts on the item level.

Topic modeling and concept drifts. Topic modeling is a machine learning task which consists in associating a document seen as an unordered list of words with a vector of *topics*, i.e., of word distributions. One of the most influential topic modeling method is the generative model of Latent Dirichlet Allocation (LDA) [Blei et al., 2003] (Section 4.3.1). LDA models a document as a multivariate distribution, the parameter of which is drawn from a Dirichlet distribution. The popularity of LDA is due to its simplicity and modularity, as well as its interpretability. We refer the reader to [Jelodar et al., 2017] for a detailed survey on LDA.

Vanilla LDA is not designed for evolving environments but only to infer topics based on a batch of accessible documents. Variants have been proposed to add a temporal aspect to LDA, including situations where the distribution evolves over time. A first variant, called Dynamic Topic Models (DTM) [Blei and Lafferty, 2006], considers that the word-topic distribution varies over time. At each time step, parameters are re-evaluated, conditioned by their values at the previous step. The same idea, but implemented at the level of a paragraph in a book, is proposed by SeqLDA [Du et al., 2010]. A major drawback of these methods and other temporal adaptations of LDA (such as [AlSumait et al., 2008; Griffiths and Steyvers, 2004]) is the use of time slices, the size of which is arbitrary and does not depend on the observed data. In particular, changes in topic distribution can happen within a time period significantly smaller than the length of the chosen window. Continuous time models offer solutions to this problem [Iwata et al., 2010]. A pioneer continuous time method [Wang and McCallum, 2006] modifies LDA by assuming that the word distribution over topics depends on word co-occurrences, and also on the date of the document. Despite the benefit of this method, it cannot be used in practice for stream analysis since the learning is made offline: It requires the whole dataset to be accessible in one batch to be able to infer the model.

For this reason, existing topic modeling methods are not suitable for data stream mining. Learning from data streams presents two major difficulties: the online nature of learning and the presence of concept drifts (Section 7.3). Regarding the former point, an online learning algorithm for LDA has been proposed in [Hoffman et al., 2010]. It is based on online stochastic optimization and can handle documents arriving in a stream by using each one of them to update the topic model parameters. On the other hand, concept drift designating the possible change in distribution that can happen in temporal and non-stationary environments, we propose to use change detection methods to estimate changes of topics in document streams. While the approach presented in [Hoffman et al., 2010] passively update the model for each received document, we actively detect drifts and update the model accordingly. Among active methods, ADaptive WINdowing (ADWIN) [Bifet and Gavalda, 2007] gained a lot of interest recently for the simplicity of its approach (comparing average values of a time series on subwindows) and for the theoretical guarantees it proposes. We propose to combine LDA model and ADWIN algorithm for drift detection in streams of documents.

News recommendation. The problem we address in this chapter is common in the setting of tweets [Diaz-Aviles et al., 2012b], articles [Wang and Blei, 2011], and news recommendation [Epure et al., 2017]. Our approach can be used to perform online recommendation of new items in any domain, whenever a textual description is available. We review in particular the related problem of news recommendation since it has been specifically studied in the literature.

News articles are continuously generated and while some of them could be relevant several weeks after the publication date, others have a shorter life cycle. In addition, readers' behavior is unstable compared to other domains given that users are often affected by external events such as breaking news and important occasions. Therefore, news recommendation often considers recency and popularity [Ahmed et al., 2012; Doychev et al., 2014]. This is done for example by filtering candidate articles for recommendation based on recency [Billsus and Pazzani, 2007], by periodically re-building models using fresh data [Das et al., 2007], or by using knowledge from Twitter to determine popular events [De Francisci Morales et al., 2012], among other solutions. Readers' interests are captured using the categories of the articles (if available) or keywords related to the articles' topics [Doychev et al., 2014; Liu et al., 2010a], and are used for CBF recommendation. Hybrid approaches combining CBF and CF usually perform best for the problem of news recommendation [Das et al., 2007; Lin et al., 2014; Liu et al., 2010a]. Combining long-term and short-term preferences has also proven to be beneficial in this context [Epure et al., 2017; Liu et al., 2010a]: While long-term preferences designate users' genuine interests, short-term preferences can be triggered by temporary events and fade away with time. In addition, session-aware RS have been used to address this problem given that users are not always identified when browsing on news platforms. Session-aware RS focus on transitions between items, formulating the problem as a Markov decision process [Epure et al., 2017] or using recurrent neural networks [Quadrana et al., 2018].

Our work addresses the wider and more generic problem of online recommendation that could occur in any domain and with no existence of sessions. We only require a textual description of items which can be easily collected from abstracts or reviews. We leverage CF and online topic models, and we consider the evolution of content through drift detection for modeling items.

10.3 Proposed Approach for Online Topic Modeling

In this section, we present our algorithm designed for topic drift detection in LDA using ADWIN, which is called Adaptive Window based Incremental LDA (AWILDA). LDA is a probabilistic graphical model designed to provide a definition of documents based on latent features called topics. We refer the reader to Section 4.3.1 for a presentation of LDA and for additional details about notations used in this section. While training of LDA can be performed either offline [Blei et al., 2003] or online [Hoffman et al., 2010], our approach considers the online training given our real-time setting. We first start by presenting ADWIN followed by a detailed presentation of AWILDA, our proposed approach.

ADaptive WINdowing (ADWIN). ADWIN [Bifet and Gavalda, 2007] is an algorithm developed for active mining of data streams. The idea of ADWIN is to keep a sliding window W with the most recent observations of a stream of real-valued elements x_t. At every time step, the algorithm adds the new element to the window W and decides whether the new window W contains a drift or not. In a supervised context, these observations usually correspond to risk measures.

The principle of ADWIN can be summed up as follows. The algorithm compares means of elements of all possible sub-windows W_0 and W_1 of W. If the difference of means μ_{W_0} and μ_{W_1} of the two sub-windows is large enough and the size of the windows is large enough, then a drift is detected. The non-rigorous notions of "large enough" are defined through the choice of a statistical test for the detection.

Besides its real simplicity, ADWIN admits a couple of interesting theoretical properties. Among them, bounds are given for the probability of *incorrectly* splitting the current window, i.e., false positive rate bound, and *correctly* splitting the window, i.e., false negative rate bound.

Motivation. The proposed method for topic change detection is based on the use of ADWIN combined with a training of LDA. We consider a framework in which documents arrive one by one in the form of a document stream. A document received at time step t is denoted by $\mathbf{w}_t$. Given parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ where $\boldsymbol{\alpha}$ is the parameter of the Dirichlet distribution and $\boldsymbol{\beta}$ the word-topic distribution, and given known latent variables $(\boldsymbol{z}, \boldsymbol{\theta})$ where $\boldsymbol{z}$ is the set of topics and $\boldsymbol{\theta}$ the topic distribution in a document, the likelihood of the model is given by:

$$\mathcal{L}(\mathbf{w_t}) = \frac{\Gamma\left(\sum_i \alpha_i\right)}{\prod_i \Gamma(\alpha_i)} \int \left(\prod_{i=1}^{K} \theta_i^{\alpha_i - 1}\right) \left(\prod_{m=1}^{M} \sum_{i=1}^{K} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_m^j}\right) d\theta \tag{10.1}$$

where Γ is the gamma function, w_n^j measures the quantity of word j in the document, K is the number of topics, V is the vocabulary size, and M is the number of words in the document.

A change in the stream of likelihood corresponds to a change in the data distribution and can be detected by ADWIN algorithm. The retained documents, i.e., documents kept in the window after a drift is detected, correspond to the documents received after the drift. The principle of our method relies on the following intuitions:

- The likelihood measures the generative quality of the model with regards to observed data. When a model is not adapted, the likelihood decreases.
- ADWIN is sensitive to changes in the mean value of a time series. Thus, it will detect a change in the likelihood caused by a change of the model.
- ADWIN will select large windows to train a new LDA model. The drift will be predicted with a better accuracy for large window sizes, which is also optimal to train a LDA model.

Following this idea, our method can be described as follows. At time step t, the system has access to a LDA model $\mathtt{M_t}$ which describes the data. When the system gets a new document, we compute the likelihood of observing the document, adds it to the current window, and inspects it with ADWIN to check if a drift occurred. When a drift is detected, the current LDA model is trained on the documents selected by the retained sub-window.

Algorithm. The algorithm we present, AWILDA, is a direct implementation of these ideas. AWILDA relies on two LDA models. The first model, denoted by $\mathtt{LDA_m}$, is used for document modeling. The second model, denoted by $\mathtt{LDA_d}$, is only used for the detection of drifts. The algorithm works as described in Algorithm 8.

Algorithm 8 Overview of AWILDA

for $\mathbf{w_t}$ in the stream of documents $\mathcal{W}$ **do**
  Compute likelihood $\mathcal{L} = p(\mathbf{w_t}|\mathtt{LDA_d})$ for model $\mathtt{LDA_d}$
  Process $\mathcal{L}$ with ADWIN
  if ADWIN detects a drift for the window decomposition $W = W_0W_1$ **then**
    Retrain $\mathtt{LDA_m}$ based on the documents in W_1
    Retrain $\mathtt{LDA_d}$ based on the documents in W_1
  end if
  Update $\mathtt{LDA_m}$ using the document $\mathbf{w_t}$ based on the online LDA algorithm
end for

We split the task of prediction and the task of drift detection by separating the models. The model used for prediction, $\mathtt{LDA_m}$, is kept up to date while no drift is detected. In contrast, the model $\mathtt{LDA_d}$ is not modified; otherwise, the detected changes might not originate only from a change in the data distribution, but also from the change of the model. Both models are retrained on the batch of documents retained by ADWIN once a drift is detected. This results in a *hybrid* adaptation to drifts for $\mathtt{LDA_m}$, combining active and passive approaches, i.e., incrementally updating the model for each received document and a more significant update when a drift is detected.

The AWILDA algorithm benefits from all the advantages of ADWIN. In particular, it offers a strict control of false positive rate and false negative rate. If the underlying data generation process does not change, the distribution of the likelihood is stationary, which makes the theorems relative to ADWIN still valid. We discuss in the following how the theoretical properties of ADWIN can be transposed for AWILDA.

Theoretical guarantees. AWILDA presents interesting theoretical properties which guarantee the quality of its results regarding drift detection. Using notations similar to the ones introduced in [Bifet and Gavalda, 2007], we consider a window W of length n_w which is divided into two sub-windows W_0 and W_1 of respective sizes n_0 and n_1. Let h be the harmonic mean of n_0 and n_1 (hence $\frac{1}{h} = \frac{1}{n_0} + \frac{1}{n_1}$). We suppose that, in ADWIN, the drift is detected for $|\hat{\mu}_{W_1} - \hat{\mu}_{W_0}| \geq \epsilon_{cut}$ (where $\hat{\mu}_{W_0}$ designates the mean value over sub-window W_0). Let δ be such that:

$$\epsilon_{cut} = \sqrt{\frac{1}{2h} \ln \frac{4n_w}{\delta}} \tag{10.2}$$

With these parameters, Theorem 3.1 in [Bifet and Gavalda, 2007] ensures both false positive rate bound and false negative rate bound. These results can be adapted to our setting.

Theorem 10.1. *At every time step, if documents are generated by a single LDA model in the time period covered by W, the probability that AWILDA detects a drift at this step is at most δ.*

Proof. On the covered window, ADWIN gets a time series $X_t = \mathcal{L}(D_t)$ where D_t are equally distributed (for a single LDA model) and $\mathcal{L}$ represents the likelihood of $\texttt{LDA}_\texttt{d}$ which is constant on W for AWILDA. Thus the mean of the variables remains constant on W. The conclusion follows from the properties of ADWIN. □

Following the same direction, the following theorem can be proven for false negative rate bound:

Theorem 10.2. *Suppose that, at a time step t, window W can be split in two parts W_0 and W_1 and documents are independent and identically distributed by a LDA distribution $\texttt{LDA}_\texttt{0}$ (resp. $\texttt{LDA}_\texttt{1}$) on sub-window W_0 (resp. W_1). If $|\mathbb{E}_{D\sim\texttt{LDA}_\texttt{d}}[p_{\texttt{LDA}_\texttt{1}}(D) - p_{\texttt{LDA}_\texttt{0}}(D)]| \geq 2\epsilon_{cut}$, then with probability $1-\delta$ AWILDA detects a drift inside sub-window W_1.*

Proof. The idea of the proof is the same. The mean value of $X_t = \mathcal{L}(D_t)$ on sub-window W_0 is:

$$\mu_t = \mathbb{E}_{D\sim\texttt{LDA}_\texttt{0}}[p_{\texttt{LDA}_\texttt{d}}(D)] = \mathbb{E}_{D\sim\texttt{LDA}_\texttt{d}}[p_{\texttt{LDA}_\texttt{0}}(D_t)]$$

An equivalent result can be found for W_1, and the theorem comes directly. □

Unlike for Theorem 10.1, a simple interpretation of Theorem 10.2 is not direct. For instance, two LDA models can be distinct and not share the targeted property. Finding conditions on the parameters of the three distributions is an interesting task that we will not address. However, it has to be noticed here that the guarantee on the false negative rate depends on the choice of $\texttt{LDA}_\texttt{d}$.

In practice, concept drift can happen in different ways (Section 7.3). The case of abrupt drift has been explicitly studied with the setting of Theorem 10.2. It corresponds to the case where the document distribution changes from one given state to another between sub-windows W_0 and W_1. Results given in [Bifet and Gavalda, 2007] show that the detection delay can be estimated by $O(\mu \ln(1/\delta)/\epsilon^2)$ where μ is the mean of the distribution before drift. In our case, this delay is of critical importance since it defines the size of the chunk for retraining the model. AWILDA faces a trade-off between predicting a drift as early as possible (in order to maximize the likelihood) and collecting as much data as possible to get a good estimator of the underlying LDA model.

The case of gradual drift is less adapted to the developed framework. Properties of ADWIN have been demonstrated in case of linear gradual drift, but these results are difficult to translate directly into our setting where the time series tracked by ADWIN has a complex mathematical definition. Understanding the behavior of AWILDA in the case of gradual drift is a task that would come together with a proper study of Theorem 10.2.

Practical considerations. As defined in Equation 10.1, the likelihood of a LDA model is not computable. Thus, we rely on an upper-bound $\mathcal{L}'$ proposed in variational inference (see Equation 1 in [Hoffman et al., 2010]). In practice, the results observed with this upper-bound are not satisfying due to the lack of precision: The probabilities of observing documents are very low and the method fails to discriminate them with enough accuracy. In order to overcome this difficulty, we consider the logarithm of $\mathcal{L}'$ (hence an upper-bound of the log-likelihood). This quantity is theoretically unbounded, which is a problem for ADWIN, but in practice, it is observed that the values vary only in a small interval (the width of which depends on the dataset). In our experiments, we prevented this quantity to decrease too much by fixing a minimal bound so that the quantity of interest becomes bounded. A reasonably low value for this threshold was never reached in the scope of the presented experiments (Section 10.5.1). However, we do not have any method to evaluate an optimal value for this bound in a general case.

10.4 Proposed Approach for Online Recommendation

In this section, we present our proposed algorithm for adaptive collaborative topic modeling, CoAWILDA, merging two components: AWILDA for topic modeling with drift detection and Incremental Matrix Factorization (IMF) for CF (Section 7.4.2.2, Algorithm 5) [Vinagre et al., 2014b]. CoAWILDA is based on the framework of Collaborative Topic Regression (CTR) (Section 4.3.1). It benefits from the advantages of CTR, i.e., coping with sparsity and item cold-start, is adapted to the online setting, and takes into account the non-stationarity nature of data in an evolving environment.

In our setting, observations are supposed to arrive in real-time and are mainly of two types. First, *interactions*, denoted by (u, i), designate positive actions, e.g., clicks, ratings, performed by users and related to a certain item. Second, *additions of items*, denoted by $(i, \mathbf{w}_i)$, usually occur when a new item becomes available at a certain time step and we consider that it is accompanied by a textual description.

CoAWILDA is presented in Algorithm 9. When a new item is received, we use AWILDA to model the descriptive document and extract topic proportions $\boldsymbol{\theta}_i$. The item latent vector $\mathbf{q}_i$ representing an item i results of the addition of the topic proportions $\boldsymbol{\theta}_i$ and an item latent offset $\boldsymbol{\epsilon}_i$. When a new interaction (u, i) is observed, we update the user latent factor $\mathbf{p}_u$ and the item latent offset $\boldsymbol{\epsilon}_i$ following the procedure of IMF (Section 7.4.2.2). Recommendation is performed as defined in the MF framework, and $\hat{r}_{ui} = \mathbf{p}_u.\mathbf{q}_i^\top = \mathbf{p}_u.(\boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i)^\top$.

10.5 Experimental Results

In this section, we present the experiments we conducted to prove the effectiveness of our approach. We first show how AWILDA performs when modeling a stream of documents and then discuss how CoAWILDA performs when addressing the problem of online recommendation, using synthetic and real-world datasets.

Algorithm 9 Overview of CoAWILDA

Data: stream of observations $\mathcal{O}$
Input: number of factors K, learning rate η, regularization parameters λ_P and λ_Q
Output: $\mathbf{P}$, $\mathbf{Q}$

for o in $\mathcal{O}$ **do**
  if $o = (i, \mathbf{w}_i)$ **then** ▷ new item added
    $\boldsymbol{\theta}_i \leftarrow AWILDA(\mathbf{w}_i)$
    $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \lambda_Q^{-1}\mathbf{I_K})$
    $\mathbf{q}_i \leftarrow \boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i$
  end if
  if $o = (u, i)$ **then** ▷ interaction observed
    if $u \notin Rows(\mathbf{P})$ **then** ▷ new user observed
      $\mathbf{p}_u \sim \mathcal{N}(0, \lambda_P^{-1}\mathbf{I_K})$
    end if
    $e_{ui} \leftarrow \mathbf{p}_u.\mathbf{q}_i^\top - 1$
    $\mathbf{p}_u \leftarrow \mathbf{p}_u - 2\eta(e_{ui}\mathbf{q}_i + \lambda_P\mathbf{p}_u)$
    $\boldsymbol{\epsilon}_i \leftarrow \boldsymbol{\epsilon}_i - 2\eta(e_{ui}\mathbf{p}_u + \lambda_Q\boldsymbol{\epsilon}_i)$
    $\mathbf{q}_i \leftarrow \boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i$
  end if
end for

Synthetic datasets for evaluating AWILDA. To demonstrate the ability to detect drifts, we generate synthetic datasets where we artificially insert drifts at random moments throughout the sequence of documents. Synthetic datasets are denoted by SD_R, where R is the number of simulated drifts. Documents observed between two consecutive drifts are generated by a single LDA model following its generative process. At each occurring drift, we draw uniformly the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The number of topics is fixed for all the models used to generate one dataset.

We present experiments performed on the following two synthetic datasets: SD_4 and SD_9, containing 4 and 9 drifts respectively. Handling document streams is a very common task in environments where short texts are generated and shared, e.g., newswires, Twitter. Therefore, we choose to generate documents containing 100 words, and we fix the vocabulary size to 10,000 words and the number of topics, K, to 15. Following the setting in [Blei et al., 2003], the elements of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are first set to $50/K$ and 0.1 respectively, and are then changed at each drift. In SD_4, we generate exactly 2,000 documents from each distribution, separating two consecutive drifts by the same number of documents. In SD_9, we vary the number of documents generated by each model between 500 and 1,000 documents.

Semi-synthetic datasets for evaluating AWILDA. We also conduct experiments on semi-synthetic data. We use the dataset REUTERS-21758 [1] consisting of newswire articles classified by categories and ordered by their date of issue. The ApteMod version of this database contains 12,902 documents and each document is classified in multiple categories

[1] http://archive.ics.uci.edu/ml/

TABLE 10.1: Statistics of the real-world datasets used to evaluate CoAWILDA

Dataset	# users	# items	# transactions	Period
ML-100K	1,000	1,700	100,000	18 mos.
PLISTA	1,362,097	8,318	32,706,307	1 mo.

for a total of 90 categories. In the procedure of data preprocessing, we down-cased and stemmed all words in the articles.

To demonstrate the functionality of AWILDA, we reorder the newswire articles based on their categories. We artificially ensure a sudden emergence of topics at specific points of the document stream, trying to provoke a drift in the topic distributions. We derive from the initial ordered dataset two sets of articles that we use in our experiments. In the first set, denoted by REUTERS_1, we select the articles belonging to the category "acq" followed by the articles belonging to the category "earn". We expect the algorithm to detect the sudden change in topics mentioned in the documents. In the second set, denoted by REUTERS_4, we select all articles from a specific category and add them consecutively to the dataset, considering the five following categories: "interest", "trade", "crude", "grain", and "money-fx".

Real-world datasets for evaluating AWILDA and CoAWILDA. Data used to evaluate our approach for online recommendation should be chronologically ordered and should include user interactions and the addition of items over time with a corresponding textual description. These two characteristics are not available in all datasets commonly used to evaluate RS. In our work, we use two real-world datasets: the ML-100K and the PLISTA datasets.

The ML-100K dataset corresponds to the MovieLens 100k dataset [2] [Harper and Konstan, 2016] and gathers 100,000 ratings from 1,000 users on 1,700 movies, spanning over 18 months. Since we are addressing the problem of recommendation with implicit feedback, our goal is to recommend the movies the user is going to rate. Movies become available according to their reported release date, and we use DBpedia [3] to collect abstracts written in English and describing each one of them.

The PLISTA dataset is described in [Kille et al., 2013] and contains a collection of news articles published in German on several news portals. The available dataset captures interactions collected during the month of February 2016. We remove from the dataset interactions corresponding to unknown users, users with less than three interactions, and items with no available textual description. Finally, the dataset gathers 32,706,307 interactions from 1,362,097 users on 8,318 news articles.

Documents from both datasets have been preprocessed by mainly removing stop words, removing words occurring once, and stemming remaining words. Statistics about the real-world datasets used are summarized in Table 10.1.

[2] http://www.movielens.org
[3] http://www.dbpedia.org

10.5.1 Performance of AWILDA for Online Topic Modeling

Evaluation protocol. AWILDA is proposed to model a stream of documents using drift detection. In our experimental setting, we consider that we are receiving the documents one after the other, ordered by their availability date, e.g., release or publication date for real-world datasets. For each received document, we first evaluate the topic model and then process the document to update the underlying model. We use the first 20% documents of the stream to initially train the model which is evaluated and updated using the remaining documents. Our evaluation concerns the tasks of topic drift detection and document modeling.

Task of topic drift detection. We evaluate the ability to detect drifts by checking the latency between the moment when the real drift occurs and the moment it is detected. This is only performed for synthetic and semi-synthetic datasets since they are annotated with the occurrence of drifts whereas real-world datasets are not.

Task of document modeling. Given a LDA model trained on a set of documents, the goal in document modeling is to maximize the likelihood on unseen documents ($\mathcal{D}_{test}$). Perplexity is the tool used by default in language modeling and measures the ability of a model to generalize to new data [Blei et al., 2003]. It is defined as follows:

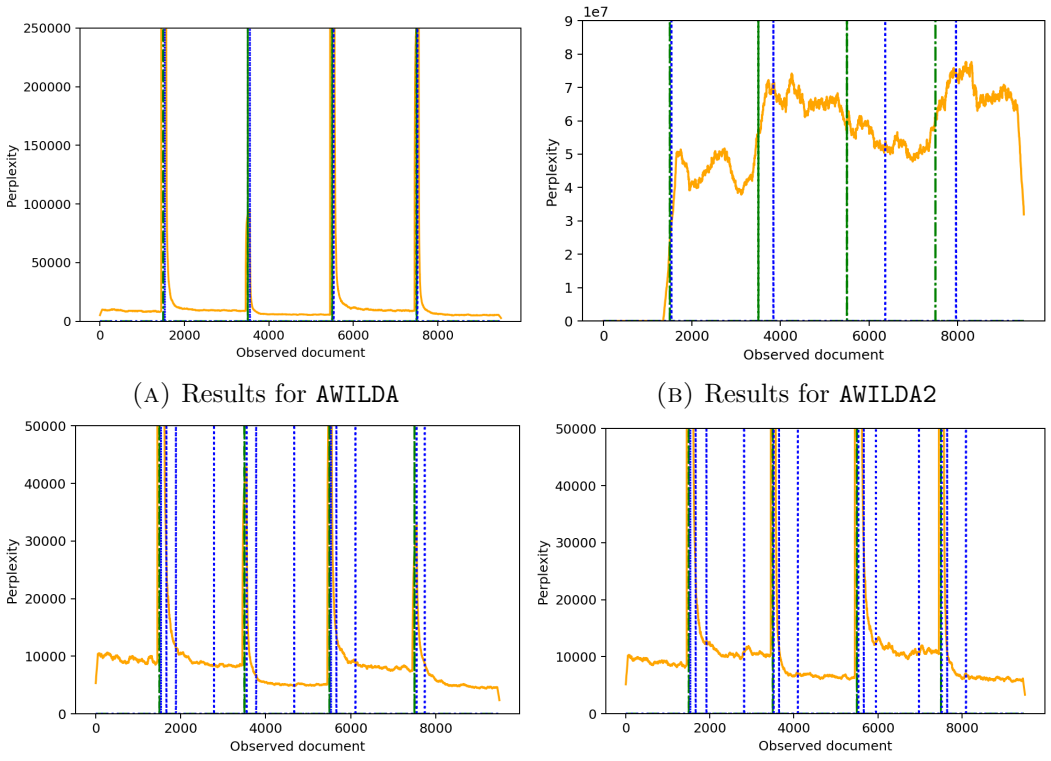
$$perplexity(\mathcal{D}_{test}) = \exp\left\{-\frac{\sum_{d=1}^{|\mathcal{D}_{test}|}\log_2 p(\mathbf{w}_d)}{\sum_{d=1}^{|\mathcal{D}_{test}|} M_d}\right\} \tag{10.3}$$

where M_d designates the number of words in document d. In the actual use of perplexity, the probability $p(\mathbf{w}_d)$ is approximated by its upper-bound (given by the variational inference) as explained in Section 10.3. A lower value of perplexity indicates a better generalization capacity. Since we are handling document streams, the perplexity is reported for each received document using the current model. We note that in AWILDA, the perplexity is measured using $\texttt{LDA}_\texttt{m}$ since it is the actual module used to model documents.

Methods compared. Considering the task of topic drift detection, we compare AWILDA to three other variants. In these variants, the model $\texttt{LDA}_\texttt{m}$ is updated in a similar way as for AWILDA, but the methods differ in the way the detection model $\texttt{LDA}_\texttt{d}$ is updated:

- `AWILDA2`. $\texttt{LDA}_\texttt{d}$ is initially trained on a small chunk of documents that is used to initialize the models. It is not updated as more documents are being received.
- `AWILDA3`. $\texttt{LDA}_\texttt{d}$ is updated for each received document. It is therefore equivalent to a classical online LDA model [Hoffman et al., 2010].
- `AWILDA4`. $\texttt{LDA}_\texttt{d}$ is updated for each received document using the online LDA algorithm. In addition, when a drift is detected, the model is retrained on the sub-window selected by ADWIN.

Regarding the theoretical study, it can be easily verified that Theorem 10.1 and Theorem 10.2 remain valid for `AWILDA2`, but not for `AWILDA3` and `AWILDA4`. In particular, it is noticeable that we do not have guarantees for the performances of `AWILDA3` and `AWILDA4`



(A) Results for AWILDA

(B) Results for AWILDA2

(C) Results for AWILDA3

(D) Results for AWILDA4

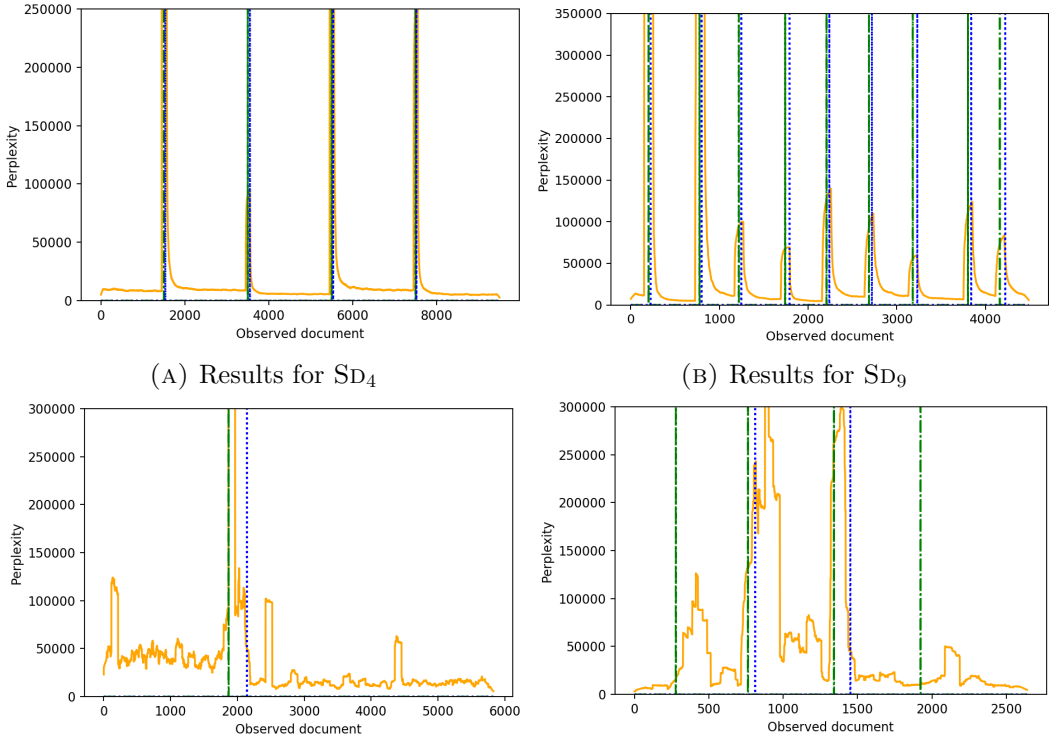
FIGURE 10.1: Topic drift detection for the dataset SD_4 using AWILDA and its variants

since the model $\texttt{LDA}_\texttt{d}$ is updated at each step. *A priori*, there is no chance that the means remain constant when the likelihood function $\mathcal{L}$ varies. We show the results for experiments where we set the number of topics to 10, knowing that similar patterns appear for different values of this parameter.

Considering the task of topic modeling, we compare the performance of AWILDA to the online version of LDA [Hoffman et al., 2010], denoted by OnlineLDA, given its capacity to handle document collections arriving in a stream. OnlineLDA is considered to process documents one by one as they are received and updates the underlying model at each step.

10.5.1.1 Results for Topic Drift Detection

Comparison of AWILDA and its variants. In the first set of experiments, we compare the performance of AWILDA and its variants when performing the task of topic drift detection on the synthetic dataset SD_4. The results are presented in Figure 10.1. The perplexity represented is related to the measure we track for drift detection. The LDA model used to compute perplexity is learned and updated differently depending on the method considered. We represent the perplexity as a moving average with a sliding window of 100 observations. The exact occurrence of drifts is marked by a green dashed vertical line and the detection of drifts is marked by a blue dotted vertical line.



(A) Results for SD_4

(B) Results for SD_9

(C) Results for REUTERS_1

(D) Results for REUTERS_4

FIGURE 10.2: Topic drift detection using `AWILDA` for synthetic and semi-synthetic datasets

`AWILDA` and `AWILDA2` only detect true positive drifts, while `AWILDA3` and `AWILDA4` detect false and true positive drifts. `AWILDA` is also more reactive than `AWILDA2` and spots drifts faster. Updating the $\texttt{LDA}_\texttt{d}$ model with each received document in `AWILDA3` and `AWILDA4` modifies the underlying distribution of topics, leading ADWIN to detect false positive drifts. `AWILDA` performs best for all the studied datasets, and we present in the following results for the other datasets.

Performance of `AWILDA` on all the datasets. As shown in Figure 10.2, `AWILDA` is able to detect all the drifts occurring in the datasets SD_4, SD_9, and REUTERS_1 after receiving only a few observations from the new distribution. Concerning the REUTERS_4 dataset, our approach spots two drifts and misses the two others. We note that in this particular dataset, we switch from one topic to another relatively fast, i.e., around 500 documents per category. Topics in articles can also be interconnected which makes the task even more complicated.

Discussion. We notice that the observed properties of AWILDA and its variants are close to the predictions which were given by Theorems 10.1 and 10.2. In particular, it has been shown that `AWILDA` and `AWILDA2` perform better than `AWILDA3` and `AWILDA4` with regards to false positives.

The superiority of `AWILDA` over the other variants raises interesting questions. It is noticeable that the best algorithm in terms of drift detection is also the only one which detection model is actively updated at each drift and not passively at each step or for each observation. This non-updating property is of particular interest: It illustrates the idea

that good drift detection does not require to have good modeling properties, which may be counter-intuitive in a way. The extreme case, `AWILDA2`, also shows rather good performance while the model is not updated at all, which means that it does not encode any information relative to the underlying distribution. A random LDA model could also work for this task. The false-negative error rate might be affected though, which can be observed even here: If the detection model is too different from the actual model, there is a chance that the likelihood change (when the underlying model varies) may not be important enough to be detected.

10.5.1.2 Results for Document Modeling

Comparing `AWILDA` with `OnlineLDA`. Figure 10.3 shows the perplexity measured on the document streams of Reuters$_1$, Ml-100k, and Plista for `AWILDA` and `OnlineLDA`. The perplexity is represented as a moving average with a sliding window of 100 observations for Reuters$_1$ and 200 observations for Ml-100k and Plista. `AWILDA` detects the existing drift for Reuters$_1$, two drifts for Ml-100k, and five drifts for Plista. This difference in behavior is expected knowing the volume and nature of the datasets, e.g., movies vs. news.

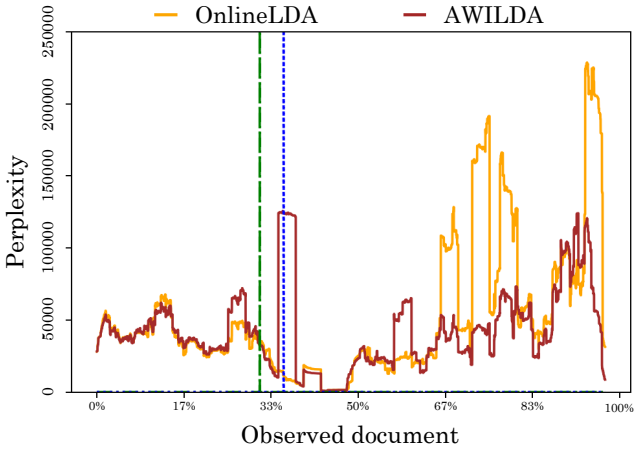
Before detecting any drift, `OnlineLDA` and `AWILDA` are trained in the same way and on the same data, which explains the close values of perplexity. After detecting the first drift, `AWILDA` outperforms `OnlineLDA` for the task of document modeling. As documents continue to arrive, `AWILDA` is more adapted to the new data. Its drift detection component allows it to adjust to changes after each drift, resulting in a better performance. Further analysis of the datasets with experts from respective domains will help to establish the link between detected drifts and real-life events occurring in the same time period, for better understanding and explainability.

10.5.2 Performance of CoAWILDA for Online Recommendation

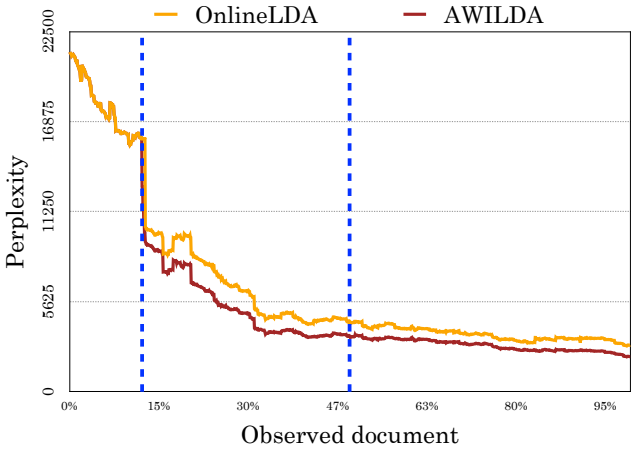
Evaluation protocol and measures. In order to evaluate CoAWILDA for the task of online recommendation on Ml-100k and Plista, we rely on the protocol introduced in Section 7.4.4. We use recall@N and DCG@N to measure the quality of recommendation. We report the results for the *Stream Test and Train* subset.

Parameters. We performed a grid search over the parameter space of the methods in order to find the parameters that give the best performance. We report the performance corresponding to the parameters leading to the best results. The parameters are provided along with the methods below.

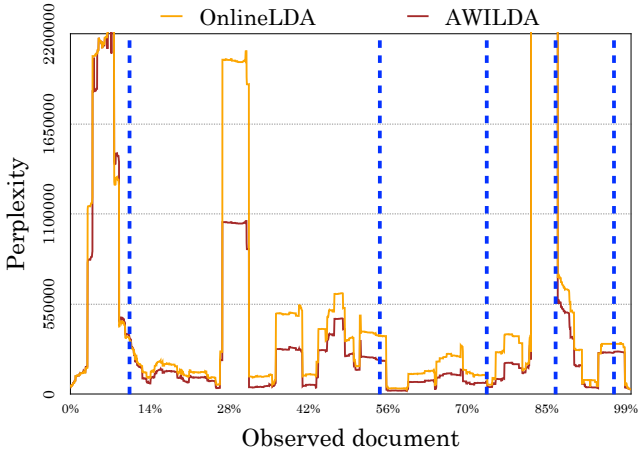
Methods compared. Since previous work has demonstrated the advantages of using online recommendation compared to batch recommendation [Frigó et al., 2017; Vinagre et al., 2014b], we focus on incremental methods for comparison. We also only consider one approach for IMF knowing that our method, CoAWILDA, can integrate any other algorithm for IMF or any model-based method. We compare the performances of several incremental methods adapted to the online setting, including variants of the one we propose, mentioned in the following:



(A) Results for REUTERS$_1$



(B) Results for ML-100K



(C) Results for PLISTA

FIGURE 10.3: Performance evaluation of `AWILDA` and `OnlineLDA` for the task of document modeling using the measure of perplexity

- `CoAWILDA` is the method we propose, combining ADaptive Window based Incremental LDA (AWILDA) for topic modeling and Incremental MF (IMF) for CF. For ML-100K, we set the number of topics $K = 20$, $\eta = 0.04$, $\lambda_P = 0.01$, and $\lambda_Q = 0.1$. For PLISTA, we set $K = 10$, $\eta = 0.042$, $\lambda_P = 0.01$, and $\lambda_Q = 0.1$.

- `CoLDA` relies on classical online LDA [Hoffman et al., 2010] for topic modeling and Incremental MF (IMF) for CF. It replaces AWILDA from CoAWILDA with classical online LDA. For ML-100K, we set $K = 20$, $\eta = 0.05$, $\lambda_P = 0.01$, and $\lambda_Q = 0.1$. For PLISTA, we set $K = 10$, $\eta = 0.045$, $\lambda_P = 0.01$, and $\lambda_Q = 0.1$.

- `AWILDA` denotes the method we propose for adaptive topic modeling. We try to use it for recommendation without the collaborative component by representing users in the space of topics and updating their profiles as we get more observations. This is equivalent to setting the elements of ϵ, from Section 10.4, to zero. For ML-100K, we set $K = 20$, $\eta = 0.04$, and $\lambda_P = 0.01$. For PLISTA, we set $K = 10$, $\eta = 0.042$, and $\lambda_P = 0.01$.

- `MF` denotes the Incremental MF (IMF) [Vinagre et al., 2014b]. Compared to `CoAWILDA` and `CoLDA`, `MF` does not leverage content information about items. For ML-100K, we set $K = 50$, $\eta = 0.01$, $\lambda_P = 0.02$, and $\lambda_Q = 0.02$. For PLISTA, we set $K = 50$, $\eta = 0.008$, $\lambda_P = 0.01$, and $\lambda_Q = 0.01$.

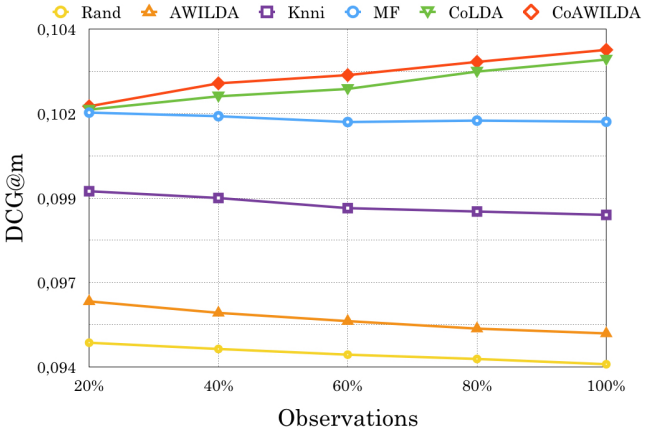
- `Knni` is the incremental item-based approach proposed in [Miranda and Jorge, 2009] (Section 7.4.2.1) and using the cosine similarity. We set the number of neighbors to 300.

- `Rand` randomly selects items for recommendation.

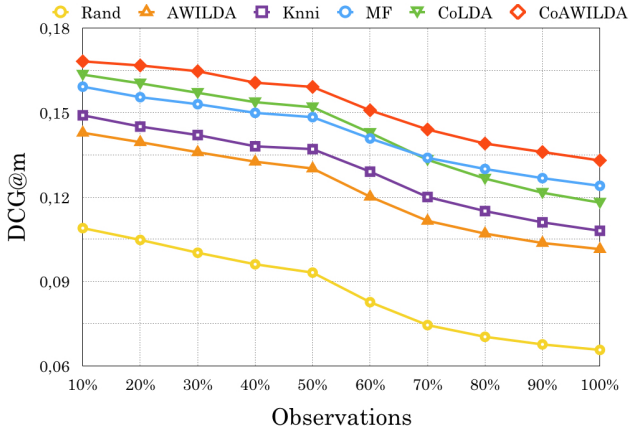
Results. Figure 10.4 shows the DCG@m of the methods we compare for ML-100K and PLISTA, where m is the total number of items included in each dataset. The idea is to evaluate how each approach performs when ranking the items for each user. We report the metric value with respect to the number of processed observations in order to analyze its evolution over the time spanned by the *Stream Test and Train* subset.

`CoAWILDA` outperforms all the other methods evaluated for both datasets. The comparison between `CoAWILDA` and `CoLDA` demonstrates the effectiveness of AWILDA for modeling document streams describing new items and for improving the quality of item modeling and thus recommendation. `CoLDA` is not able to adjust to drifts occurring in topic modeling which deteriorates the recommendation quality over time.

The performance of `CoLDA` for PLISTA can be divided into two phases. In the first one, the topic model is still able to carry out good document modeling and is beneficial for recommendation: `CoLDA` performs better in terms of item ranking than `MF` which does not account for content analysis. In the second phase (after having processed 70% of observations), and with the incapacity of online LDA to adjust to drifts, `MF` outperforms `CoLDA`. This means that not only the topic model is not adapted to newly received data, but it is also badly affecting the recommendation quality and there is no interest in using it anymore. We also note the importance of evaluating the performance of the models over time to show how they are affected by eventual changes occurring in the data. This phenomenon appears for



(A) Results for ML-100K



(B) Results for PLISTA

FIGURE 10.4: DCG@m of `CoAWILDA` and other variants and incremental methods, where m is the number of available items. The evolution of DCG@m with the number of evaluated observations is reported.

PLISTA where drifts occur more frequently over time, mainly due to the nature of news data. Concerning ML-100K, `CoLDA` performs better than `MF` but still worse than `CoAWILDA`. `AWILDA` is a content-based method and only relies on topics extracted from items to model user preferences. It performs poorly compared to the other methods and proves the importance of having a CF component. `Knni` performs better than `AWILDA` but is not as robust as `MF` and the other hybrid approaches evaluated.

The number of available items grows significantly over time in PLISTA. This results in the dropping of performances of all methods in terms of ranking over time. This is not the case in ML-100K given that only a few movies are added to the set of available items in the time period considered. More data is received and more learning is done over time, which can explain the improvements in the performances of `CoAWILDA` and `CoLDA`.

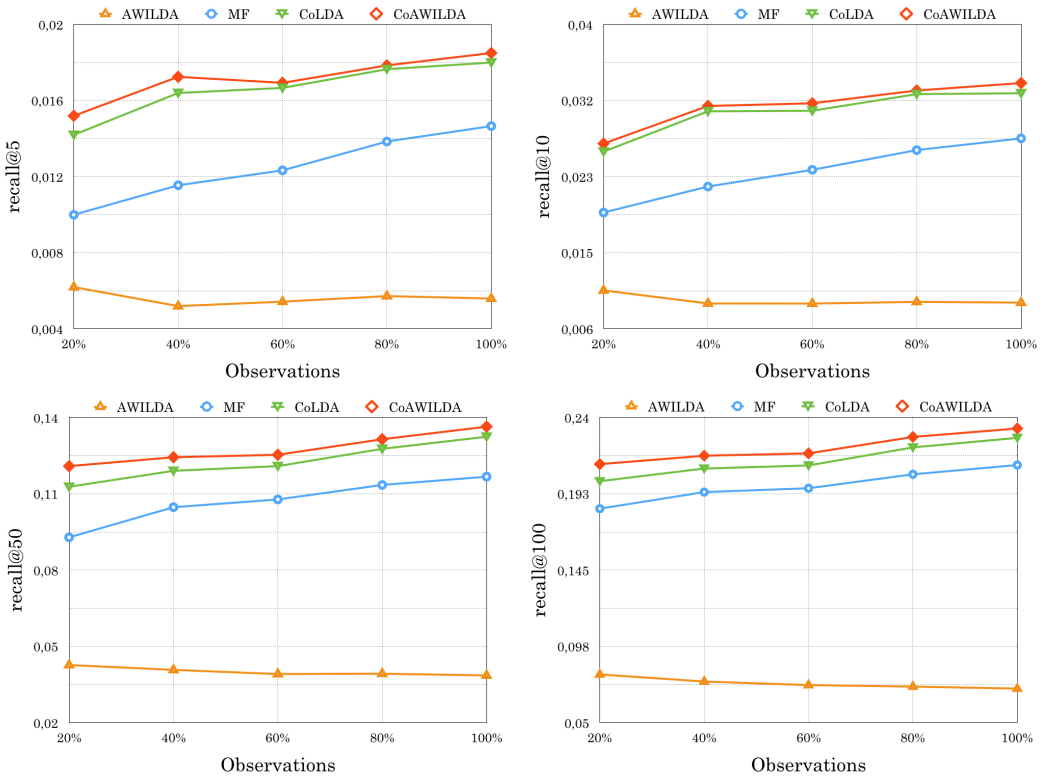


FIGURE 10.5: Recall@5, recall@10, recall@50, and recall@100 of `CoAWILDA` and its variants for the dataset ML-100K

Figure 10.5 shows the recall@5, recall@10, recall@50, and recall@100 of `CoAWILDA` and its variants on ML-100K. The experiments confirm the ideas we mentioned before. `CoAWILDA` outperforms the other variants and performs better than `CoLDA` which relies on online LDA and does not adapt to changes in the data. `CoLDA` performs better than `MF` demonstrating the benefits of using content information. `AWILDA` relies only on content information which is a weak approach to model user preferences.

Figure 10.6 shows the recall@5, recall@10, recall@50, and recall@100 of `CoAWILDA` and its variants on PLISTA. The experiments highlight an interesting behavior. For recall@5 and recall@10, `MF` performs better than `CoLDA` for all the observations considered. For recall@50 and recall@100, we observe two different behaviors where, first, `CoLDA` performs better than `MF`, and then `MF` outperforms `CoLDA`. We recall that the reported results are measured on the second half of the dataset (*Stream Test and Train* subset). Drifts may have occurred during the training phase, which is typically the case for PLISTA. When measuring the recall@N, `CoLDA` is already weakened by the drifts that have happened and that were not taken into account. This leads to a point where the information learned by the topic model hurts the recommendation quality, and `MF` starts performing better than `CoLDA`. This change of behavior occurs at different points in time depending on the recall we are measuring. For higher values of N, i.e., recall@50 and recall@100, the performance of `CoLDA` remains superior to the performance of `MF` for a longer time than for lower values of N, i.e., recall@5 and recall@10. Top list recommendation is thus more affected by the deterioration of the topic model.

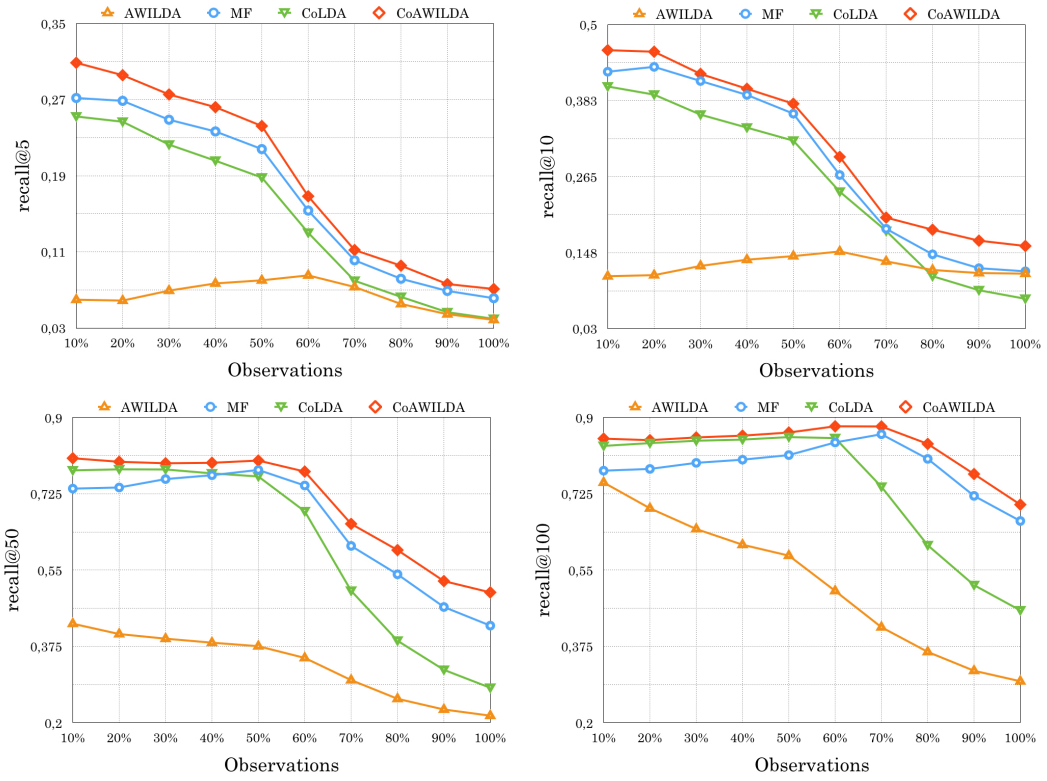


FIGURE 10.6: Recall@5, recall@10, recall@50, and recall@100 of `CoAWILDA` and its variants for the dataset PLISTA

All experiments demonstrate the effectiveness of using `CoAWILDA`, the strength of which relies on adapting to changes occurring in the data. Methods that do not detect and adapt to these changes (`CoLDA` in particular) perform worse than `CoAWILDA`.

10.6 Conclusion

In this chapter, we address the problem of online hybrid recommendation in dynamic environments and we tackle several subjects at once: hybrid approach for recommendation (merging CF and CBF), recommendation with concept drifts, and collaborative topic modeling. The solution we propose, which we call CoAWILDA, is designed for online recommendation where textual descriptions of items are provided as new items are becoming available. It combines the advantages of the drift detection method ADWIN, the flexibility of online LDA, and the online nature of Incremental Matrix Factorization (IMF). In the proposed setting, user interactions arrive in a stream and the method adapts to new items becoming available and to user interactions. Leveraging the advantages of both document analysis and users' interactions, CoAWILDA is particularly suitable to alleviate the problem of cold start and offers an elegant and generic solution to deal with drifting item distributions. Since very few training steps are required, the algorithm is truly online and can run in real-time.

An experimental validation has shown the actual efficiency of the proposed methodology. CoAWILDA outperforms its variants and other incremental methods we evaluated. In particular, it has been shown that a topic model which is not adjusting to drifts can hurt the quality of recommendation to an extent where its removal results in a much better performance. For that matter, recommendation at the top of the list is firstly affected.

The CoAWILDA framework is well-designed to handle abrupt drifts and can also work when gradual drifts occur. In some cases where the data distribution changes over time, previous states can reoccur and recurrent drifts can happen. Handling such re-occurrences requires to manage a memory of past states in order to benefit from previous observations corresponding to the same distribution [Al-Ghossein et al., 2018f]. Future work will focus on extending CoAWILDA to handle such types of drifts.

Part IV

Concluding Remarks

Chapter 11

Conclusions and Future Work

This chapter briefly summarizes the work presented in this thesis and indicates directions for future work.

11.1 Summary and Conclusions

Overall, this thesis focuses on the problem of non-stationarity in Recommender Systems (RS). Initially, in quest of offering relevant suggestions, RS try to uncover user preferences for items based on past user behavior, given that it exhibits user interests. Along this line, a countless number of works have proposed recommendation approaches that learn preferences based on user interactions and model users and items in a common space. These approaches assume that user interests and item perceptions remain static over time and in all circumstances.

While it is true that analyzing user behavior gives insights about preferences for items, the environment delimited by a RS is complex and dynamic, and involves several entities that evolve each on its own and in a unique way. Dynamics occurring within users and items can be explained by the presence of varying contextual factors impacting the environment. Previous work valued the relevance of context in RS and proposed Context-Aware RS (CARS). These systems integrate contextual factors for a better recommendation quality. In this thesis, we discuss the limitations of conventional CARS in the scope of two real-world applications: hotel recommendation and online recommendation. While the traditional view of CARS considers *fully observable* context, we argue that context is *partially observable* in the hotel domain and we study *unobservable* context in the online setting, requiring the development of novel appropriate approaches.

At the broadest level, the contributions of this thesis fall into one of three following categories:

Analysis. We thoroughly analyze and present the two problems addressed in this thesis, which are the hotel recommendation (Chapter 3) and the online recommendation (Chapter 7) problems. We formulate the hotel recommendation problem, discuss its relation with

other well-known recommendation problems, and present its characteristics and specific challenges that make the use of traditional recommendation approaches insufficient. We also provide a description of the notion of context as emerging in the hotel domain and discuss why it qualifies as *partially observable*, following our definition in Chapter 1.

On the other hand, after highlighting the importance of considering the online setting, we formulate the problem of online adaptive recommendation based on a framework inspired by adaptive learning in data stream mining. We discuss the limitations of the relatively few existing approaches that attempt to account for drifts occurring due to *unobservable* context in online RS. We explain how we adopt a substantially different direction to address the problem.

Algorithms. We design novel context-aware approaches and algorithms that take into account the dynamics existing in user interactions. Considering hotel recommendation, we propose recommendation approaches that integrate contextual factors affecting the user's decision-making process. These factors include the geography and temporal dimensions, textual reviews extracted from social media, the trips' intents, planned events, and users' mobility. Some of these factors are explicitly provided by the user while others are only collected by the system, and some of them are observed in auxiliary domains different than the hotel domain. The proposed approaches allow the incorporation of data related to the user's environment into hotel RS in order to improve the recommendation quality.

Considering online recommendation, we propose novel online algorithms that account for drifts in user preferences, item perceptions, and item descriptions. The novelty of our approaches rely on actively detecting drifts and on dynamically updating models according to the observed drifts, rather than only passively updating models in a constant incremental fashion. Our approaches allow the continuous integration of user-generated feedback and maintaining recommendation models up to date in real-time.

Lessons learned. We evaluate the proposed approaches on real-world datasets, compare them to existing approaches, and derive conclusions with regards to the performance of methods and the benefits of taking into account the dynamic aspect of user-generated data. Experiments performed in the scope of hotel recommendation show the interest of using contextual information to improve the recommendation quality. They further highlight the sensitivity of different user subsets to the considered contextual factors.

Experiments performed in the scope of online recommendation show the importance of actively accounting for different types of drifts occurring at the user and item levels. Models are maintained adapted to the current states of users and items, and the recommendation quality improves compared to other incremental approaches. Results also demonstrate how conventional recommendation approaches that learn at a constant pace fail to adapt to drifts and contribute in deteriorating the RS performance.

We now conclude the thesis by presenting open challenges and directions for future research, related to each one of the studied problems.

11.2 Future Work

There are numerous possible directions for future work, some of which we mention in the following.

Hotel recommendation. The problem of hotel recommendation can first benefit from the integration of additional contextual factors that affect users and that were not considered in this thesis. We mention in particular the impact of the weather forecast on travelers [Trattner et al., 2018]. The attractivity of destinations is known to change under different weather conditions, and users often check the weather forecast when organizing a trip. While considering temporality and seasonality is a first step towards addressing this problem, it is not a sufficient solution. This is especially true given that the weather is not constant during one season and that it is not the same during several occurrences of the same season. Integrating the weather forecast into hotel RS could benefit hotel recommendation but requires an appropriate framing of the weather dimension in addition to the design of appropriate methods.

Another variable that could benefit hotel recommendation and that was not studied in this thesis concerns the impact of prices on hotel selection [Sanchez-Vazquez et al., 2017]. We considered a number of factors that are expected to implicitly affect prices, e.g., brands and seasonality, but not the price variable itself. Compared to other domains, prices in the hotel domain are particularly characterized by their dynamic aspect where the same room is priced differently each day of the year. An interesting direction would consider the impact of prices on user behavior and would incorporate this dimension into hotel RS. Considering all the relevant factors in one RS also requires further developments that would combine the benefits acquired by each one of them.

While the main goal of hotel RS is to offer relevant recommendations for users, other interesting use cases can be considered from a business point of view. First, users are usually confronted with hotels and accommodations owned by several companies during the selection process. The presence of competitors in the considered geographical region affects the decision-making process. Hotel companies can adjust their recommendation strategy according to information about competitors in each region to attract more travelers. On the other hand, when implemented in hotel organizations, RS can be used as an indicator of preferences that could be integrated into the process of dynamically setting room prices.

Online recommendation. As mentioned in this thesis, online adaptive recommendation remains an underexplored problem albeit its relevance in real-world applications. It is becoming essential to adopt the online setting for recommendation, especially given that it would invalidate several recommendation techniques that perform well in batch. While we prove in this thesis the importance of actively detecting and adapting to drifts in RS in real-time, there are many directions for improving online RS that we present below.

The change detection module. While the work presented in this thesis considers incremental and abrupt drifts, recurrent drifts are often appearing in online RS. Users can be affected by recurring factors derived from seasonality variables for example. Handling such drifts allows the RS to benefit from previous observations corresponding to the same concepts. It requires

managing a memory of past states in order to leverage previously acquired knowledge [Al-Ghossein et al., 2018f]. Future work should consider developing online recommendation approaches that account for these types of drifts. Specific methods should also be designed to handle all types of drifts occurring on the user and item levels at once and in one framework.

The retrieval module. The problem of efficiently retrieving recommendations has emerged in recent work, especially given that real-world systems handle large numbers of users and items. In these settings, and considering the Matrix Factorization (MF) framework, constructing the entire feedback matrix given a learned model requires heavy computations and resources. Therefore, a number of works have proposed solutions to this problem [Koenigstein and Koren, 2013; Koenigstein et al., 2012], but the focus has been on efficiently retrieving recommendations once the model is trained. These approaches cannot be applied for online RS where models are constantly being updated. Nevertheless, adopting similar strategies is essential in the online setting that is subject to limitations in terms of resources and computation time. Future work should consider the development of efficient methodologies to retrieve recommendations in an online setting where models are constantly updated.

The evaluation module. The prequential evaluation faces several limitations in the context of online recommendation [Vinagre et al., 2014a]. In particular, recommendations are only evaluated against the single received observation. Relevant recommended items that were selected in the near past or that will be selected in the near future can be penalized. In addition, previous work related to recommendation in online environments only considers the evaluation of the accuracy and ranking of recommendations, e.g., using recall and DCG. Metrics related to other criteria such as diversity and novelty have to be adapted to the online setting and evaluated for existing online recommendation approaches. In addition, the evolution of metrics over time in a sequential setting has to be assessed. The feedback received from these evaluations could also be exploited to adapt recommendations in real-time. Future work should consider solutions to the limitations faced by prequential evaluation and additional metrics for the evaluation of online RS.

Connections with other areas in RS. It could be interesting to investigate connections between online adaptive recommendation and other areas in RS. In particular, one promising direction would consider the design of cross-domain online adaptive RS. Cross-domain RS are known for their ability to enrich a target domain by transferring knowledge from auxiliary domains [Cremonesi et al., 2011]. Knowledge acquired in real-time could then benefit the target domain and could help in anticipating events or drifts. Further advances could consider the design of such methods and the evaluation of their performances.

Overall, the work presented in this thesis highlights the importance of dynamics existing in the ecosystem of RS and the benefits of taking them into account. It also underlines that deploying RS in real-world settings is subject to specific constraints that should be clearly identified and considered when developing new approaches. Despite all the advances that have been made in the field, the recommendation problem is still rich of challenging real-world problems, bringing on exciting new directions for research.

Annexe A

Résumé en français

Au début des années 1960, un certain nombre de travaux liés à la diffusion sélective de l'information ont vu le jour. Des systèmes intelligents furent conçus pour filtrer les flux de documents électroniques en fonction des préférences individuelles [Hensley, 1963]. Ces systèmes utilisent des informations explicites relatives aux documents, telles que les mots-clés, pour effectuer le filtrage. Au cours des décennies suivantes, avec l'utilisation croissante des courriels et dans le but de contrôler le flot d'informations et de filtrer les spams, le système de filtrage de courriel Tapestry [Goldberg et al., 1992] fut développé. La nouveauté de Tapestry reposait sur l'exploitation des opinions exprimées par l'ensemble des utilisateurs afin de bénéficier chacun d'eux individuellement, ce qui s'est avéré être une approche prometteuse. Ces premiers efforts ont ouvert la voie à une catégorie de systèmes qui ont été ultérieurement dénommés par *systèmes de recommandation (RS)* [Resnick and Varian, 1997].

Dans leur forme la plus générale, les RS sont utilisés pour recommander des articles de différents types aux utilisateurs, en filtrant et sélectionnant ceux qui sont les plus pertinents. Ces articles peuvent être des produits, des livres, des films, des articles de presse ou des amis, entre autres. Des études de marketing mettant en évidence les multiples avantages de la recommandation de produits accompagnèrent les premières avancées techniques liées aux RS [West et al., 1999]. Le principal avantage consiste à aider les utilisateurs à surmonter la surcharge informationnelle dans les domaines où le catalogue d'articles est énorme, améliorant ainsi l'expérience et la satisfaction des utilisateurs. La promesse de renforcement de la fidélité des utilisateurs et de l'augmentation du volume des ventes attira les services en ligne qui se sont empressés de mettre en place des RS pour améliorer leurs performances. *Amazon.com* fut l'un des premiers à adopter les RS et reporta l'impact considérable qu'ont ces systèmes sur son activité [Linden et al., 2003]. Cette réussite motiva de nombreux acteurs à appliquer ces concepts dans leurs domaines respectifs [Bennett and Lanning, 2007; Celma, 2010; Mooney and Roy, 2000], ce qui alimenta la recherche dans le domaine.

Depuis, la communauté scientifique développe des idées et des techniques pour les RS et combine des travaux multidisciplinaires extraits de divers domaines voisins, tels que l'intelligence artificielle, l'exploration de données et l'interaction homme-machine. Toutefois, l'élément fondamental d'un RS est l'algorithme de recommandation qui estime la pertinence des articles pour chaque utilisateur et peut être considéré comme effectuant une tâche

de prédiction [Adomavicius and Tuzhilin, 2005]. Cette prédiction de pertinence d'articles passe par l'analyse et l'exploitation du comportement passé des utilisateurs, collecté sous plusieurs formes et supposé manifester leurs préférences.

Non-stationnarité dans les RS. En examinant le problème général d'apprentissage à partir de données et de mise en place de modèles prédictifs [Mitchell, 1997], la principale hypothèse faite est que les observations qui seront générées à l'avenir suivent des motifs similaires à celles précédemment collectées dans un même environnement. En d'autres termes, toutes les observations sont supposées être générées par une même distribution. Dans le contexte d'un RS, cela signifie que le comportement passé des utilisateurs permet de prévoir leur comportement futur du moment que les préférences des utilisateurs et les perceptions des articles restent constantes avec le temps. Dans un monde dynamique où les utilisateurs et les articles évoluent constamment et sont influencés par de nombreux facteurs [Koren, 2009], cette hypothèse ne tient pas.

Exemple illustratif. Imaginez Alice organisant son voyage touristique à Paris. Ceci ne sera pas sa première visite à Paris ; elle s'y était déjà rendue lors de plusieurs voyages d'affaires. Alors qu'elle réserve d'habitude un hôtel près des bureaux de son entreprise, elle préférerait cette fois-ci un hôtel situé au centre-ville afin de se rapprocher des nombreux sites touristiques. Alice aime généralement se promener dans les rues et admirer les beaux bâtiments. Cependant, comme la météo prévoit de fortes pluies pour la période de son séjour, elle finira probablement par faire le tour des musées. Vendredi soir, elle rencontre Carl, un ancien ami à elle. Leur rituel au cours des dernières années consistait à manger des sandwichs et à traîner dans un parc. Maintenant qu'ils ont tous les deux des emplois bien rémunérés, ils pourront se permettre de dîner dans un restaurant gastronomique. Imaginez maintenant l'assistant qui a pour rôle d'aider Alice dans l'organisation de son voyage. En se basant sur son comportement passé, l'assistant lui recommanderait probablement un hôtel vers *La Défense*, une promenade à pied sur les *Champs-Élysées* sous la pluie et un camion-restaurant offrant des hot-dogs près du *jardin des Tuileries*. Un assistant idéal est supposé prendre en compte les *facteurs contextuels* influençant les préférences d'Alice tels que la raison de son voyage, son statut social et la météo, afin de pouvoir offrir des recommandations pertinentes.

RS contextuels (CARS). L'importance du contexte de l'activité humaine et de son influence sur les préférences des individus [Suchman, 1987] a conduit au développement des *RS contextuels (CARS)* [Adomavicius and Tuzhilin, 2015]. Ces RS incorporent des informations relatives au contexte et offrent des recommandations adaptées aux circonstances spécifiques des utilisateurs. Le développement des CARS a commencé au début des années 2000 avec une série de travaux montrant l'intérêt d'utiliser les informations contextuelles et les exploitant dans plusieurs domaines [Hayes and Cunningham, 2004; Van Setten et al., 2004]. De nos jours, les CARS couvrent un large éventail de paradigmes et de techniques. Malgré les multiples classifications existantes, les CARS prennent en compte, sous toutes leurs formes, le caractère dynamique des données générées par les utilisateurs. Les principaux défis relatifs à la mise en place de CARS concernent, d'une part, la compréhension et la modélisation de la notion de contexte et, d'autre part, le développement d'algorithmes intégrant cette notion. Sachant que le deuxième défi dépend fortement du premier, plusieurs limitations liées à la compréhension et à la modélisation du contexte persistent dans les solutions utilisées actuellement.

A.1 La notion de contexte dans les systèmes de recommandation du monde réel

Le contexte est une notion complexe qui fut étudiée dans différentes disciplines de recherche [Adomavicius and Tuzhilin, 2015]. La définition introduite par [Dey, 2001] a été largement adoptée pour les CARS et énonce ce qui suit : « *Le contexte est toute information pouvant être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un lieu ou un objet considéré comme pertinent dans le cadre d'une interaction entre un utilisateur et une application.* ». Sans perte de généralité, la notion de contexte peut être représentée par un ensemble de facteurs contextuels pertinents étant donné une application spécifique. Plusieurs efforts ont été réalisés afin de modéliser et de représenter ces facteurs dans le cadre des CARS. Cependant, la plupart des CARS proposés dans la littérature adoptent des concepts relativement semblables et présentés dans ce qui suit.

La définition du contexte. La vue *représentationnelle* du contexte, introduite par [Dourish, 2004], est l'approche standard utilisée pour définir la notion de contexte dans les CARS. Elle suppose que les facteurs contextuels sont représentés par un ensemble prédéfini d'attributs observables qui ont une structure statique et des valeurs connues *a priori*. Par opposition, la vue *interactionnelle* définit le contexte comme une propriété relationnelle établie entre les activités et déterminée dynamiquement. Il n'est donc pas possible d'énumérer les facteurs contextuels pertinents avant que l'activité de l'utilisateur ne se produise.

L'intégration du contexte. Même si les facteurs contextuels peuvent avoir des types différents, il est souvent supposé que les attributs sont des variables catégorielles et atomiques, par exemple, « *famille* » ou « *collègues* » pour le facteur *compagnie* et « *été* » ou « *hiver* » pour le facteur *saison*. Deux modèles de données ont été largement utilisés afin de représenter le contexte avec les utilisateurs et les articles dans les CARS : le modèle *hiérarchique* et le modèle d'*espace de données multidimensionnel*. Suivant le modèle hiérarchique [Palmisano et al., 2008], le contexte est modélisé comme un ensemble de facteurs contextuels et chaque facteur comme un ensemble d'attributs ayant une structure hiérarchique. Chaque niveau de la hiérarchie définit un niveau de granularité différent relatif à la connaissance du contexte. D'autre part, le modèle d'espace de données multidimensionnel [Adomavicius et al., 2005] représente le contexte par le produit cartésien des facteurs contextuels, chaque facteur étant à nouveau le produit cartésien d'un ou de plusieurs attributs. Chaque situation est décrite par la combinaison des valeurs d'attributs pour tous les attributs de tous les facteurs.

Limitations dans les RS du monde réel. Bien qu'elle soit largement adoptée dans les CARS, la vue représentationnelle du contexte ne permet pas de résoudre plusieurs problèmes rencontrés dans les applications du monde réel et qui restent inexplorés. De plus, il existe un écart entre la notion de contexte traditionnellement modélisée dans les CARS et celle émergeante dans les RS du monde réel, ce qui rend les approches précédemment proposées pour les CARS insuffisantes. Cet écart survient à plusieurs niveaux et est décrit dans ce qui suit :

- **Accessibilité.** La vue représentationnelle du contexte suppose que le système observe explicitement les facteurs contextuels affectant le comportement de l'utilisateur. Ceci n'est naturellement pas toujours le cas, en particulier lorsqu'il s'agit de facteurs difficilement accessibles, tels que l'humeur ou l'intention de l'utilisateur. De plus, dans certains cas, le contexte n'est révélé qu'une fois l'action réalisée, et donc ni avant ni indépendamment de son occurrence. Une classification d'approches pour la représentation du contexte allant au-delà des vues représentationnelle et interactionnelle a été présentée dans [Adomavicius et al., 2011] et est basée sur l'aspect d'observabilité, à savoir ce que le RS sait sur les facteurs contextuels, leur structure et leurs valeurs. Ces connaissances appartiennent à l'une des trois catégories suivantes : *intégralement observable*, *partiellement observable* et *non observable*. La vue représentationnelle du contexte considère le cas où le contexte est entièrement observable, alors que les deux autres cas n'ont pas été vraiment étudiés dans la littérature.

- **Pertinence.** Le contexte est modélisé comme une variable multidimensionnelle où toutes les dimensions, désignant les multiples facteurs, sont traitées de manière égale. Comme les méthodes de recommandation génériques n'intègrent pas une connaissance liée du domaine, tous les facteurs contextuels sont supposés affecter le comportement de l'utilisateur de la même manière. En réalité, cela n'est pas toujours vrai et les utilisateurs ont tendance à privilégier certains facteurs par rapport à d'autres, conduisant à des cas où les facteurs ne contribuent pas de manière égale au processus de prise de décision.

- **Acquisition.** Traditionnellement, le contexte peut être acquis *explicitement*, *implicitement* ou par *inférence* [Adomavicius and Tuzhilin, 2015]. Un contexte explicite (par exemple, l'humeur de l'utilisateur) est fourni par l'utilisateur et un contexte implicite (par exemple, la date et la localisation) est collecté par le système et ne nécessite aucune action de la part de l'utilisateur. Le contexte peut également être inféré à l'aide de méthodes statistiques ou d'exploration de données. En réalité, le contexte apparaît fréquemment dans des domaines autres que le domaine cible, c'est-à-dire le domaine dans lequel la recommandation est effectuée. Identifier le contexte de l'action n'est alors pas trivial. Cela nécessite d'établir des liens entre les domaines et de transférer les connaissances de l'un à l'autre.

- **Modélisation.** La grande majorité des CARS considèrent les attributs des facteurs contextuels comme des variables catégorielles et atomiques. Toutefois, cela n'est pas toujours raisonnable. Le contexte peut se présenter sous des formes complexes et peut inclure des données non structurées, et un transtypage direct peut donc mener à la perte d'informations de valeur. De plus, l'ensemble des valeurs d'attributs peut être inconnu à l'avance, étant donné que de nouveaux événements peuvent être accompagnés de nouvelles valeurs contextuelles.

La définition du contexte dans le cadre des vues représentationnelle et interactionnelle ne couvre pas les limitations qui sont mentionnées ci-dessus et qui apparaissent dans les applications du monde réel. Par conséquent, nous nous basons sur les définitions du contexte *partiellement observable* et *non observable*, initialement proposées dans [Adomavicius et al., 2011], que nous étendons pour prendre en compte les contraintes du monde réel.

Contexte partiellement observable. Le contexte est considéré partiellement observable dans les cas où seules certaines informations relatives aux facteurs contextuels sont explicitement connues tandis que de les autres sont manquantes. Cette notion d'observabilité partielle peut toucher un ou plusieurs des niveaux définis précédemment, comme expliqué ci-dessous :

- **Accessibilité.** Le contexte est considéré partiellement observable dans les cas où certains facteurs contextuels sont non observables ou inaccessibles. Ceci inclut également les cas où, étant donné un facteur contextuel, la disponibilité des valeurs des attributs de ce facteur est retardée : le contexte n'est pas disponible au moment de la recommandation, mais une fois l'interaction terminée.

- **Pertinence.** Le contexte est considéré partiellement observable dans les cas où l'information concernant les pertinences relatives des facteurs contextuels n'est pas disponible. Ceci inclut également les cas où tous les facteurs contextuels ne sont pas également importants pour tous les utilisateurs et où cette information est manquante.

- **Acquisition.** Le contexte est considéré partiellement observable dans les cas où il est observé dans un domaine autre que le domaine cible. Les connaissances liées au contexte doivent alors être transférées afin d'être exploitées dans le domaine cible où la recommandation est effectuée. Ainsi, le lien direct entre le contexte et les entités d'un RS, c'est-à-dire les utilisateurs et les articles, est absent et doit alors être établi.

- **Modélisation.** Le contexte est considéré partiellement observable dans les cas où la structure de certains facteurs contextuels est inconnue. Ceci inclut également les cas où l'ensemble des valeurs possibles pour les attributs des facteurs contextuels n'est pas connu *a priori*.

Contexte non observable. D'autre part, le contexte est considéré non observable lorsque les informations correspondantes sont totalement inconnues ou inaccessibles. Ce problème se pose lors du traitement de certains facteurs comme l'humeur ou l'intention de l'utilisateur. Par conséquent, le contexte ne peut pas être modélisé explicitement et exploité par la suite pour la recommandation. Néanmoins, ce contexte non observable cause l'émergence de dynamiques temporelles dans les données générées par les utilisateurs et devrait être pris en compte afin de proposer des recommandations pertinentes. Par exemple, et suivant notre définition, les RS sensibles à la dimension temporelle [Campos et al., 2014] peuvent être considérés comme gérant une sorte de contexte non observable qui influence les utilisateurs et les articles. En modélisant l'évolution des utilisateurs et des articles au cours du temps en ayant accès qu'aux comportements des utilisateurs, ils tiennent compte du contexte caché qui pousse les différentes entités à évoluer.

A.2 Contributions

Cette thèse aborde les problèmes de contextes *partiellement observable* et *non observable* dans deux applications différentes : la recommandation d'hôtels et la recommandation en ligne. D'un point de vue global, chaque problème est analysé de manière approfondie, de nouvelles approches contextuelles tenant compte de la dynamique existante au niveau des utilisateurs et des articles sont proposées, et l'impact sur la qualité de recommandation est étudié.

Recommandation d'hôtels. La recommandation d'hôtels [Zoeter, 2015] est un problème intéressant mais complexe et qui a été peu étudié. L'objectif est de recommander des hôtels que l'utilisateur appécierait, en se basant sur son comportement passé. Bien que le développement de RS soit difficile en général, le développement de ces systèmes dans le domaine hôtelier en particulier doit répondre à des contraintes spécifiques, rendant l'application directe des approches classiques insuffisante. Le processus de prise de décision lors de la sélection d'hôtels diffère radicalement de celui de l'acquisition de biens matériels et les voyageurs adoptent des comportements très différents en fonction de leur situation. De plus, les RS font régulièrement face au problème de démarrage à froid, à cause de la volatilité des préférences et du changement des comportements en fonction du contexte. Bien que les CARS représentent un moyen prometteur pour résoudre ce problème, la notion de contexte reste complexe et difficile à intégrer dans les CARS existants.

Contexte partiellement observable dans le cadre de la recommandation d'hôtels. Exploiter le contexte afin d'améliorer la recommandation d'hôtels implique l'intégration de données provenant de différentes sources. Chacune de ces sources est supposée fournir des informations différentes relatives au contexte de l'utilisateur, tout en introduisant un nombre de défis au niveau de la modélisation. Après avoir identifié les caractéristiques distinctives du problème, nous explorons l'incorporation de plusieurs facteurs contextuels considérés comme partiellement observables dans les RS d'hôtels, afin d'améliorer leurs performances. Nos principales contributions peuvent être résumées comme suit :

- **Exploitation du contexte explicite.** Nous proposons un CARS pour la recommandation d'hôtels prenant en compte les contextes physique, social et modal des utilisateurs. Nous concevons des modèles contextuels qui intègrent les dimensions géographique et temporelle, les commentaires textuels extraits des réseaux sociaux et la raison du voyage, afin de pallier les limitations liées à l'utilisation d'informations uniquement liées aux comportements passés des utilisateurs. La dénomination *contexte explicite* est utilisée pour désigner des informations contextuelles directement liées aux utilisateurs ou aux hôtels et explicitement fournies par les utilisateurs. Nous démontrons l'efficacité de l'utilisation du contexte pour améliorer la qualité des recommandations et étudions la sensibilité des utilisateurs aux différents facteurs contextuels.
- **Exploitation du contexte implicite.** Étant donné que les événements organisés constituent une motivation majeure pour les voyages, nous proposons un nouveau RS qui exploite le calendrier d'événements à venir pour effectuer la recommandation d'hôtels. Les événements en question ont une occurrence unique et sont de nature éphémère. Alors que ces événements influencent les décisions des utilisateurs, les hôtels et les événements appartiennent à deux domaines différents et il n'existe aucun lien explicite établi entre les utilisateurs et les hôtels d'un côté, et les événements de l'autre. La dénomination *contexte implicite* est utilisée pour désigner des informations contextuelles collectées implicitement par le système. Nous proposons une solution à ce problème et montrons les avantages de l'exploitation des données d'événements pour la recommandation d'hôtels.

- **Transfert d'informations contextuelles entre domaines.** Nous proposons un RS interdomaine qui exploite les check-ins partagés sur les réseaux sociaux pour apprendre des préférences géographiques et les utiliser pour la recommandation d'hôtels, étant donné que le choix de destination est important pour la sélection d'hôtels. La recommandation interdomaine est un moyen de faire face au problème de sparsité, en exploitant les connaissances d'un domaine connexe où les données peuvent être facilement collectées. Les connaissances relatives à la mobilité des utilisateurs sont acquises dans le domaine des réseaux sociaux et ensuite transférées au domaine de l'hôtellerie. Nous proposons alors une approche de recommandation interdomaine et montrons dans quels cas ces connaissances peuvent être bénéfiques pour la recommandation d'hôtels.

Suivant les définitions précédemment introduites, les facteurs contextuels pertinents pour la recommandation d'hôtels sont considérés *partiellement observables* pour plusieurs raisons détaillées dans les parties correspondantes de la thèse, et nécessitent donc la conception de nouvelles approches. Par ailleurs, nous étudions le problème du contexte *non observable* dans le cadre de la recommandation en ligne.

Recommandation en ligne. Avec l'explosion du volume de données générées par les utilisateurs, la conception de RS en ligne exploitant des flux de données est devenue essentielle [Vinagre et al., 2014b]. La plupart des RS proposés dans la littérature construisent d'abord un modèle à partir d'un vaste ensemble de données statique, puis le reconstruisent périodiquement, au fur et à mesure de la réception de nouvelles données. Apprendre un modèle à partir d'un jeu de données constamment en croissance est coûteux et la fréquence de mise à jour du modèle dépend généralement de sa complexité et de sa modularité. Par conséquent, les données générées par un utilisateur après la mise à jour du modèle ne peuvent pas être prises en compte par le RS avant la mise à jour suivante, ce qui signifie que le modèle ne peut pas s'adapter aux changements rapides et génère donc des recommandations de mauvaise qualité. Une façon de résoudre ce problème consiste à aborder le problème de recommandation comme un problème d'apprentissage sur flux de données et à développer des RS en ligne qui exploitent des flux de données continus et s'adaptent aux changements en temps réel.

Contexte non observable dans le cadre de la recommandation en ligne. La difficulté introduite par l'apprentissage à partir de données réelles est que le concept d'intérêt peut dépendre d'un contexte caché qui n'est pas explicitement fourni [Tsymbal, 2004]. Des changements au niveau de ce contexte caché entraînent des modifications au niveau du concept cible. Cette notion est connue sous le nom de *dérive de concept* et constitue un défi pour l'apprentissage à partir de flux de données [Gama et al., 2014]. Plusieurs efforts ont été déployés pour développer des techniques de détection de dérive de concepts et adapter les modèles en conséquence. Le problème est encore plus compliqué dans le cas des RS en ligne car plusieurs concepts, à savoir les utilisateurs et les articles, évoluent de manières différentes et à des rythmes différents. Nous étudions le problème de la recommandation adaptative en ligne, qui reste un problème sous-exploré malgré sa grande pertinence dans les applications du monde réel, montrons les limites du nombre réduit d'approches existantes et introduisons des approches plus performantes. Nos contributions peuvent être résumées comme suit :

- **Modèles locaux dynamiques.** Nous proposons un RS en ligne reposant sur des modèles locaux dynamiques avec une capacité d'adaptation aux *dérives des utilisateurs* ou aux changements au niveau des préférences des utilisateurs. Les modèles locaux sont connus pour leur capacité de capturer des préférences de granularités fines à travers les sous-ensembles d'utilisateurs. Notre approche détecte automatiquement les changements de préférences qui conduisent un utilisateur à adopter un comportement plus proche des utilisateurs d'un autre sous-ensemble et ajuste les modèles en conséquence. Nous montrons l'intérêt d'utiliser des modèles locaux afin de modéliser les dérives des utilisateurs.

- **Factorisation de matrices incrémentale adaptative.** Nous proposons une stratégie d'adaptation dynamique du taux d'apprentissage pour la factorisation de matrices incrémentale (IMF) qui tient en compte des *dérives d'articles* ou des changements au niveau des perceptions des articles, qui se produisent de façon indépendante pour chaque article. Le taux d'apprentissage est adapté dynamiquement en fonction des performances de chaque modèle d'article, ce qui permet de maintenir les modèles à jour. Nous démontrons l'avantage de l'apprentissage adaptatif dans les environnements non stationnaires comparé à un apprentissage à rythme constant.

- ***Topic model* collaboratif adaptatif.** Nous concevons un RS hybride en ligne qui exploite la description textuelle pour la modélisation de nouveaux articles reçus en temps réel en plus des préférences inférées à partir des interactions des utilisateurs. Notre approche prend en compte les *dérives d'articles* ou les modifications survenues au niveau des descriptions d'articles, en combinant un *topic model* avec une technique de détection de dérive de concepts. En plus de la gestion du problème de démarrage à froid en temps réel, nous veillons à ce que les modèles représentent l'état actuel des utilisateurs et des articles. Nous soulignons qu'en l'absence du module de détection de dérive, les informations textuelles tendent à introduire du bruit avec le temps et conduisent à de mauvaises recommandations.

Nous montrons, pour toutes les approches proposées, l'évolution de la performance du RS au fur et à mesure que l'apprentissage adaptatif est effectué, et nous prouvons l'intérêt de prendre en compte les dérives de concepts et les dynamiques temporelles afin d'aboutir à une meilleure qualité de recommandation.

A.3 Organisation de la thèse

La thèse est organisée comme suit :

Chapitre 2. Nous introduisons des notions préliminaires liées aux RS. Nous couvrons la définition du problème de recommandation, ses défis et ses limites, les méthodologies utilisées pour évaluer la qualité de recommandation, ainsi que le large ensemble d'approches de recommandation existantes, en fournissant plus de détails concernant celles directement reliées à la thèse.

Chapitre 3. Nous présentons le problème de recommandation d'hôtels, ses caractéristiques distinctives et sa relation avec d'autres problèmes de recommandation. Nous donnons également une analyse du comportement des voyageurs et décrivons la notion de contexte telle qu'elle apparaît dans le domaine.

Chapitre 4. Nous proposons un CARS pour la recommandation d'hôtels, intégrant les contextes physique, social et modal des utilisateurs, notamment les dimensions géographique et temporelle, les commentaires textuels extraits des réseaux sociaux et la raison du voyage. Nous présentons l'architecture du système développé en industrie et montrons l'impact de la prise en compte du contexte et de la segmentation des utilisateurs sur la qualité de recommandation.

Chapitre 5. Nous proposons un CARS intégrant les informations relatives aux événements organisés pour la recommandation d'hôtels et introduisons les problèmes de recommandation *centrée hôtel* et *centrée événement*. Nous démontrons le fonctionnement du CARS que nous proposons à travers des évaluations qualitative et quantitative et montrons les avantages de l'utilisation des données reliées aux événements pour la recommandation d'hôtels.

Chapitre 6. Nous proposons un RS interdomaine qui extrait des connaissances reliées à la mobilité géographique à partir des réseaux sociaux et les transfère au domaine de l'hôtellerie pour effectuer la recommandation. Nous décrivons comment nous associons les utilisateurs et les articles des deux domaines et montrons dans quels cas les données considérées contribuent à améliorer la qualité de recommandation.

Chapitre 7. Nous introduisons le problème de recommandation adaptative en ligne et exposons sa relation avec les travaux existants dans les domaines de l'exploration de flux de données et des RS. Nous présentons une structure pour la recommandation adaptative en ligne que nous utilisons pour passer en revue les travaux antérieurs et souligner leurs limitations.

Chapitre 8. Nous présentons DOLORES, le RS en ligne que nous proposons et qui s'adapte aux dérives des utilisateurs. DOLORES est basé sur des modèles locaux, capables de capturer des préférences diverses et opposées à travers les sous-ensembles d'utilisateurs. Nous montrons l'efficacité de l'utilisation de modèles locaux dynamiques pour s'adapter aux changements des préférences des utilisateurs.

Chapitre 9. Nous présentons AdaIMF, le RS en ligne que nous proposons et qui s'adapte aux dérives des perceptions d'articles. AdaIMF se base sur une nouvelle stratégie dynamique pour l'adaptation du taux d'apprentissage de la factorisation de matrices incrémentale (IMF). Nous montrons comment AdaIMF se comporte en présence de dérives et l'importance de la prise en compte de ces changements afin de délivrer de bonnes recommandations.

Chapitre 10. Nous présentons CoAWILDA, le RS en ligne que nous proposons et qui s'adapte aux dérives des descriptions d'articles. CoAWILDA combine des techniques de filtrage collaboratif (CF), de *topic modeling* et de détection de dérive. Nous montrons que les descriptions textuelles détériorent la qualité de recommandation en l'absence d'un composant de détection de dérive.

Chapitre 11. Nous résumons nos contributions et indiquons les directions pour les travaux futurs.

A.3.1 Publications

Certains des résultats présentés dans cette thèse sont basés sur les publications suivantes :

Chapitre 4 contient des résultats extraits de [Al-Ghossein et al., 2018d] :

[Al-Ghossein et al., 2018d] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Exploiting Contextual and External Data for Hotel Recommendation. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization (UMAP)*, pages 323–328, 2018d

Chapitre 5 contient des résultats extraits de [Al-Ghossein et al., 2018a] :

[Al-Ghossein et al., 2018a] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Open data in the hotel industry : leveraging forthcoming events for hotel recommendation. *Journal of Information Technology & Tourism*, pages 1–26, 2018a

Chapitre 6 contient des résultats extraits de [Al-Ghossein and Abdessalem, 2016; Al-Ghossein et al., 2018c] :

[Al-Ghossein and Abdessalem, 2016] Marie Al-Ghossein and Talel Abdessalem. SoMap : Dynamic Clustering and Ranking of Geotagged Posts. In *Proc. 25th International Conference Companion on World Wide Web (WWW)*, pages 151–154, 2016

[Al-Ghossein et al., 2018c] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Cross-Domain Recommendation in the Hotel Sector. In *Proc. Workshop on Recommenders in Tourism at the 12th ACM Conference on Recommender Systems (RecTour@RecSys)*, pages 1–6, 2018c

Chapitre 8 contient des résultats extraits de [Al-Ghossein et al., 2018b] :

[Al-Ghossein et al., 2018b] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Dynamic Local Models for Online Recommendation. In *Companion Proc. of the The Web Conference (WWW)*, pages 1419–1423, 2018b

Chapitre 10 contient des résultats extraits de [Al-Ghossein et al., 2018e,f; Murena et al., 2018] :

[Murena et al., 2018] Pierre-Alexandre Murena, Marie Al-Ghossein, Talel Abdessalem, and Antoine Cornuéjols. Adaptive Window Strategy for Topic Modeling in Document Streams. In *Proc. International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2018

[Al-Ghossein et al., 2018f] Marie Al-Ghossein, Pierre-Alexandre Murena, Antoine Cornuéjols, and Talel Abdessalem. Online Learning with Reoccurring Drifts : The Perspective of Case-Based Reasoning. In *Proc. Workshop on Synergies between CBR and Machine Learning at the 26th International Conference on Case-Based Reasoning (CBRML@ICCBR)*, pages 133–142, 2018f

[Al-Ghossein et al., 2018e] Marie Al-Ghossein, Pierre-Alexandre Murena, Talel Abdessalem, Anthony Barré, and Antoine Cornuéjols. Adaptive Collaborative Topic Modeling for Online Recommendation. In *Proc. 12th ACM Conference on Recommender Systems (RecSys)*, pages 338–346, 2018e

Bibliography

AccorHotels Panorama Juin 2018, 2018. https://www.accorhotels.group/-/media/Corporate/Home/Documents/Publications/PDF-for-pages/AccorHotels-Panorama-FR-Sept18.pdf.

Accor launches its digital transformation - "Leading Digital Hospitality", Oct 2014. https://www.hospitalitynet.org/news/4067567.html.

Panagiotis Adamopoulos and Alexander Tuzhilin. On Over-Specialization and Concentration Bias of Recommendations: Probabilistic Neighborhood Selection in Collaborative Filtering Systems. In *Proc. RecSys*, 2014.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE TKDE*, (6), 2005.

Gediminas Adomavicius and Alexander Tuzhilin. Context-Aware Recommender System. In *Recommender Systems Handbook*. 2015.

Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. *ACM TOIS*, 23(1), 2005.

Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. Context-Aware Recommender Systems. *AI Magazine*, 32(3), 2011.

Deepak Agarwal and Bee-Chung Chen. Regression-based Latent Factor Models. In *Proc. KDD*, 2009.

Deepak Agarwal and Bee-Chung Chen. fLDA: Matrix Factorization through Latent Dirichlet Allocation. In *Proc. WSDM*, 2010.

Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. Fast Online Learning through Offline Initialization for Time-sensitive Recommendation. In *Proc. KDD*, 2010.

Amr Ahmed, Choon Hui Teo, S.V.N. Vishwanathan, and Alex Smola. Fair and Balanced: Learning to Present News Stories. In *Proc. WSDM*, 2012.

Fabio Aiolli. Efficient Top-N Recommendation for Very Large Scale Binary Rated Datasets. In *Proc. RecSys*, 2013.

Marie Al-Ghossein and Talel Abdessalem. SoMap: Dynamic Clustering and Ranking of Geotagged Posts. In *Proc. WWW*, pages 151–154, 2016.

Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Open data in the hotel industry: leveraging forthcoming events for hotel recommendation. *JITT*, pages 1–26, 2018a.

Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Dynamic Local Models for Online Recommendation. In *Proc. WWW*, pages 1419–1423, 2018b.

Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Cross-Domain Recommendation in the Hotel Sector. In *Proc. Workshop on Recommenders in Tourism@RecSys*, pages 1–6, 2018c.

Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. Exploiting Contextual and External Data for Hotel Recommendation. In *Proc. UMAP*, pages 323–328, 2018d.

Marie Al-Ghossein, Pierre-Alexandre Murena, Talel Abdessalem, Anthony Barré, and Antoine Cornuéjols. Adaptive Collaborative Topic Modeling for Online Recommendation. In *Proc. RecSys*, pages 338–346, 2018e.

Marie Al-Ghossein, Pierre-Alexandre Murena, Antoine Cornuéjols, and Talel Abdessalem. Online Learning with Reoccurring Drifts: The Perspective of Case-Based Reasoning. In *Proc. Workshop on Synergies between CBR and Machine Learning@ICCBR*, pages 133–142, 2018f.

Cesare Alippi and Manuel Roveri. An Adaptive CUSUM-based Test for Signal Change Detection. In *Proc. ISCAS*. IEEE, 2006.

Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. On-Line LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking. In *Proc. IEEE ICDM*, 2008.

Xavier Amatriain. Mining Large Streams of User Data for Personalized Recommendations. *SIGKDD Explorations*, 14(2), 2013.

Xavier Amatriain and Justin Basilico. *Netflix Recommendations: Beyond the 5 stars (Part 1)*, 2012. `https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429`.

Xavier Amatriain and Justin Basilico. Past, Present, and Future of Recommender Systems: An Industry Perspective. In *Proc. RecSys*, 2016.

Xavier Amatriain and Josep M. Pujol. Data Mining Methods for Recommender Systems. In *Recommender Systems Handbook*. Springer, 2015.

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval: the concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England, 2011.

Marko Balabanović and Yoav Shoham. Fab: Content-Based, Collaborative Recommendation. *Commun. ACM*, 40(3), 1997.

Linas Baltrunas and Xavier Amatriain. Towards Time-Dependant Recommendation based on Implicit Feedback. In *Proc. Workshop on Context-Aware Recommender Systems@RecSys*, 2009.

Linas Baltrunas and Francesco Ricci. Experimental evaluation of context-dependent collaborative filtering using item splitting. *UMUAI*, 24(1-2), 2014.

Yang Bao, Hui Fang, and Jie Zhang. TopicMF: Simultaneously Exploiting Ratings and Reviews for Recommendation. In *Proc. AAAI*. AAAI Press, 2014.

Roberto Battiti. Accelerated Backpropagation Learning: Two Optimization Methods. *Complex Systems*, 3(4), 1989.

Robert M Bell and Yehuda Koren. Lessons from the Netflix Prize Challenge. *SIGKDD Explorations*, 9(2), 2007a.

Robert M Bell and Yehuda Koren. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. In *Proc. IEEE ICDM*, 2007b.

David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. RecSys Challenge 2015 and the YOOCHOOSE Dataset. In *Proc. RecSys*, 2015.

James Bennett and Stan Lanning. The Netflix Prize. In *Proc. KDD Cup and Workshop*, 2007.

Shlomo Berkovsky, Dan Goldwasser, Tsvi Kuflik, and Francesco Ricci. Identifying Inter-Domain Similarities through Content-Based Analysis of Hierarchical Web-Directories. In *Proc. ECAI*, 2006.

Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. Cross-Domain Mediation in Collaborative Filtering. In *Proc. UMAP*, 2007.

Lucas Bernardi, Jaap Kamps, Julia Kiseleva, and Melanie J. I. Müller. The Continuous Cold Start Problem in e-Commerce Recommender Systems. In *Proc. Workshop on New Trends on Content-Based Recommender Systems@RecSys*, 2015.

Alex Beutel, Ed H Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. Beyond Globally Optimal: Focused Learning for Improved Recommendations. In *Proc. WWW*, 2017.

Albert Bifet and Ricard Gavalda. Learning from Time-Changing Data with Adaptive Windowing. In *Proc. SDM*, 2007.

Daniel Billsus and Michael J Pazzani. Adaptive News Access. In *The Adaptive Web*. Springer, 2007.

David M Blei and John D Lafferty. Dynamic Topic Models. In *Proc. ICML*, 2006.

David M Blei and John D Lafferty. Topic Models. In *Text Mining: Theory and Applications*. Taylor and Francis, 2009.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *JMLR*, 3 (Jan), 2003.

Joan Borràs, Antonio Moreno, and Aida Valls. Intelligent tourism recommender systems: A survey. *Expert Systems with Applications*, 41(16), 2014.

Léon Bottou and Olivier Bousquet. The Tradeoffs of Large Scale Learning. In *Proc. NIPS*, 2008.

Andrew P Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7), 1997.

Matthew Brand. Fast online SVD revisions for lightweight recommender systems. In *Proc. SDM*. SIAM, 2003.

John S Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proc. UAI*. Morgan Kaufmann Publishers Inc., 1998.

Derek Bridge, Mehmet H Göker, Lorraine McGinty, and Barry Smyth. Case-Based Recommender Systems. *Knowledge Eng. Review*, 20(3), 2005.

Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *UMUAI*, 12(4), 2002.

Robin Burke. Evaluating the Dynamic Properties of Recommendation Algorithms. In *Proc. RecSys*, 2010.

Pedro G Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *UMUAI*, 24(1-2), 2014.

Iván Cantador, Alejandro Bellogín, and David Vallet. Content-based Recommendation in Social Tagging Systems. In *Proc. RecSys*, 2010.

Iván Cantador, Ignacio Fernández-Tobías, and Alejandro Bellogín. Relating Personality Types with User Preferences in Multiple Entertainment Domains. In *Proc. International Workshop on Emotions and Personality in Personalized Services@UMAP*, 2013.

Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. Cross-Domain Recommender Systems. In *Recommender Systems Handbook*. 2015.

Huanhuan Cao, Enhong Chen, Jie Yang, and Hui Xiong. Enhancing Recommender Systems Under Volatile User Interest Drifts. In *Proc. CIKM*, 2009.

Òscar Celma. Music Recommendation. In *Music recommendation and discovery*. Springer, 2010.

Òscar Celma and Perfecto Herrera. A New Approach to Evaluating Novel Recommendations. In *Proc. RecSys*, 2008.

Badrish Chandramouli, Justin J Levandoski, Ahmed Eldawy, and Mohamed F Mokbel. StreamRec: A Real-Time Recommender System. In *Proc. SIGMOD*, 2011.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading Tea Leaves: How Humans Interpret Topic Models. In *Proc. NIPS*, 2009.

Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A. Hasegawa-Johnson, and Thomas S. Huang. Streaming Recommender Systems. In *Proc. WWW*, 2017.

Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected Reciprocal Rank for Graded Relevance. In *Proc. CIKM*, 2009.

Li Chen and Pearl Pu. Critiquing-based recommenders: Survey and emerging trends. *UMUAI*, 22(1-2), 2012.

Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. A Learning-rate Schedule for Stochastic Gradient Methods to Matrix Factorization. In *Proc. PAKDD*. Springer, 2015.

Evangelia Christakopoulou and George Karypis. Local Item-Item Models for Top-N Recommendation. In *Proc. RecSys*, 2016.

Evangelia Christakopoulou and George Karypis. Local Latent Space Models for Top-N Recommendation. In *Proc. KDD*, 2018.

Ronald Chung, David Sundaram, and Ananth Srinivasan. Integrated Personal Recommender Systems. In *Proc. ICEC*, 2007.

Mark Claypool, Anuja Gokhale, Tim Miranda, Paul Murnikov, Dmitry Netes, and Matthew Sartin. Combing Content-Based and Collaborative Filters in an Online Newspaper. 1999.

Paul Covington, Jay Adams, and Emre Sargin. Deep Neural Networks for Youtube Recommendations. In *Proc. RecSys*, 2016.

Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An Experimental Comparison of Click Position-Bias Models. In *Proc. WSDM*, 2008.

Paolo Cremonesi and Massimo Quadrana. Cross-domain Recommendations without Overlapping Data: Myth or Reality? In *Proc. RecSys*, 2014.

Paolo Cremonesi, Antonio Tripodi, and Roberto Turrin. Cross-Domain Recommender Systems. In *Proc. ICDMW*, 2011.

Paolo Cremonesi, Franca Garzotto, and Massimo Quadrana. Evaluating Top-N Recommendations "When the Best are Gone". In *Proc. RecSys*, 2013.

Paolo Cremonesi, Franca Garzotto, Roberto Pagano, and Massimo Quadrana. Recommending without Short Head. In *Proc. WWW*, 2014.

Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google News Personalization: Scalable Online Collaborative Filtering. In *Proc. WWW*, 2007.

Gianmarco De Francisci Morales, Aristides Gionis, and Claudio Lucchese. From Chatter to Headlines: Harnessing the Real-Time Web for Personalized News Recommendation. In *Proc. WSDM*, 2012.

Marco De Gemmis, Pasquale Lops, Giovanni Semeraro, and Pierpaolo Basile. Integrating Tags in a Semantic Content-based Recommender. In *Proc. RecSys*, 2008.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y Ng. Large Scale Distributed Deep Networks. In *Proc. NIPS*, 2012.

Mukund Deshpande and George Karypis. Item-Based Top-N Recommendation Algorithms. *ACM TOIS*, 22(1), 2004.

Christian Desrosiers and George Karypis. A Comprehensive Survey of Neighborhood-based Recommendation Methods. In *Recommender Systems Handbook*. Springer, 2011.

Robin Devooght, Nicolas Kourtellis, and Amin Mantrach. Dynamic Matrix Factorization with Priors on Unknown Values. In *Proc. KDD*, 2015.

Anind K Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5 (1), 2001.

M Benjamin Dias, Dominique Locher, Ming Li, Wael El-Deredy, and Paulo JG Lisboa. The Value of Personalised Recommender Systems toe-Business: A Case Study. In *Proc. RecSys*, 2008.

Ernesto Diaz-Aviles, Lucas Drumond, Zeno Gantner, Lars Schmidt-Thieme, and Wolfgang Nejdl. What is Happening Right Now ... That Interests Me? Online Topic Discovery and Recommendation in Twitter. In *Proc. CIKM*, 2012a.

Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. Real-Time Top-N Recommendation in Social Streams. In *Proc. RecSys*, 2012b.

Jörg Diederich and Tereza Iofciu. Finding Communities of Practice from User Profiles Based on Folksonomies. In *Proc. EC-TEL06 Workshops*, 2006.

Yi Ding and Xue Li. Time Weight Collaborative Filtering. In *Proc. CIKM*, 2005.

Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in Nonstationary Environments: A Survey. *IEEE CIM*, 10(4), 2015.

Paul Dourish. What We Talk About When We Talk About Context. *Personal and Ubiquitous Computing*, 8(1), 2004.

Doychin Doychev, Aonghus Lawlor, Rachael Rafter, and Barry Smyth. An Analysis of Recommender Algorithms for Online News. In *Proc. CLEF*, 2014.

Lan Du, Wray Lindsay Buntine, and Huidong Jin. Sequential Latent Dirichlet Allocation: Discover Underlying Topic Structures within a Document. In *Proc. IEEE ICDM*, 2010.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *JMLR*, 12(Jul), 2011.

Ryan Elwell and Robi Polikar. Incremental Learning of Concept Drift in Nonstationary Environments. *IEEE Transactions on Neural Networks*, 22(10), 2011.

Elena Viorica Epure, Benjamin Kille, Jon Espen Ingvaldsen, Rebecca Deneckere, Camille Salinesi, and Sahin Albayrak. Recommending Personalized News in Short User Sessions. In *Proc. RecSys*, 2017.

Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *J. Comput. Sys. Sci.*, 66(4), 2003.

Alexander Felfernig, Sergiu Gordea, Dietmar Jannach, Erich Teppan, and Markus Zanker. A Short Survey of Recommendation Technologies in Travel and Tourism. *ÖGAI-Journal*, 25(7), 2007.

Ignacio Fernández-Tobías, Iván Cantador, and Laura Plaza. An Emotion Dimensional Model Based on Social Tags: Crossing Folksonomies and Enhancing Recommendations. In *Proc. EC-Web*, 2013.

Raffaele Filieri, Salma Alguezaui, and Fraser McLeay. Why do travelers trust TripAdvisor? Antecedents of trust towards consumer-generated media and its influence on recommendationadoption and word of mouth. *Tourism Management*, 51, 2015.

Brian Fling. *Mobile Design and Development: Practical concepts and techniques for creating mobile sites and Web apps.* O'Reilly Media, Inc., 2009.

Erzsébet Frigó, Róbert Pálovics, Domokos Kelen, Levente Kocsis, and András A. Benczúr. Online ranking prediction in non-stationary environments. In *Proc. Workshop on Temporal Reasoning in Recommender Systems@RecSys*, 2017.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine learning*, 90(3), 2013.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A Survey on Concept Drift Adaptation. *ACM CSUR*, 46(4), 2014.

Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning Attribute-to-Feature Mappings for Cold-start Recommendations. In *Proc. IEEE ICDM*, 2010a.

Zeno Gantner, Steffen Rendle, and Lars Schmidt-Thieme. Factorization Models for Context-/Time-Aware Movie Recommendations. In *Proc. Workshop on Context-Aware Movie Recommendation@RecSys*, 2010b.

L Gao, J Wu, C Zhou, and Y Hu. Collaborative Dynamic Sparse Topic Regression with User Profile Evolution for Item Recommendation. In *Proc. AAAI*, 2017.

Sheng Gao, Hao Luo, Da Chen, Shantao Li, Patrick Gallinari, and Jun Guo. Cross-Domain Recommendation via Cluster-Level Latent Factor Model. In *ECML/PKDD*. Springer, 2013.

Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In *Proc. RecSys*, 2010.

Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent. In *Proc. KDD*, 2011.

Thomas George and Srujana Merugu. A Scalable Collaborative Filtering Framework based on Co-clustering. In *Proc. IEEE ICDM*, 2005.

Donald Getz. Event tourism: Definition, evolution, and research. *Tourism management*, 29 (3), 2008.

David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM*, 35(12), 1992.

Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A Survey on Ensemble Learning for Data Stream Classification. *ACM CSUR*, 50(2), 2017.

Carlos A Gomez-Uribe and Neil Hunt. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM TMIS*, 6(4), 2016.

Ulrike Gretzel. Intelligent Systems in Tourism: A Social Science Perspective. *Annals of Tourism Research*, 38(3), 2011.

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proc. NAS*, 101(suppl 1), 2004.

Asela Gunawardana and Guy Shani. Evaluating Recommender Systems. In *Recommender Systems Handbook*. 2015.

John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proc. RecSys*, 2010.

Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns. In *Proc. RecSys*, 2012.

F Maxwell Harper and Joseph A Konstan. The Movielens Datasets: History and Context. *ACM TiiS*, 5(4), 2016.

John A Hartigan and Manchek A Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1), 1979.

Conor Hayes and Pádraig Cunningham. Context Boosting Collaborative Recommendations. *J. Knowledge-Based Systems*, 2(17), 2004.

Ruining He and Julian McAuley. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *Proc. AAAI*. AAAI Press, 2016.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proc. SIGIR*, 2016.

C.B. Hensley. Selective Dissemination of Information (SDI): State of The Art in May, 1963. In *Proc. AFIPS*, 1963.

Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proc. RecSys*, 2016.

Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and Evaluating Choices in a Virtual Community of Use. In *Proc. CHI*. ACM Press/Addison-Wesley Publishing Co., 1995.

Matthew Hoffman, Francis R Bach, and David M Blei. Online Learning for Latent Dirichlet Allocation. In *Proc. NIPS*, 2010.

Susan Horner and John Swarbrooke. *Consumer Behaviour in Tourism*. Routledge, 2016.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *Proc. IEEE ICDM*, 2008.

Xunpeng Huang, Le Wu, Enhong Chen, Hengshu Zhu, Qi Liu, and Yijun Wang. Incremental Matrix Factorization: A Linear Feature Transformation Perspective.

Yanxiang Huang, Bin Cui, Wenyu Zhang, Jie Jiang, and Ying Xu. TencentRec: Real-time Stream Recommendation in Practice. In *Proc. SIGMOD*, 2015.

Yanxiang Huang, Bin Cui, Jie Jiang, Kunqian Hong, Wenyu Zhang, and Yiran Xie. Real-time Video Recommendation Exploration. In *Proc. SIGMOD*, 2016.

Tomoharu Iwata, Takeshi Yamada, Yasushi Sakurai, and Naonori Ueda. Online Multiscale Dynamic Topic Models. In *Proc. KDD*, 2010.

Kurt Jacobson, Vidhya Murali, Edward Newett, Brian Whitman, and Romain Yon. Music Personalization at Spotify. In *Proc. RecSys*, 2016.

Michael Jahrer, Andreas Töscher, and Robert Legenstein. Combining Predictions for Accurate Recommender Systems. In *Proc. KDD*, 2010.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated Gain-based Evaluation of IR Techniques. *ACM TOIS*, 20(4), 2002.

Hamed Jelodar, Yongli Wang, Chi Yuan, and Xia Feng. Latent Dirichlet Allocation (LDA) and Topic modeling: models, applications, a survey. *arXiv preprint arXiv:1711.04305*, 2017.

Katerina Kabassi. Personalizing Recommendations for Tourists. *Telematics and Informatics*, 27(1), 2010.

Alexandros Karatzoglou. Collaborative Temporal Order Modeling. In *Proc. RecSys*, 2011.

Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering. In *Proc. RecSys*, 2010.

Mohammad Khoshneshin and W Nick Street. Incremental Collaborative Filtering via Evolutionary Co-clustering. In *Proc. RecSys*, 2010.

Houda Khrouf and Raphaël Troncy. Hybrid Event Recommendation using Linked Data and User Diversity. In *Proc. RecSys*, 2013.

Benjamin Kille, Frank Hopfgartner, Torben Brodt, and Tobias Heintz. The plista dataset. In *Proc. International News Recommender Systems Workshop and Challenge@RecSys*, 2013.

Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *Proc. RecSys*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, volume 5, 2015.

Julia Kiseleva, Melanie JI Mueller, Lucas Bernardi, Chad Davis, Ivan Kovacek, Mats Stafseng Einarsen, Jaap Kamps, Alexander Tuzhilin, and Djoerd Hiemstra. Where to Go on Your Next Trip? Optimizing Travel Destinations Based on User Preferences. In *Proc. SIGIR*, 2015.

Julia Kiseleva, Alexander Tuzhilin, Jaap Kamps, Melanie JI Mueller, Lucas Bernardi, Chad Davis, Ivan Kovacek, Mats Stafseng Einarsen, and Djoerd Hiemstra. Beyond Movie Recommendations: Solving the Continuous Cold Start Problem in E-commerce Recommendations. *arXiv preprint arXiv:1607.07904*, 2016.

Takuya Kitazawa. Incremental Factorization Machines for Persistently Cold-starting Online Item Recommendation. *arXiv preprint arXiv:1607.02858*, 2016.

Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intell. Data Anal.*, 8(3), 2004.

Daniel Kluver and Joseph A Konstan. Evaluating Recommender Behavior For New Users. In *Proc. RecSys*, 2014.

Noam Koenigstein and Yehuda Koren. Towards Scalable and Accurate Item-Oriented Recommendations. In *Proc. ReSys*, 2013.

Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! Music Recommendations: Modeling Music Ratings with Temporal Dynamics and Item Taxonomy. In *Proc. RecSys*, 2011.

Noam Koenigstein, Parikshit Ram, and Yuval Shavitt. Efficient Retrieval of Recommendations in a Matrix Factorization Framework. In *Proc. CIKM*, 2012.

Yehuda Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In *Proc. KDD*, 2008.

Yehuda Koren. Collaborative Filtering with Temporal Dynamics. In *Proc. KDD*, 2009.

Yehuda Koren and Robert Bell. Advances in Collaborative Filtering. In *Recommender Systems Handbook*. Springer, 2015.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques For Recommender Systems. *IEEE Computer*, (8), 2009.

Bruce Krulwich. LIFESTYLE FINDER: Intelligent User Profiling Using Large-Scale Demographic Data. *AI magazine*, 18(2), 1997.

Yohei Kurata. CT-Planner2: More Flexible and Interactive Assistance for Day Tour Planning. In *Proc. ENTER*, 2011.

Miklós Kurucz, András A Benczúr, and Károly Csalogány. Methods for large scale SVD with missing values. In *Proc. KDD Cup and Workshop*, volume 12, 2007.

Neal Lathia, Stephen Hailes, and Licia Capra. Temporal Collaborative Filtering With Adaptive Neighbourhoods. In *Proc. SIGIR*, 2009.

Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local Low-rank Matrix Approximation. In *Proc. ICML*, 2013.

Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local Collaborative Ranking. In *Proc. WWW*, 2014.

Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. A Survey Of Algorithms For Dense Subgraph Discovery. In *Managing and Mining Graph Data*. Springer, 2010.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, 6(2), 2015.

Daniel Lemire and Anna Maclachlan. Slope One Predictors for Online Rating-Based Collaborative Filtering. In *Proc. SDM*. SIAM, 2005.

Lukas Lerche and Dietmar Jannach. Using Graded Implicit Feedback for Bayesian Personalized Ranking. In *Proc. RecSys*, 2014.

Matthaios Letsios, Oana Denisa Balalau, Maximilien Danisch, Emmanuel Orsini, and Mauro Sozio. Finding Heaviest k-Subgraphs and Events in Social Media. In *Proc. ICDMW*, 2016.

Justin J Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F Mokbel. LARS: A Location-Aware Recommender System. In *Proc. IEEE ICDE*. IEEE, 2012.

Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. Finding a Needle in a Haystack of Reviews: Cold Start Context-Based Hotel Recommender System. In *Proc. RecSys*, 2012.

Bin Li, Qiang Yang, and Xiangyang Xue. Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction. In *Proc. IJCAI*, volume 9, 2009.

Dongsheng Li, Chao Chen, Qin Lv, Hansu Gu, Tun Lu, Li Shang, Ning Gu, and Stephen M. Chu. AdaError: An Adaptive Learning Rate Method for Matrix Approximation-based Collaborative Filtering. In *Proc. WWW*, 2018.

Chen Lin, Runquan Xie, Xinjun Guan, Lei Li, and Tao Li. Personalized news recommendation via implicit social experts. *Inf. Sci.*, 254, 2014.

Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, (1), 2003.

Nick Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine learning*, 2(4), 1988.

Stephen W Litvin, Ronald E Goldsmith, and Bing Pan. Electronic Word-of-Mouth in Hospitality and Tourism Management. *Tourism management*, 29(3), 2008.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized News Recommendation Based on Click Behavior. In *Proc. IUI*, 2010a.

Nathan N Liu, Bin Cao, Min Zhao, and Qiang Yang. Adapting neighborhood and matrix factorization models for context aware recommendation. In *Proc. Workshop on Context-Aware Movie Recommendation@RecSys*, 2010b.

Nathan N Liu, Min Zhao, Evan Xiang, and Qiang Yang. Online Evolutionary Collaborative Filtering. In *Proc. RecSys*, 2010c.

Xin Liu. Modeling Users' Dynamic Preference for Personalized Recommendation. In *Proc. IJCAI*, 2015.

Xin Liu and Karl Aberer. Towards a dynamic top-N recommendation framework. In *Proc. RecSys*, 2014.

Xingjie Liu, Qi He, Yuanyuan Tian, Wang-Chien Lee, John McPherson, and Jiawei Han. Event-based Social Networks: Linking the Online and Offline Social Worlds. In *Proc. KDD*, 2012.

Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. An Experimental Evaluation of Point-of-Interest Recommendation in Location-Based Social Networks. *PVLDB*, 10(10), 2017.

Babak Loni, Yue Shi, Martha Larson, and Alan Hanjalic. Cross-Domain Collaborative Filtering with Factorization Machines. In *Proc. ECIR*, 2014.

Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook*. Springer, 2011.

Yucheng Low, Deepak Agarwal, and Alexander J Smola. Multiple Domain User Personalization. In *Proc. KDD*, 2011.

Zhongqi Lu, Sinno Jialin Pan, Yong Li, Jie Jiang, and Qiang Yang. Collaborative Evolution for User Profiling in Recommender Systems. In *Proc. IJCAI*. AAAI Press, 2016.

Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. SoRec: Social Recommendation Using Probabilistic Matrix Factorization. In *Proc. CIKM*, 2008.

Augusto Q Macedo, Leandro B Marinho, and Rodrygo LT Santos. Context-Aware Event Recommendation in Event-based Social Networks. In *Proc. RecSys*, 2015.

Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. Collaborative Filtering and the Missing at Random Assumption. In *Proc. UAI*, 2007.

Pawel Matuszyk and Myra Spiliopoulou. Selective Forgetting for Incremental Matrix Factorization in Recommender Systems. In *Proc. DS*, 2014.

Pawel Matuszyk, João Vinagre, Myra Spiliopoulou, Alípio Mário Jorge, and João Gama. Forgetting Methods for Incremental Matrix Factorization in Recommender Systems. In *Proc. SAC*, 2015.

Einat Minkov, Ben Charrow, Jonathan Ledlie, Seth Teller, and Tommi Jaakkola. Collaborative Future Event Recommendation. In *Proc. CIKM*, 2010.

Catarina Miranda and Alípio Mário Jorge. Item-Based and User-Based Incremental Collaborative Filtering for Web Recommendations. In *Proc. EPIA*. Springer, 2009.

Tom M Mitchell. Machine Learning. *Burr Ridge, IL: McGraw Hill*, 45(37), 1997.

Koji Miyahara and Michael J Pazzani. Collaborative Filtering with the Simple Bayesian Classifier. In *Proc. PRICAI*. Springer, 2000.

Douglas C Montgomery and George C Runger. *Applied statistics and probability for engineers*. John Wiley & Sons, 2010.

Raymond J Mooney and Loriene Roy. Content Based Book Recommending Using Learning for Text Categorization. In *Proc. ACM DL*, 2000.

Orly Moreno, Bracha Shapira, Lior Rokach, and Guy Shani. TALMUD – Transfer Learning for Multiple Domains. In *Proc. CIKM*, 2012.

Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. Metrics for Evaluating the Serendipity of Recommendation Lists. In *Proc. JSAI*. Springer, 2007.

Pierre-Alexandre Murena, Marie Al-Ghossein, Talel Abdessalem, and Antoine Cornuéjols. Adaptive Window Strategy for Topic Modeling in Document Streams. In *Proc. IJCNN*, pages 1–7, 2018.

Olfa Nasraoui, Jeff Cerwinske, Carlos Rojas, and Fabio Gonzalez. Performance of Recommendation Systems in Dynamic Streaming Environments. In *Proc. SDM*. SIAM, 2007.

Tien T. Nguyen, Pik-Mai Hui, F. Maxwell Harper, Loren Terveen, and Joseph A. Konstan. Exploring the Filter Bubble: The Effect of Using Recommender Systems on Content Diversity. In *Proc. WWW*, 2014.

Mehrbakhsh Nilashi, Othman bin Ibrahim, Norafida Ithnin, and Nor Haniza Sarmin. A multi-criteria collaborative filtering recommender system for the tourism domain using Expectation Maximization (EM) and PCA–ANFIS. *Electronic Commerce Research and Applications*, 14(6), 2015.

Xia Ning and George Karypis. Slim: Sparse Linear Methods for Top-N Recommender Systems. In *Proc. IEEE ICDM*, 2011.

Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*. 2015.

Douglas W. Oard and Jinmook Kim. Implicit Feedback for Recommender Systems. In *Proc. Workshop on Recommender Systems@AAAI*, 1998.

Mark O'Connor and Jon Herlocker. Clustering Items for Collaborative Filtering. In *Proc. Workshop on Recommender Systems@SIGIR*, volume 128, 1999.

Òscar Celma. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.

Roberto Pagano, Paolo Cremonesi, Martha Larson, Balázs Hidasi, Domonkos Tikk, Alexandros Karatzoglou, and Massimo Quadrana. The Contextual Turn: from Context-Aware to Context-Driven Recommender Systems. In *Proc. RecSys*, 2016.

Cosimo Palmisano, Alexander Tuzhilin, and Michele Gorgoglione. Using Context to Improve Predictive Modeling of Customers in Personalization Applications. *IEEE TKDE*, 20(11), 2008.

Róbert Pálovics, András A Benczúr, Levente Kocsis, Tamás Kiss, and Erzsébet Frigó. Exploiting Temporal Influence in Online Recommendation. In *Proc. RecSys*, 2014.

Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-Class Collaborative Filtering. In *Proc. IEEE ICDM*, 2008.

Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. Transfer Learning in Collaborative Filtering for Sparsity Reduction. In *Proc. AAAI*, volume 10, 2010.

Weike Pan, Nathan N. Liu, Evan W. Xiang, and Qiang Yang. Transfer Learning to Predict Missing Ratings via Heterogeneous User Feedbacks. In *Proc. IJCAI*, 2011.

Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental Comparison of Pre- vs. Post-Filtering Approaches in Context-Aware Recommender systems. In *Proc. RecSys*, 2009.

Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, and Elias Theoharopoulos. Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms. In *Proc. ISMIS*. Springer, 2005.

Seung-Taek Park and Wei Chu. Pairwise Preference Regression for Cold-start Recommendation. In *Proc. RecSys*, 2009.

Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. KDD Cup and Workshop*, volume 2007, 2007.

Michael Pazzani and Daniel Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine learning*, 27(3), 1997.

Michael J Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artif. Intell. Rev.*, 13(5-6), 1999.

Michael J Pazzani and Daniel Billsus. Content-Based Recommendation Systems. In *The Adaptive Web*. Springer, 2007.

Alexandrin Popescul, David M Pennock, and Steve Lawrence. Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. In *Proc. UAI*, 2001.

Zhi Qiao, Peng Zhang, Chuan Zhou, Yanan Cao, Li Guo, and Yanchun Zhang. Event Recommendation in Event-Based Social Networks. In *Proc. AAAI*, 2014.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-Aware Recommender Systems. *ACM CSUR*, 51(4), 2018.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Proc. NIPS*, 2011.

Steffen Rendle. Factorization Machines with libFM. *ACM TIST*, 3(3), 2012.

Steffen Rendle and Lars Schmidt-Thieme. Online-Updating Regularized Kernel Matrix Factorization Models for Large-Scale Recommender Systems. In *Proc. RecSys*, 2008.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proc. UAI*, 2009.

Jasson DM Rennie and Nathan Srebro. Fast Maximum Margin Matrix Factorization for Collaborative Prediction. In *Proc. ICML*, 2005.

Paul Resnick and Hal R Varian. Recommender Systems. *Commun. ACM*, 40(3), 1997.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. CSCW*, 1994.

Francesco Ricci. Travel Recommender Systems. *IEEE Intelligent Systems*, 17(6), 2002.

Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender Systems: Introduction and Challenges. In *Recommender Systems Handbook*. Springer, 2015.

Ryosuke Saga, Yoshihiro Hayashi, and Hiroshi Tsuji. Hotel Recommender System Based on User's Preference Transition. In *Proc. SMC*. IEEE, 2008.

Ruslan Salakhutdinov and Andriy Mnih. Probabilistic Matrix Factorization. In *Proc. NIPS*, 2008a.

Ruslan Salakhutdinov and Andriy Mnih. Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. In *Proc. ICML*, 2008b.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann Machines for Collaborative Filtering. In *Proc. ICML*, 2007.

Raul Sanchez-Vazquez, Jordan Silva, and Rodrygo LT Santos. Exploiting Socio-Economic Models for Lodging Recommendation in the Sharing Economy. In *Proc. RecSys*, 2017.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of Dimensionality Reduction in Recommender System – A Case Study. 2000.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proc. WWW*, 2001.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. In *Proc. CIT*, 2002.

Martin Saveski and Amin Mantrach. Item Cold-Start Recommendations: Learning Local Collective Embeddings. In *Proc. RecSys*, 2014.

J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative Filtering Recommender Systems. In *The Adaptive Web*. Springer, 2007.

Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and Metrics for Cold-Start Recommendations. In *Proc. SIGIR*, 2002.

Laura Sebastia, Inma Garcia, Eva Onaindia, and Cesar Guzman. e-Tourism : A Tourist Recommendation And Planning Application. *IJAIT*, 18(05), 2009.

Bracha Shapira, Lior Rokach, and Shirley Freilikhman. Facebook single and cross domain data for recommendation systems. *UMUAI*, 23(2-3), 2013.

Upendra Shardanand and Pattie Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proc. CHI*. ACM Press/Addison-Wesley Publishing Co., 1995.

Yue Shi, Martha Larson, and Alan Hanjalic. List-Wise Learning to Rank with Matrix Factorization for Collaborative Filtering. In *Proc. RecSys*, 2010.

Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering. In *Proc. RecSys*, 2012.

Zaigham Faraz Siddiqui, Eleftherios Tiakas, Panagiotis Symeonidis, Myra Spiliopoulou, and Yannis Manolopoulos. xStreams: Recommending Items to Users with Time-evolving Preferences. In *Proc. WIMS*, 2014.

Ian Soboroff and Charles Nicholas. Combining Content and Collaboration in Text Filtering. In *Proc. Workshop on Machine Learning for Information Filtering@IJCAI*, 1999.

Qiang Song, Jian Cheng, and Hanqing Lu. Incremental Matrix Factorization via Feature Space Re-learning for Recommender System. In *Proc. RecSys*, 2015.

Nathan Srebro and Tommi Jaakkola. Weighted Low-Rank Approximations. In *Proc. ICML*, 2003.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-Margin Matrix Factorization. In *Proc. NIPS*, 2005.

Karthik Subbian, Charu Aggarwal, and Kshiteesh Hegde. Recommendations For Streaming Data. In *Proc. CIKM*, 2016.

Lucy A. Suchman. Plans and Situated Actions. 1987.

Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major components of the Gravity Recommendation System. *SIGKDD Explorations*, 9(2), 2007.

Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable Collaborative Filtering Approaches for Large Recommender Systems. *JMLR*, 10(Mar), 2009.

Liang Tang, Yexi Jiang, Lei Li, and Tao Li. Ensemble Contextual Bandits for Personalized Recommendation. In *Proc. RecSys*, 2014.

Christina Teflioudi, Faraz Makari, and Rainer Gemulla. Distributed Matrix Completion. In *Proc. IEEE ICDM*, 2012.

Christina Teflioudi, Rainer Gemulla, and Olga Mykytiuk. LEMP: Fast Retrieval of Large Entries in a Matrix Product. In *Proc. SIGMOD*, 2015.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The New Data in Multimedia Research. *Commun. ACM*, 59(2), 2016.

Nava Tintarev and Judith Masthoff. Explaining Recommendations: Design and Evaluation. In *Recommender Systems Handbook*. Springer, 2015.

Christoph Trattner, Alexander Oberegger, Leandro Marinho, and Denis Parra. Investigating the Utility of the Weather Context for Point of Interest Recommendations. *JITT*, 19(1-4), 2018.

Alexey Tsymbal. The Problem of Concept Drift: Definitions and Related Work. *Technical Report, Department of Computer Science, TCD*, 106(2), 2004.

Lyle H Ungar and Dean P Foster. Clustering Methods for Collaborative Filtering. In *Workshop on Recommendation Systems@AAAI*, 1998.

Mark Van Setten, Stanislav Pokraev, and Johan Koolwaaij. Context-Aware Recommendations in the Mobile Tourist Application COMPASS. In *Proc. AH*. Springer, 2004.

Saúl Vargas and Pablo Castells. Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. In *Proc. RecSys*, 2011.

Koen Verstrepen, Kanishka Bhaduriy, Boris Cule, and Bart Goethals. Collaborative Filtering for Binary, Positive-only Data. *SIGKDD Explorations*, 19(1), 2017.

João Vinagre and Alípio Mário Jorge. Forgetting mechanisms for scalable collaborative filtering. *J. Braz. Comp. Soc.*, 18(4), 2012.

João Vinagre, Alípio Mário Jorge, and João Gama. Evaluation of Recommender Systems in Streaming Environments. In *Proc. Workshop on Recommender Systems Evaluation: Dimensions and Design@RecSys*, 2014a.

João Vinagre, Alípio Mário Jorge, and João Gama. Fast Incremental Matrix Factorization for Recommendation with Positive-Only Feedback. In *Proc. UMAP*, 2014b.

João Vinagre, Alípio Mário Jorge, and João Gama. Collaborative filtering with recency-based negative feedback. In *Proc. SAC*, 2015a.

João Vinagre, Alípio Mário Jorge, and João Gama. An overview on the exploitation of time in collaborative filtering. *WIREs Data Min. Knowl. Discov.*, 5(5), 2015b.

João Vinagre, Alípio Mário Jorge, and João Gama. Online Bagging for Recommender Systems. *Expert Systems*, 35(4), 2018.

Jeffrey S Vitter. Random Sampling with a Reservoir. *ACM TOMS*, 11(1), 1985.

Chong Wang and David M Blei. Collaborative Topic Modeling for Recommending Scientific Articles. In *Proc. KDD*, 2011.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative Deep Learning for Recommender Systems. In *Proc. KDD*, 2015.

Jialei Wang, Steven CH Hoi, Peilin Zhao, and Zhi-Yong Liu. Online Multi-Task Collaborative Filtering for On-the-Fly Recommender Systems. In *Proc. RecSys*, 2013.

Wei Wang, Jiong Yang, and Richard Muntz. STING : A Statistical Information Grid Approach to Spatial Data Mining. In *Proc. VLDB*, 1997.

Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. Streaming Ranking Based Recommender Systems. In *Proc. SIGIR*, 2018.

Xuerui Wang and Andrew McCallum. Topics over Time: A Non-Markov Continuous-Time Model of Topical Trends. In *Proc. KDD*, 2006.

Feng Wei, Hao Guo, Shaoyin Cheng, and Fan Jiang. AALRSMF: An Adaptive Learning Rate Schedule for Matrix Factorization. In *Proc. APWeb*. Springer, 2016.

Markus Weimer, Alexandros Karatzoglou, Quoc V Le, and Alex J Smola. COFIRANK Maximum Margin Matrix Factorization for Collaborative Ranking. In *Proc. NIPS*, 2008.

Hannes Werthner and Francesco Ricci. E-commerce and tourism. *Commun. ACM*, 47(12), 2004.

Hannes Werthner, Aurkene Alzua-Sorzabal, Lorenzo Cantoni, Astrid Dickinger, Ulrike Gretzel, Dietmar Jannach, Julia Neidhardt, Birgit Pröll, Francesco Ricci, Miriam Scaglione, Brigitte Stangl, Oliviero Stock, and Markus Zanker. Future Research Issues in IT and Tourism. *JITT*, 15(1), 2015.

Patricia M West, Dan Ariely, Steve Bellman, Eric Bradlow, Joel Huber, Eric Johnson, Barbara Kahn, John Little, and David Schkade. Agents to the Rescue? *Marketing letters*, 10(3), 1999.

Gerhard Widmer and Miroslav Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine learning*, 23(1), 1996.

Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal Recommendation on Graphs via Long- and Short-term Preference Fusion. In *Proc. KDD*, 2010.

Zheng Xiang, Vincent P Magnini, and Daniel R Fesenmaier. Information Technology and Consumer Behavior in Travel and Tourism: Insights from Travel Planning Using the Internet. *J. Retailing and Consumer Services*, 22, 2015.

Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. An Exploration of Improving Collaborative Recommender Systems via User-Item Subgroups. In *Proc. WWW*, 2012.

Dingqi Yang, Daqing Zhang, and Bingqing Qu. Participatory Cultural Mapping Based on Collective Behavior Data in Location-Based Social Networks. *ACM TIST*, 7(3), 2016.

Shuang-Hong Yang, Bo Long, Alexander J Smola, Hongyuan Zha, and Zhaohui Zheng. Collaborative Competitive Filtering: Learning Recommender Using Context of User Choice. In *Proc. SIGIR*, 2011.

Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. Exploiting Geographical Influence for Collaborative Point-of-Interest Recommendation. In *Proc. SIGIR*, 2011.

Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. Dynamic User Modeling in Social Media Systems. *ACM TOIS*, 33(3), 2015a.

Hongzhi Yin, Bin Cui, Xiaofang Zhou, Yingxia Shao, Hao Wang, and Shazia Sadiq. Joint Modeling of User Check-in Behaviors for Real-time Point-of-Interest Recommendation. In *Proc. CIKM*, 2015b.

Tong Yu, Ole J Mengshoel, Alvin Jude, Eugen Feller, Julien Forgeat, and Nimish Radia. Incremental Learning for Matrix Factorization in Recommender Systems. In *Proc. IEEE BigData*, 2016.

Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, SVN Vishwanathan, and Inderjit Dhillon. NOMAD: Non-locking, stOchastic Multi-machine algorithm for Asynchronous and Decentralized matrix completion. *PVLDB*, 7(11), 2014.

Matthew D Zeiler. ADADELTA: An Aadaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*, 2012.

Kai Zhang, Keqiang Wang, Xiaoling Wang, Cheqing Jin, and Aoying Zhou. Hotel recommendation based on user preference analysis. In *Proc. ICDEW*. IEEE, 2015.

Mi Zhang and Neil Hurley. Avoiding monotony: Improving the diversity of recommendation lists. In *Proc. RecSys*, 2008.

Wancai Zhang, Hailong Sun, Xudong Liu, et al. An Incremental Tensor Factorization Approach for Web Service Recommendation. In *Proc. ICDMW*. IEEE, 2014.

Xiangyu Zhao, Zhendong Niu, and Wei Chen. Opinion-Based Collaborative Filtering to Solve Popularity Bias in Recommender Systems. In *Proc. DEXA*. Springer, 2013.

Yong Zheng, Robin Burke, and Bamshad Mobasher. The Role of Emotions in Context-aware Recommendation. 2013.

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale Parallel Collaborative Filtering for the Netflix Prize. In *Proc. AAIM*. Springer, 2008.

Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving Recommendation Lists Through Topic Diversification. In *Proc. WWW*, 2005.

Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. Using Temporal Data for Making Recommendations. In *Proc. UAI*, 2001.

Onno Zoeter. Recommendations in Travel. In *Proc. RecSys*, 2015.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Systèmes de recommandation contextuels pour les applications du monde réel

Mots clés : Systèmes de recommandation, recommandation d'hôtels, recommandation en ligne

Résumé : L'exploitation de l'information contextuelle dans les systèmes de recommandation permet une meilleure modélisation de l'aspect dynamique des utilisateurs et des articles. La définition traditionnelle du contexte, adoptée dans la plupart des systèmes de recommandation contextuels, ne répond pas à plusieurs contraintes rencontrées dans les applications du monde réel. Dans cette thèse, nous abordons les problèmes de recommandation en présence d'informations contextuelles *partiellement observables* et *non observables* dans deux applications respectives, la recommandation d'hôtels et la recommandation en ligne, remettant en question plusieurs aspects de la définition traditionnelle du contexte, notamment l'accessibilité, la pertinence, l'acquisition et la modélisation.

La première partie de la thèse étudie le problème de recommandation d'hôtels qui souffre du démarrage à froid continu, limitant la performance des approches classiques de recommandation. Le voyage n'est pas une activité fréquente et les utilisateurs ont tendance à adopter des comportements diversifiés en fonction de leurs situations spécifiques. Après une analyse du comportement des utilisateurs dans ce domaine, nous proposons de nouvelles approches de recommandation intégrant des informations contextuelles partiellement observables affectant les utilisateurs. Nous montrons comment cela contribue à améliorer la qualité des recommandations.

La deuxième partie de la thèse aborde le problème de recommandation adaptative en ligne en présence de flux de données où les observations apparaissent continûment à haute fréquence. Nous considérons que les utilisateurs et les articles reposent sur des informations contextuelles non observables par le système et évoluent de façons différentes à des rythmes différents. Nous proposons alors d'effectuer de la détection active de changements et d'assurer la mise à jour des modèles en temps réel. Nous concevons de nouvelles méthodes qui s'adaptent aux changements qui apparaissent au niveau des préférences des utilisateurs et des perceptions et descriptions des articles, et montrons l'importance de la recommandation adaptative en ligne pour garantir de bonnes performances au cours du temps.

Title : Context-Aware Recommender Systems for Real-World Applications

Keywords : Recommender systems, hotel recommendation, online recommendation

Abstract : Recommender systems have proven to be valuable tools to help users overcome the information overload, and significant advances have been made in the field over the last two decades. In particular, contextual information has been leveraged to model the dynamics occurring within users and items. Context is a complex notion and its traditional definition, which is adopted in most recommender systems, fails to cope with several issues occurring in real-world applications. In this thesis, we address the problems of *partially observable* and *unobservable* contexts in two particular applications, hotel recommendation and online recommendation, challenging several aspects of the traditional definition of context, including accessibility, relevance, acquisition, and modeling.

The first part of the thesis investigates the problem of hotel recommendation which suffers from the continuous cold-start issue, limiting the performance of classical approaches for recommendation. Traveling is not a frequent activity and users tend to have multifaceted behaviors depending on their specific situation. Following an analysis of the user behavior in this domain, we propose novel recommendation approaches integrating *partially observable* context affecting users and we show how it contributes in improving the recommendation quality.

The second part of the thesis addresses the problem of online adaptive recommendation in streaming environments where data is continuously generated. Users and items may depend on some *unobservable context* and can evolve in different ways and at different rates. We propose to perform online recommendation by actively detecting drifts and updating models accordingly in real-time. We design novel methods adapting to changes occurring in user preferences, item perceptions, and item descriptions, and show the importance of online adaptive recommendation to ensure a good performance over time.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France