Compression guidée par automate et noyaux rationnels

Ahmed Amarni

UNIVERSITÉ PARIS-EST

THESE DE DOCTORAT DE L'UNIVERSITÉ Paris-Est

Spécialité
Informatique

École Doctorale Mathématiques et STIC (MSTIC)

Présentée par
Ahmed AMARNI

Pour obtenir le grade de

DOCTEUR de l'UNIVÉRSITÉ Paris-Est Marne-la-vallée

Sujet de la thèse :

Compression guidée par automates et noyaux rationnels

Soutenue le 11 Mai 2015

Devant le jury composé de :

Sylvain LOMBARDY	**LABRI**	**Directeur de thèse**
Marie-pierre BÉAL	**LIGM**	**Codirecteur de thèse**
Olivier CARTON	**LIAFA**	**Rapporteur**
Pascal CARON	**LITIS**	**Rapporteur**

A ma grand mère Tassadit.

Remerciements

Je remercie vivement Sylvain Lombardy pour m'avoir dirigé pendant cette thèse, ainsi que pour ses conseils précieux pendant tout le temps qu'il consacrait pour moi.

Je remercie chaleuresement Marie-pierre Béal pour m'avoir accueilli au sein du laboratoire LIGM.

Je remercie Pascal Caron et Olivier Carton d'avoir accepté d'être rapporteurs de mon mémoire de thèse.

Je remercie tout le personnel de l'école doctorale MSTIC Paris-Est pour avoir permis que cette thèse se deroule dans les meilleurs conditions.

Je remercie les chercheurs du LIGM dont l'accueil fut chaleureux ainsi en particulier les doctorants : Sven, Vincent, Manar...

Je remercie tous ceux qui m'ont permis de passer toutes les étapes qui me conduisent aujourd'hui à rediger ce memoire, tous mes professeurs de l'école primaire à la preparation de cette thèse, en passant par le master où j'ai mené un stage au sein du laboratoire LITIS de l'université de Rouen qui m'a permis de me familiariser avec les transducteurs et les noyaux rationnels, je remercie donc vivement Eric Laugerotte, et Djelloul Ziadi de m'avoir encadré.

Je tiens à remercier chaleureusement tous mes amis en particulier Marion, Ahmed, Manuella, Sarah et Sébastien.

Enfin, tout ceci n'aurait pas été possible sans le soutien de ma famille, ainsi que mes frères Mohammed et Amar, mes soeurs Lila, Nadia, Akila et Sarah et plus que tous mes parents... Je les remercie du fond du cœur.

Abstract

Due to the expansion of datas, compression algorithms are now crucial algorithms. We address here the problem of finding an optimal compression algorithm with respect to a given Markovian source. To this purpose, we extend the classical Huffman algorithm. We describe some algorithm and analyse their efficiency.

The kernels are popular methods to measure the similarity between words for classication and learning. We generalize the definition of rational kernels in order to apply kernels to the comparison of languages. We study this generalization for factor and subsequence kernels and prove that these kernels are defined for parameters chosen in an appropriate interval. We give different methods to build weighted transducers which compute these kernels.

Résumé

En raison de l'expansion des données, les algorithmes de compression sont désormais cruciaux. Nous abordons ici le problème de trouver des algorithmes de compression optimaux par rapport à une source de Markov donnée. A cet effet, nous étendons l'algorithme de Huffman classique. Le codage à chaque instant dépend alors à la fois de la probabilité d'apparition des lettres mais aussi de l'état de la source. Partant du constat que l'algorithme ainsi décrit se comporte mieux si les lettres ou facteurs à encoder ont une probabilité d'apparition proche de $\frac{1}{2}$, on décrit un nouvel algorithme qui encode des facteurs de longueur variable afin de se placer le plus près possibles de ces conditions idéales.

Les noyaux sont des méthodes courantes pour mesurer la similarité entre les mots pour la classification et l'apprentissage. Nous généralisons la définition des noyaux rationnels afin d'appliquer les noyaux à la comparaison des langages. Nous étudions cette généralisation pour les noyaux de facteurs et de sous-mots et prouvons que ces noyaux sont définis pour les paramètres choisis dans un intervalle approprié. Nous donnons différentes méthodes pour construire des transducteurs pondérés qui calculent ces noyaux. Ensuite, on illustre cette étude par des exemples réels pour mesurer si ces noyaux permettent effectivement de classifier des documents selon différents corpus.

Table des matières

Chapitre 1

Introduction

Cette thèse est consacrée d'une part à l'étude de la généralisation des algorithmes très connus de compression symbole par symbole vers une compression d'une source Markovienne introduit dans ([21]), d'autre part à donner des algorithmes de calcul de noyaux et signatures de langages rationnels. Jusqu'à présent, ces même algorithmes ont été couramment appliqués aux chaînes de caractères, c'est-à-dire aux noyaux de mots.

Voici quelques motivations des noyaux. En informatique, l'apprentissage automatique consiste à exploiter des données préexistantes pour créer des modèles mathématiques capables de faire des prédictions qui permettent de décider la séparation, la classification ou la reconnaissance de ces données : lecture des écritures manuscrites par un système informatique, traitement de la voix et du langage naturel, etc.

De nombreux travaux et de progrès ont été constatés durant cette dernière décennie en apprentissage ([1, 34, 17]), combinatoire des mots ([26]) et dans de nombreux domaines d'applications comme ceux introduits dans ([29]) pour la reconnaissance de la voix.

Au départ les algorithmes conçus pour cette problématique traitaient des objets de taille fixe (vecteurs de composante). Actuellement, beaucoup de problèmes font intervenir des séquences de tailles variables.

Le problème consiste à trouver des algorithmes très efficaces (telles que les fonctions noyaux) qui permettent l'apprentissage ou la séparation de données. Des algorithmes efficaces dans la mesure où ils répondent rapidement à leur interrogation en un temps le plus court possible quand la taille des objets traités devient importante.

En classification et en apprentissage, les méthodes à noyaux, comme *support vector machine*, sont largement utilisées ([4, 22, 19, 7, 16, 33]).

Dans plusieurs domaines, comme la reconnaissance de voix et des documents manuscrits, ou dans les calculs de biologie, les méthodes à noyaux offrent une réponse simple et efficace pour la classification de motifs.

Les résultats obtenus dans ce mémoire reposent pour la plupart sur des constructions de transducteurs. Premièrement, on emploie des algorithmes déjà

connus, comme l'algorithme de Huffman. Cet algorithme est appliqué d'abord localement à chaque état d'une chaîne de Markov donnée. Deuxièmement, on s'inspire de la méthode connue du calcul de noyaux entre deux mots pour l'étendre à la comparaison de langages. Les noyaux sont en fait le produit scalaire des signatures des objets que l'on compare. Les signatures et les noyaux de langages sont donnés explicitement, que ce soit les algorithmes associés ou les transducteurs les calculant.

Le premier chapitre est consacré à l'exposition des structures déjà existantes : automates à poids, transducteurs et séries rationnelles réalisées par ces transducteurs. Elles seront très utiles pour calculer les noyaux de langages rationnels.

Le second chapitre est consacré au détail des transducteurs réalisant une fonction de compression d'un automate décrivant une source Markoviénne. On donne un exemple expliquant comment appliquer Huffman localement à une source de Markov et le résultat de ce transducteur. Ensuite, on présente un autre algorithme quasi optimal se basant toujours sur Huffman local ; mais cette fois-ci, à la différence, les préfixes à partir de chaque état de la source Markovienne sont d'une longueur variable de manière à obtenir un préfixe d'une probabilité d'une puissance de $\frac{1}{2}$. Dans ce chapitre on voit bien comment une source Markovienne sert de mémoire au texte déjà codé et le texte prochainement à coder, ce qui explique l'efficacité de cet algorithme. Dans ce même chapitre on calcule l'efficacité et l'entropie du transducteur compresseur et on examine la convergence de l'efficacité vers l'entropie (optimal théorique).

Dans ce deuxième chapitre on suit en détail le plan suivant pour aborder ces points.

— Dans un premier temps, nous formalisons la notion d'algorithme de compression et introduisons les compressions réalisables par transducteurs. Nous étudions les propriétés des transducteurs qui permettent une compression (et une décompression efficace). Nous illustrons ces définitions à l'aide de l'algorithme de Huffman [13].

— Nous examinons ensuite précisément ce que signifie la compression d'une source Markovienne. Nous introduisons alors l'entropie d'une telle source. Nous définissons alors l'efficacité d'un algorithme de compression pour cette source et montrons que l'entropie est un optimal pour cette efficacité. Nous montrons enfin comment évaluer l'efficacité d'un transducteur réalisant une compression.

— Nous présentons ensuite la transposition directe de l'algorithme de Huffman au cas des sources Markoviennes. Nous comparons l'efficacité d'un tel codage en fonction de la longueur des facteurs encodés.

— Enfin, nous examinons les cas dans lesquels ce codage est optimal et nous en déduisons un algorithme encodant les facteurs de longueur variable. Nous analysons enfin cet algorithme et montrons qu'il permet de s'approcher autant que souhaité d'une compression optimale.

Puis, dans la section 3.2.3 nous allons décrire comment la qualité d'un

codage effectué par un transducteur séquentiel peut être évalué par rapport à une source de Markov.

Dans le troisième chapitre, nous abordons d'abord un type particulier de noyaux, appelé *noyaux rationnels*. Ces noyaux ont été introduits dans [9] et ont été largement étudiées dans [10].

Nous généralisons les constructions introduites dans [10] et définissons les noyaux entre langages rationnels.

Plus précisément, le chapitre suit le plan suivant.

— Nous rappelons d'abord la définition de noyau et montrons comment certains noyaux peuvent être réalisés comme produits scalaires de *signatures*. Nous définissons l'extension de ces notions pour les langages.

— Nous présentons ensuite les noyaux rationnels qui sont des noyaux calculables par transducteurs ; nous transposons aussi cette notion aux signatures rationnelles. Nous montrons qu'un noyau rationnel sur les mots s'étend en un noyau rationnel pour les langages.

— Nous nous concentrons ensuite sur deux noyaux particuliers, les noyaux de facteurs et les noyaux de sous-mots. Nous étudions les conditions de convergence des séries qui interviennent dans le calcul de ces noyaux.

— Nous appliquons ces noyaux à un corpus afin d'étudier leur pertinence.

— Enfin, nous présentons quelques extensions possibles de notre travail.

Chapitre 2

Notions de Base

2.1 Structures algébriques

Les structures algébriques, semi-anneaux, automates, langages, sont largement étudiés dans [20] ou [30], on rappelle quelques notions dont on a besoin dans cette thèse.

2.1.1 Monoïde

Définition 1 *Un ensemble X, muni d'une loi associative et qui contient un élément neutre noté 1_X, est un monoïde. Lorsqu'on veut préciser que X est muni de la loi ".", on désigne le monoïde par $(X, .)$.*

Définition 2 *On désigne par alphabet un ensemble fini A non vide de symboles appelés lettres. On peut former des mots par concaténation de ces lettres. L'opération de concaténation, associative, fait de l'ensemble des mots sur A, noté A^*, un monoïde : le monoïde libre engendré par A. Le mot vide, élément neutre de ce monoide est noté 1_{A^*}. On notera A^+ l'ensemble des mots de A^* différents du mot vide. Un sous-ensemble de A^* est appelé langage. La longueur d'un mot est le nombre de lettres qu'il comporte ; on note la longueur d'un mot u par $|u|$. On notera par ailleurs u_i la i-ème lettre du mot u.*

2.1.2 Semi-anneaux

Définition 3 *Un semi-anneau est un ensemble $\mathbb{K}$, muni de deux lois associatives et d'un élément neutre pour chaque loi. La première loi $(\oplus)$ est commutative, la seconde $(\otimes)$ est distributive sur la première. On les appellera respectivement addition et multiplication. L'élément neutre de l'addition $(0_{\mathbb{K}})$ doit de plus être absorbant pour la multiplication. Dans le cas où l'on souhaite préciser quelles sont les lois du semi-anneau, on note $(\mathbb{K}, \oplus, \otimes)$. On note $\mathbb{K}_*$, l'ensemble des éléments de $\mathbb{K}$ différents de $0_{\mathbb{K}}$.*

Définition 4 *Soit $(\mathbb{K}, \oplus, \otimes)$ un semi-anneau. Pour tout x dans $\mathbb{K}$, pour tout n dans $\mathbb{N}$, x^n est defini inductivement par $x^0 = 1_{\mathbb{K}}$ et $x^n = x \otimes x^{n-1}$. On pose la*

quantité suivante, pour n dans $\mathbb{N}$:

$$X_n = \bigoplus_{k=0}^{n} x^k$$

Si X_n admet une limite dans $\mathbb{K}$ lorsque n tend vers l'infini, cette limite est appelée étoile de x et notée x^*. Dans ce cas, on note

$$x^* = \bigoplus_{k=0}^{\infty} x^k, \qquad x^+ = \bigoplus_{k=1}^{\infty} x^k.$$

Si x est un nombre réel positif inférieur à 1, on obtient :

$$x^* = \sum_{n \geqslant 0} x^n = \frac{1}{1-x} \text{ et } x^+ = \sum_{n>0} x^n = \frac{x}{1-x}.$$

2.1.3 Expression rationnelle à multiplicité

La définition des expressions rationnelles sur un alphabet A à multiplicité dans un semi-anneau $\mathbb{K}$ se fait par récurrence.

Définition 5 *Soit $\{0, 1, +, ., *\}$ un ensemble de symboles et A un alphabet.*

i) 0, 1 et a, pour tout a appartenant à A, sont des expression rationnelles (atomiques).

ii) Si E est une expression rationnelle,et k un élément de $\mathbb{K}$, alors (kE) et (Ek) sont des expression rationnelles.

iii) si E et F sont des expression rationnelles, alors $(E+F)$, $(E.F)$ et (E^*) aussi.

*Les symboles 0 et 1 sont donc des constantes, + et . sont des opérateurs binaires ; * est un opérateur unaire. On note $\mathbb{K}ExpA$ l'ensemble des expression rationnelles sur A à multiplicité dans $\mathbb{K}$.*

2.1.4 Polynômes et séries formelles

Définition 6 *Soit A un alphabet et $(\mathbb{K}, \oplus, \otimes)$ un semi-anneau. On appelle semi-anneau des séries (formelles) sur A^* à coefficients (ou à multiplicité) dans $\mathbb{K}$, qu'on note $\mathbb{K}\langle\langle A^* \rangle\rangle$, l'ensemble des applications de A^* dans $\mathbb{K}$ muni des opérations d'addition et de multiplication suivantes :*

$$\alpha \oplus \beta : x \mapsto \langle \alpha, x \rangle \oplus \langle \beta, x \rangle,$$

$$\alpha \otimes \beta : x \mapsto \bigoplus_{y.z=x} \langle \alpha, y \rangle \otimes \langle \beta, z \rangle,$$

où $\langle \alpha, u \rangle = u\alpha$ est le coefficient d'un mot u dans la série α. On note formellement la série comme une somme infinie :

$$\alpha = \bigoplus_{u \in A^*} \langle \alpha, u \rangle u.$$

Si on identifie un scalaire k de $\mathbb{K}$ à la série $k1_{A^*}$, on définit du même coup la multiplication d'une série par un scalaire .

Définition 7 *Le support d'une série s de $\mathbb{K}\langle\langle A^* \rangle\rangle$, noté $Supp(s)$ est l'ensemble des mots dont les coefficients sont non nuls.*

$$Supp(s) = \{u \in A^* : \langle s, u \rangle \neq 0_{\mathbb{K}}\}.$$

Une série dont le support est fini est un polynôme. L'ensemble des polynômes forme un sous-semi-anneau de $\mathbb{K}\langle\langle A^* \rangle\rangle$ noté $\mathbb{K}\langle A^* \rangle$.

Définition 8 *On appelle terme constant de la série s, noté $c(s)$, le coefficient du mot vide dans s : $\langle s, 1_{A^*} \rangle$. La partie propre d'une série s est la série s_p définie par :*

$$\langle s_p, 1_{A^*} \rangle = 0_{\mathbb{K}}, \forall u \in A^+, \langle s_p, u \rangle = \langle s, u \rangle.$$

On peut donc écrire $s = c(s)1_{A^*} \bigoplus s_p$. Comme dans tout semi-anneau, on peut vouloir calculer l'étoile d'une série. La définition d'une telle quantité est fortement liée à la définition de l'étoile dans le semi-anneau de coefficients.

Proposition 1 *L'étoile d'une série s de $\mathbb{K}\langle\langle A^* \rangle\rangle$ est définie si et seulement si l'étoile du terme constant de s est définie dans $\mathbb{K}$. On a alors l'égalité suivante :*

$$s^* = (c(s)^* s_p)^* c(s)^*.$$

Remarque 1 *L'étoile d'une série propre est toujours définie, en effet :*

$$\begin{aligned} \langle s_p^*, u \rangle &= \bigoplus_{u = v_1 v_2 \ldots v_n | v_1, v_2, \ldots, v_n \in A^*} \langle s_p^*, v_1 \rangle \otimes \cdots \otimes \langle s_p^*, v_n \rangle \\ &= \bigoplus_{u = v_1 v_2 \ldots v_n | v_1, v_2, \ldots, v_n \in A^+} \langle s_p^*, v_1 \rangle \otimes \cdots \otimes \langle s_p^*, v_n \rangle. \end{aligned}$$

Le membre droit de cette équation est une somme finie car il n'existe qu'un nombre fini de factorisations de u en mots non vides.

Remarque 2 *Si l'étoile est toujours définie sur le semi-anneau $\mathbb{K}$, elle est d'après la proposition, toujours définie sur $\mathbb{K}\langle\langle A^* \rangle\rangle$.*

2.1.5 Ensemble de séries rationnelles

Les séries rationnelles ont été largement abordées dans ([2]). On reprend quelques notions et définitions

Définition 9 *Soit M un monoide. L'ensemble $\mathsf{Rat}M$ des ensembles rationnels de M est la clôture des ensemble finis de M par les opérations produit, d'union et d'étoile. Ce qui revient à dire que les ensembles rationnels de M sont les élément $\mathcal{P}(M)$ rationnels par rapport aux ensembles finis. Si M est le monoide libre, on parle de langage rationnels.*

Définition 10 *Le semi-anneau des séries rationnelles $\mathbb{K}\mathsf{Rat}A^*$ de $\mathbb{K}\langle\langle A^*\rangle\rangle$ est la clôture du semi-anneau des polynômes par les opération d'addition, de multiplication et d'étoile, si cette dernière est définie.*

Exemple. Soit $A = \{a, b\}$ un alphabet. Formons un langage rationnel dans le monoide libre A^*. Le langage ab (en fait le singleton ab) est un langage fini,le langage $a + b$ aussi. le langage $(a + b)^*$ est l'étoile d'un langage fini, donc il est rationnel. En fait, il s'agit de A^* tout entier. Le langage $L1 = (a+b)^*ab(a+b)^*$ est un produit de langages rationnels ; il est donc aussi rationnel. Pour former un mot qui appartienne à ce langage, on commence par n'importe quel mot dans $(a + b)^*$ au début et un ab au milieu et la fin on met n'importe quel mot appartenant à $(a + b)^*$. On peut analyser un peut plus finement ce que représente cette description. Plaçons nous dans le cadre des séries à multiplicité dans $\mathbb{N}$. Le mot ab est alors vu comme un polynôme formé d'un seul monome dont le coefficient est 1. La série $\chi_{A^*} = (a + b)^*$ est la série caractéristique de A^*, c'est-à-dire que pour tout mot u de A^*, $\langle \chi_{A^*}, u\rangle = 1$. La série $s_1 = (a + b)^*ab(a + b)^*$ est donc une série rationnelle. La multiplicité d'un mot u dans s_1 est le nombre de façons dont u se factorise en $u = w.ab.v$, c'est donc le nombre de facteurs ab qui apparaissent dans u.

2.2 Automates

La théorie des automates et les langages formels sont largement abordés dans [36, 5, 3, 30], on rappelle quelques définitions illustrées par des exemples.

2.2.1 Automates sur un alphabet

Définition 11 *Un automate $\mathcal{A}$ est un quintuplet (Q, A, E, I, T), où Q est un ensemble fini d'états, E un sous-ensemble de $Q \times A \times Q$, appelé ensemble des transitions et I (resp.T) un sous-ensemble de Q regroupant les états initiaux (resp.terminaux). L'étiquette d'une transition (p, a, q) de E est a.*

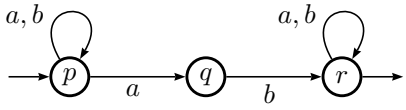

FIGURE 2.1 – Automate à trois états

EXEMPLE. Soit $\mathcal{A}_1 = (Q, A, E, I, T)$ l'automate défini par :

$$\begin{aligned} Q =& \{p, q, r\}, \\ A =& \{a, b\}, \\ E =& \{(p, a, p), (p, b, p), (p, a, q), (q, b, r), (r, a, r), (r, b, r)\}, \\ I =& \{p\}, \\ T =& \{r\}. \end{aligned}$$

La représentation graphique de cet automate est donnée Figure 2.1. Les états initiaux sont indiqués par une flèche entrante et les état terminaux par une flèche sortante. Le langage reconnu par un automate est l'ensemble de mots accepté par ce dernier, dans notre exemple il s'agit de $(a+b)^*ab(a+b)^*$.

Définition 12 *Soit $\mathcal{A} = (Q, A, E, I, T)$ un automate. Un chemin de $\mathcal{A}$ est une suite de transitions dont les images dans le graphe associé forment un chemin. L'étiquette d'un chemin est le mot formé par la concaténation des étiquettes des transitions qui le constituent. Un chemin réussi (aussi appelé calcul) de $\mathcal{A}$ est un chemin dont l'état de départ appartient à I et celui d'arrivée à T.*

2.2.2 Automates pondérés

Définition 13 *Soit $\mathbb{K}$ un semi-anneau et A un alphabet. Un $\mathbb{K}$-automate $\mathcal{A}$ sur A^* est un sextuplet $\mathcal{A} = (Q, A, \mathbb{K}, M, I, T)$, où Q est un ensemble fini d'états, M un sous-ensemble fini de $Q \times A \times \mathbb{K} \times Q$ qui sont les transitions de $\mathcal{A}$, I et T deux vecteurs de $\mathbb{K}^Q$.*
Le poids d'une transition (p, a, k, q) est k ; le poids initial (resp. final) d'un état p est I_p (resp. T_p).
Le poids d'un chemin est le produit des poids de toutes les transitions de ce chemin, le poids d'un calcul est le poids du chemin multiplié respectivement à gauche et à droite par le poids initial de l'état de départ et le poids final de l'état d'arrivée. Le poids d'un mot accepté par cet automate est la somme des poids de tous les calculs étiquetés par ce mot ; s'il n'est pas accepté par $\mathcal{A}$, son poids est fixé à zéro.

Les automates pondérés sont largement étudié dans [28, 14]. Un $\mathbb{K}$-automate $\mathcal{A}$ réalise donc une fonction des mots dans $\mathbb{K}$; cette fonction peut être vue

comme une série formelle. Cette série est alors appelée *comportement* du $\mathbb{K}$-automate, et deux $\mathbb{K}$-automates sont équivalents s'ils ont le même comportement. Pour chaque mot w dans A^*, on note $\mathcal{A}(w)$ le poids associé à w par $\mathcal{A}$.

Définition 14 *Soit $\mathcal{A} = (Q, A, \mathbb{K}, M, I, T)$ un automate pondéré. La représentation linéaire de $\mathcal{A}$ est un triplet (I, μ, T) où μ est une fonction de A^* dans $\mathbb{K}^{Q\times Q}$ telle que :*

$$\forall p, q \in Q, \forall a \in A, \mu(a)_{p,q} = k \Leftrightarrow (p, a, k, q) \in M.$$

2.3 Les Transducteurs

2.3.1 Transducteurs à états fini

Un transducteur fini est défini par un 6-uplet $\mathcal{T} = (Q, A, B, I, F, \delta)$, où :
- Q est l'ensemble fini des états du transducteur ;
- A et B sont deux alphabets ;
- I est l'ensemble des états initiaux $\subseteq Q$;
- F est l'ensemble des états terminaux $\subseteq Q$;
- δ est la table de transition : c'est un sous-ensemble de $Q \times (A \cup \{\varepsilon\}) \times (B \cup \{\varepsilon\}) \times Q$ où ε est le mot vide.

Dans cette définition, on admet les ε-transitions.

La propriété $(q, a, b, r) \in \delta$ signifie qu'il existe une transition de l'état q vers l'état r étiquetée par la paire (a, b).

Selon les applications, un transducteur sera considéré comme un accepteur qui lit une paire de mots $A^* \times B^*$. Dans ce cas, le transducteur *accepte* un ensemble de paires et reconnaît donc une relation. Toute relation reconnaissable par un transducteur fini sera appelé relation rationnelle. On notera $q \xrightarrow{(a,b)} r$ la transition d'un tel transducteur.

Dans d'autres applications, on préfère considéré le transducteur comme un automate qui lit un mot de A^* et produit un ensemble (rationnel) de mots de B^*. Dans ce cas, A est appelé alphabet d'entrée et B alphabet de sortie et le transducteur réalise une fonction de A^* dans $\mathsf{Rat}B^*$. On notera $q \xrightarrow{a|b} r$ la transition d'un tel transducteur.

Dans ce cas, il peut être intéressant de considérer le cas où cette fonction est réalisée de manière déterministe.

Définition 15 *Un transducteur est* séquentiel *s'il est déterministe par rapport à son entrée : le transducteur a un seul état initial, toute transition est étiquetée en entrée par une lettre et pour tout état p et toute lettre a, il y a au plus une transition partant de p étiquetée en entrée par a.*

Si $\mathcal{T}$ est un transducteur, pour toute paire de mots (u, v) on note $u[\mathcal{T}]v$ si la paire (u, v) est acceptée par le transducteur. De même, si u est un mot sur

l'alphabet d'entrée, on note $\mathcal{T}(u)$ l'ensemble des mots image de u par $\mathcal{T}$. On a donc

$$\mathcal{T}(u) = \{v \mid u[\mathcal{T}]v\}.$$

Composition Étant donné un transducteur $\mathcal{T}$ sur des alphabets A et B et un transducteur $\mathcal{S}$ sur des alphabets B et C, il existe un transducteur $\mathcal{S} \circ \mathcal{T}$ sur les alphabets A et C qui réalise la composition des relations réalisées par $\mathcal{T}$ et $\mathcal{S}$. La composition des transducteurs classiques se trouve par exemple dans [15]. Nous développerons cette construction dans le cas des transducteurs pondérés pour lesquelles différentes versions ont été développées dans [28, 6, 30].

2.3.2 Les transducteurs pondérés

Un transducteur pondéré est défini par un 7-uplet $\mathcal{T} = (Q, A, B, \mathbb{K}, I, F, \delta)$, où :

— Q est l'ensemble fini des états du transducteur ;
— A et B sont deux alphabets ;
— $\mathbb{K}$ est le semi-anneau avec poids ;
— I est le vecteur initial dans $\mathbb{K}^Q$;
— F est le vecteur final dans $\mathbb{K}^Q$;
— δ est la table de transition : c'est un sous-ensemble de $Q \times (A \cup \{\varepsilon\}) \times (B \cup \{\varepsilon\}) \times \mathbb{K} \times Q$ où ε est le mot vide.

De même que pour les transducteurs non pondérés, il y a deux façons de considérer les transducteurs pondérés : d'une part comme des $\mathbb{K}$-automates lisant leur entrée sur deux bandes et produisant une valeur dans $\mathbb{K}$; dans ce cas, une transition (q, a, b, k, r) du transducteur sera notée $q \xrightarrow{(a,b)|k} r$.

D'autre part, on peut le considérer comme un automate sur une bande produisant un poids dans le semi-anneau $\mathbb{K}\mathsf{Rat}B^*$ des séries rationnelles. Dans ce cas, la transition (q, a, b, k, r) du transducteur sera notée $q \xrightarrow{a|kb} r$.

On verra dans le chapitre 4 que la seconde sémantique sera utilisée pour les calculs de signatures, alors que la première permettra le calcul des noyaux.

Exemple. La figure 2.2 montre deux visions du même transducteur, d'une part comme accepteur pondéré, c'est-à-dire comme $\mathbb{K}$-automate sur $A^* \times B^*$, d'autre part comme traducteur c'est-à-dire comme $\mathbb{K}\mathsf{Rat}B^*$-automate sur A^*. Ici $A = B = \{a, b\}$.

Par exemple, le transducteur de gauche accepte la paire (aba, ab) avec le poids 2, alors que le tranducteur de droite accepte le mot aba et produit $2ab + 2ba$.

2.3.3 Composition de transducteurs pondérés

On décrit l'algorithme classique de composition des transducteurs. Étant donnés deux transducteurs finis $\mathcal{T} = (Q, A, B, I, T, \delta)$ et $\mathcal{S} = (R, B, C, J, U, \eta)$,

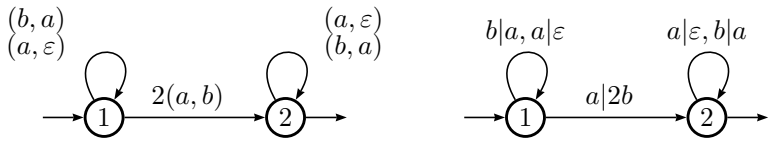

FIGURE 2.2 – Deux visions du même transducteur

on définit le transducteur $\mathcal{U} = (Q \times R, A, C, K, V, \zeta)$ de la façon suivante : [1]

$$\begin{aligned}
(p,q) \in K &\iff p \in I \text{ et } q \in J \\
(p,q) \in V &\iff p \in T \text{ et } q \in U \\
(p,q) \xrightarrow{(a,c)} (r,s) \in \zeta &\iff \exists b,\ p \xrightarrow{(a,b)} r \in \delta \text{ et } q \xrightarrow{(b,c)} s \in \eta \\
&\qquad \text{ou } p = r, a = \varepsilon \text{ et } q \xrightarrow{(\varepsilon,c)} s \in \eta \\
&\qquad \text{ou } q = s, c = \varepsilon \text{ et } p \xrightarrow{(a,\varepsilon)} r \in \delta.
\end{aligned}$$

Proposition 2 *Le transducteur $\mathcal{U}$ ainsi défini reconnaît la composition des relations réalisées par $\mathcal{T}$ et $\mathcal{S}$.*

Avec cette construction, on peut obtenir dans l'automate $\mathcal{U}$ certains calculs différents qui correspondent eux-mêmes aux paires de calculs dans $\mathcal{T}$ et $\mathcal{S}$. Ceci ne permet donc pas d'ajouter directement des poids pour que cette construction s'étendent aux relations pondérées.

Un examen attentif montre que le problème vient de la composition d'une transition $p \xrightarrow{(a,\varepsilon)} r$ de $\mathcal{T}$ avec une transition $q \xrightarrow{(\varepsilon,c)} s$ de $\mathcal{S}$. De fait, ceci donne deux chemins différents dans $\mathcal{U}$:

$$(p,q) \xrightarrow{(a,\varepsilon)} (r,q) \xrightarrow{(\varepsilon,c)} (r,s) \text{ et } (p,q) \xrightarrow{(\varepsilon,c)} (p,s) \xrightarrow{(a,\varepsilon)} (r,s).$$

Pour éviter cela, il faut interdire l'enchaînement de certaines transitions. Plusieurs constructions on été proposées dans [28, 6, 30] ; nous en présentons une qui est une hybridation de celles-ci.

Tout d'abord, on modifie le transducteur $\mathcal{T}$ de la manière suivante. Pour tout état r sur lequel arrivent à la fois des transitions avec ε et des transition avec une lettre sur la seconde bande, on remplace r par r_1 et r_2, de sorte que :

- pour toute transition $r \xrightarrow{(a,b)|k} p$, on crée des transitions $r_i \xrightarrow{(a,b)|k} p$, pour i dans $\{1, 2\}$;
- pour toute transition $p \xrightarrow{(a,\varepsilon)|k} r$, on crée une transition $p \xrightarrow{(a,\varepsilon)|k} r_1$;
- pour toute transition $p \xrightarrow{(a,b)|k} r$ $(b \neq \varepsilon)$, on crée une transition $p \xrightarrow{(a,\varepsilon)|k} r_2$;
- si r est final, r_1 et r_2 aussi (avec le même poids) ;
- si r est initial, r_2 aussi (avec le même poids).

1. Noter que dans le cas de transducteurs finis, les vecteurs booléens peuvent être assimilés à des sous-ensembles.

Le transducteur ainsi obtenu est équivalent à $\mathcal{T}$; on peut partitionner l'ensemble de ses états en Q_1 et Q_2, Q_1 étant l'ensemble des états sur lesquels on arrive après avoir lu ε sur la seconde bande et Q_2 ceux sur lesquels on arrive d'une autre manière.

À partir d'un transducteur pondérés $\mathcal{T} = (Q = Q_1 \cup Q_2, A, B, \mathbb{K}, I, T, \delta)$ vérifiant cette propriété et d'un transducteur pondéré $\mathcal{S} = (R, B, C, \mathbb{K}, J, U, \eta)$, on définit le transducteur pondéré $\mathcal{V} = (Q \times R, A, C, \mathbb{K}, V, \zeta)$ de la façon suivante :

$$\begin{aligned} K_{(p,q)} &= I_p.J_q; \\ V_{(p,q)} &= T_p.U_q; \\ (p,q) \xrightarrow{(a,c)|k} (r,s) \in \zeta \iff \exists b,\ & p \xrightarrow{(a,b)|h} r \in \delta, q \xrightarrow{(b,c)|l} s \in \eta \text{ et } k = h.l \\ & \text{ou } p = r, p \in Q_2, a = \varepsilon \text{ et } q \xrightarrow{(\varepsilon,c)|k} s \in \eta \\ & \text{ou } q = s, c = \varepsilon \text{ et } p \xrightarrow{(a,\varepsilon)|k} r \in \delta. \end{aligned}$$

Si plusieurs des conditions impliquant la création d'une transition dans $\mathcal{V}$ sont satisfaites en même temps, le poids de l'arête créée est la somme des poids induits par chaque condition.

Proposition 3 *Si l'ensemble des poids présents dans $\mathcal{T}$ et $\mathcal{S}$ commutent deux à deux, alors le transducteur $\mathcal{V}$ ainsi défini réalise la composition des relations pondérées réalisées par $\mathcal{T}$ et $\mathcal{S}$.*

Preuve. Soit (u, w) une paire de mots de $A^* \times C^*$. Montrons qu'il y a bijection entre les calculs de $\mathcal{V}$ étiquetés par (u, w) et les paires de calculs (τ, σ) de $\mathcal{T}$ et $\mathcal{S}$ pour lesquels il exite un mot v dans B^* tels que (u, v) étiquète τ et (v, w) étiquète σ. Pour cela, on supposera que les éventuelles transitions ayant mêmes extrémités et même étiquette dans $\mathcal{V}$ et créées par des conditions différentes sont dissociées (la distributivité dans le semi-anneau des poids nous garantit que ceci ne modifie pas le comportement du transducteur). Dès lors, pour chaque transition de $\mathcal{V}$, on peut savoir si elle provient de la première condition (et avec quelle lettre), ou de l'une des deux dernières.

Par récurrence sur la longueur du chemin, on peut donc reconstruire à partir d'un chemin dans $\mathcal{V}$ une unique paire de chemins dans $\mathcal{T}$ et $\mathcal{S}$ dont le produit des poids est le poids de ν. Supposons qu'à un certain chemin ν arrivant en (p, q) on associés des chemins τ et σ qui arrivent respectivement en p et q (c'est trivial si ν est de longueur nulle). Augmentons ν d'une transition $(p, q) \xrightarrow{(a,c)|k} (r, s)$ pour obtenir ν' :
– si cette transition est créée par la première condition avec une lettre b, alors les uniques chemins que l'on peut associer à ν' sont τ augmenté de $p \xrightarrow{(a,b)|h} r$ et σ augmenté de $q \xrightarrow{(b,c)|l} s$, avec $k = h.l$;
– si cette transition est créée par la seconde condition, alors les uniques chemins

que l'on peut associer à ν' sont τ lui-même et σ augmenté de $q \xrightarrow{(\varepsilon,c)|k} s$;
– si cette transition est créée par la troisième condition, alors les uniques chemins que l'on peut associer à ν' sont τ augmenté de $p \xrightarrow{(a,\varepsilon)|k} r$ et σ lui-même.
On vérifie alors que (grâce à la commutativité des poids), le produit des poids des chemins obtenu est bien égal à celui de ν'
Ainsi, à tout calcul ν de $\mathcal{V}$, on associe de manière unique une paire de calculs dont le produit des poids est égal au poids de ν.

Réciproquement, considérons τ (*resp.* σ) un calcul de $\mathcal{T}$ (*resp.* $\mathcal{S}$) étiqueté par (u, v) (*resp.* (v, w)). Notons n la longueur de v ; on peut décomposer τ en

$$p_0 \xrightarrow{(u_0,\varepsilon)|h_0} r_0 \xrightarrow{(a_1,v_1)|h'_1} p_1 \dots \xrightarrow{(a_n,v_n)|h'_n} p_n \xrightarrow{(u_n,\varepsilon)|h_n} r_n,$$

avec, pour tout i, u_i dans A^*, a_i dans $A \cup \{\varepsilon\}$ et v_i dans B. De même, on peut décomposer σ en

$$q_0 \xrightarrow{(\varepsilon,w_0)|l_0} s_0 \xrightarrow{(v_1,c_1)|l'_1} q_1 \dots \xrightarrow{(v_n,c_n)|l'_n} q_n \xrightarrow{(\varepsilon,w_n)|k_n} s_n,$$

avec, pour tout i, w_i dans C^*, c_i dans $C \cup \{\varepsilon\}$ et v_i dans B.
Pour tout i, toute transition $r_{i-1} \xrightarrow{(a_i,v_i)|h'_i} p_i$ ne peut s'apparier qu'avec une transition $s_{i-1} \xrightarrow{(v_i,c_i)|l'_i} q_i$ pour former une transition de $\mathcal{V}$ de poids $h'_i.l'_i$. Il faut donc montrer que toute paire de chemins $\tau_i = p_i \xrightarrow{(u_i,\varepsilon)|h_i} r_i$ et $\sigma_i = q_i \xrightarrow{(\varepsilon,w_i)|l_i} s_i$ ne correspond qu'à un seul chemin dans $\mathcal{V}$. Ce chemin est forcément formé de transitions provenant de l'une des deux dernières conditions.

Notons que p_i appartient à Q_2 et que tout état t suivant du chemin τ_i (y compris r_i) est dans Q_1. La seconde condition qui est la seule qui permet de "consommer" les transitions de σ_i ne peut donc être utilisée que si on n'a pas avancé dans τ. Finalement, la seule façon de simuler cette paire de chemins dans $\mathcal{V}$ est de parcourir d'abord les transitions correspondant à celles de σ_i (deuxième condition) puis les transitions correspondant à celles de τ_i (troisième condition).

On a donc montré la bijection qui implique que le comportement de $\mathcal{V}$ est bien la composition des relations pondérées réalisées par $\mathcal{T}$ et $\mathcal{S}$. □

2.3.4 Suppression des ε-transitions

Les constructions que nous utilisons sur les automates pondérés, en particulier la composition des transducteurs pondérés peut mener à la création de circuits d'ε-transitions. Dans ce cas, le nombre de calculs pour un mot (ou une paire de mots) donné n'est pas forcément fini. Nous utilisons la théorie énoncée dans [30, 31, 24, 25] qui nous permet de traiter la suppression d'ε-transition, dans le cas des automates pondérés avec des nombres réels positifs qui peuvent contenir des circuits de ε-transitions.

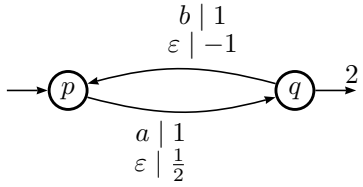

FIGURE 2.3 – Automate à ε-transition

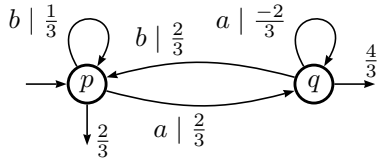

FIGURE 2.4 – suppression ε-transitions.

EXEMPLE. La figure 2.3 montre un automate avec ε-transitions ; son équivalent après suppression des ε-transitions est montré figure 2.4.

Chapitre 3

Compression Guidée par Automates

Dans ce chapitre nous nous intéresserons au codage de source très longue voire infinie, ce genre de problème a été abordé dans [18]. Pour cela on utilise la notion de compression de source de Markov comme introduit dans([8]).

3.1 Fondements

3.1.1 Compression

Etant donnés deux alphabets A et B, une fonction de codage de A^* dans B^* est une application φ de A^* dans B^*. Pour certaines applications (hachage par exemple), on n'a pas besoin que cette opération soit réversible. Nous nous intéressons ici à la compression sans perte, la fonction φ doit être inversible sur son image, elle est donc injective. Par ailleurs, on souhaite que le calcul des fonctions de compression et de décompression soit efficace.

Définition 16 *Soit A et B deux alphabets. Un algorithme de compression de A^* sur B^* est une fonction $\varphi : A^* \to B^*$ calculable et injective.*

Au sens strict, si on sait que l'on dispose d'un mot de B^* qui est dans l'image de φ, comme A^* est énumérable, l'image inverse de ce mot est calculable.

Cette définition ne dit rien de la pertinence de φ en tant qu'algorithme de compression. Par exemple, si on énumére les mots de A (par longueur, puis par ordre alphabétique), la fonction qui associe au n-éme mot le mot b^n, où b est une lettre de B est un algorithme de compression (très peu efficace).

3.1.2 Compression par transducteur

Tout encodage qui ne nécessite qu'une mémoire finie est réalisable par transducteur ; toute fonction réalisée par un transducteur est calculable ; si le transducteur est séquentiel, ce calcul est particulièrement efficace (linéaire en la taille de l'entrée).

Si la fonction de codage réalisée par le transducteur est injective, obtenir un transducteur qui réalise la fonction inverse est particulièrement facile, il suffit d'échanger l'entrée et la sortie de chacune des transitions ; on verra comment on peut prendre en compte les propriétés particulières des transducteurs qu'on construira pour obtenir un décodage efficace.

Noter que certains algorithmes de compression ne peuvent pas être réalisés par transducteur. Par exemple, le codage de Lempel-Ziv ([38, 37]) nécessite la mise à jour d'une table des préfixes et donc une mémoire infinie.

3.1.3 Codage préfixe et décodage

On généralise ici la notion de code préfixe. Classiquement un code préfixe est caractérisé par une application φ d'un alphabet A dans B^* telle que $\varphi(A)$ est un langage préfixe, c'est-à-dire que quelles que soient les lettres différentes a et b de A, les mots $\varphi(a)$ et $\varphi(b)$ ne sont pas préfixes l'un de l'autre. Le codage préfixe classique est alors l'extension de φ à A^* par morphisme : $\varphi(w_1 \dots w_k) = \varphi(w_1) \dots \varphi(w_k)$.

Définition 17 *Soit $\mathcal{T}$ un transducteur séquentiel qui réalise une application de A^* dans B^*. Le transducteur $\mathcal{T}$ est* préfixe *si, pour tout état p, quelles que soient les lettres différentes a et b de A, s'il existe des transitions (p, a, u, q) et (p, b, v, r), u et v ne sont pas préfixes l'un de l'autre.*

Comme le transducteur de codage doit accepter n'importe quelle entrée, tous les états du transducteur sont terminaux. S'il y a des transitions qui produisent le mot vide, le transducteur séquentiel ne peut donc pas réaliser une fonction injective. Pour contourner ce problème, on suppose qu'à la fin de chaque calcul, le transducteur produit un mot de longueur fixe correspondant à l'état final. Ainsi, deux mots qui ne conduisent pas dans le même état final ne peuvent pas avoir la même image.

Proposition 4 *Un transducteur* préfixe *qui ne possède aucun circuit dont la sortie est ε réalise une fonction injective.*

Preuve. Soit $\mathcal{T}$ un transducteur préfixe qui réalise une fonction φ. Supposons qu'il existe deux mots distincts u et v tels que $\varphi(u) = \varphi(v)$. Soit w le plus grand préfixe commun à u et v, et u' et v' tels que $u = wu'$ et $v = wv'$. Soit p l'état atteint en lisant w. Si $u' = \varepsilon$, soit le chemin étiqueté par v' à partir de p a une sortie différente de ε et $|\varphi(u)| < |\varphi(v)|$, soit il a une sortie égale à ε et n'arrive pas en p, donc $\varphi(u) \neq \varphi(v)$. De même si $v' = \varepsilon$. Sinon les transitions qui partent de p et qui ont pour entrées respectives la première lettre de u' et celle de v' ont des sorties qui ne sont pas préfixes l'une de l'autre, donc $\varphi(u) \neq \varphi(v)$. □

Proposition 5 *L'inverse d'une fonction réalisée par un transducteur* préfixe *est une fonction séquentielle.*

Preuve. Soit $\mathcal{T} = (Q, A, B, Q, I, F, \delta)$ un transducteur préfixe. On ignore dans cette preuve le code produit par les états terminaux ; comme il s'agit d'un code de longueur fixe qui intervient en fin de calcul, il ne remet pas en cause la séquentialité (il augmente en réalité considérablement le nombre d'états, puisqu'on doit faire une lecture en-avant de la longueur de ce code). Soit $\mathcal{T}^{-1} = (Q, B, A, Q, I, F, \delta^{-1})$ le transducteur obtenu en échangeant les entrées et sorties de chaque transition. Comme $\mathcal{T}$ est préfixe, s'il existe une ε-transition de p à q dans $\mathcal{T}^{-1}$, p n'est pas final et il n'y a pas d'autre transition qui part de p. D'ailleurs, il n'y a pas de circuit de ε-transitions dans la partie émondée du transducteur. On supprime toutes les ε-transitions par fermeture avant (*cf.* [30]) : tout chemin maximal de la forme

$$p_0 \xrightarrow{\varepsilon|a_1} p_1 \dots p_{k-1} \xrightarrow{\varepsilon|a_k} p_k \xrightarrow{u|b} q$$

est remplacé par une transition $p_0 \xrightarrow{u|a_1 \dots a_k b} q$.

Après suppression des ε-transitions, on obtient un transducteur tel que, en tout état p, l'ensemble des entrées des transitions qui partent de p forment un langage préfixe (qui n'est pas réduit au mot vide). On considère le transducteur séquentiel en forme d'arbre qui reconnaît ce langage ; la sortie de chaque transition est le mot vide, sauf la sortie qui mène à la feuille correspondant au mot u et dont la sortie est la sortie x de la transition (p, u, x, q). On remplace les transitions sortants de p par ce traducteur de la manière suivante :
– on fusionne l'état initial de ce transducteur avec p ;
– pour toute transition (p, u, x, q), on fusionne q avec la feuille correspondant à u dans l'arbre.

Le transducteur qu'on obtient est un transducteur séquentiel qui réalise l'inverse de la fonction réalisée par $\mathcal{T}$. □

3.1.4 Codage et décodage par l'algorithme de Huffman classique

On exprime dans cette partie l'algorithme classique de Huffman dans notre formalisme.

Le codage de Huffman correspond en fait à un transducteur à un seul état puisque le codage de chaque lettre se fait indépendamment des lettres précédentes.

Le codage de Huffman s'applique à une source telle que chaque lettre apparaît avec une probabilité fixe indépendante des lettres produites avant (il s'agit d'une source de Markov à 1 état).

Soit π une mesure de probabilité sur un alphabet A ; on construit un code préfixe sur B pour encoder les lettres de A, c'est-à-dire un arbre d'arité $|B|$ dans lequel chaque feuille correspond à une lettre de A. On note $B = \{b_1, \dots, b_k\}$, où k est la taille de B. On considère une file de priorité $\mathcal{F}$ qui contient des paires formées d'un nœud et d'un poids qui fixe la priorité, les nœuds de poids minimaux étant extraits en priorité de la file.

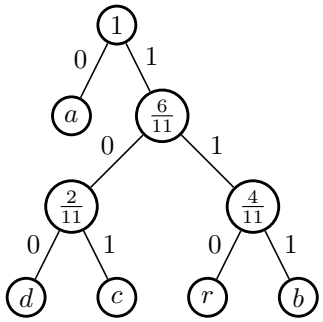

FIGURE 3.1 – Un arbre de Huffman

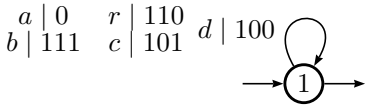

FIGURE 3.2 – Un transducteur réalisant une compression de Huffman.

On initialise la file avec les paires $(a, \pi(a))$, où on assimile toute lettre a de A à une feuille.
Tant que la file est de taille strictement supérieure à 1, on crée un noeud n, on extrait k nœuds de la file (ou tous s'il y en a moins) ; le i-ème fils de k est le i-ème nœud extrait ; on place n dans la file, avec un poids égal à la somme des poids de ses fils.

EXEMPLE. Soit $A = \{a, b, c, d, r\}$ un alphabet muni de la loi de probabilité suivante :

	a	b	c	d	r
π	$\frac{5}{11}$	$\frac{2}{11}$	$\frac{1}{11}$	$\frac{1}{11}$	$\frac{2}{11}$

Un arbre de Huffman sur un alphabet binaire B=$\{0,1\}$ obtenu à partir de cette probabilité est donné figure 3.1. Cet arbre induit un codage réalisé par le transduteur de la figure 3.2. Ce transducteur est préfixe ; la décompression est réalisée par le transducteur séquentiel de la figure 3.3.

Ainsi, le mot *abracadabra* est codé par 01111100101010001111100.

3.2 Compression avec une source de Markov

On suppose maintenant que la source est modélisée par une chaîne de Markov ; on va adapter l'algorithme de Huffman afin de tirer partie des propriétés d'une telle source. Pour évaluer l'efficacité des algorithmes que l'on propose, on utilise les outils habituels de la théorie de l'information.

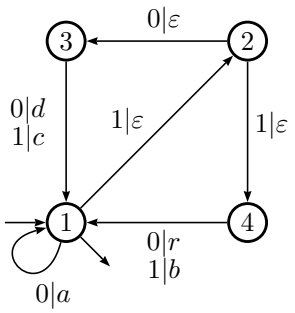

FIGURE 3.3 – Le transducteur réalisant la fonction de décompression.

3.2.1 Automates stochastiques et sources Markoviennes

Les automates probabilistes ont été largement étudiés dans [27]. Parmi eux il y a les automates stochastiques qui décrivent une chaîne de Markov. Les automates stochastiques que nous considérons dans cette thèse ne définissent pas une mesure probabiliste sur tous les mots du langage accepté. Ils correspondent à des chaînes de Markov et donc définissent une mesure probabiliste sur les mots de même longueur. Ils n'ont pas d'états terminaux, mais on peut considérer que chaque état est final avec un poids final égal à 1.

Définition 18 *Un automate stochastique sur un alphabet A est caractérisé par un ensemble fini d'états Q et une paire (I, μ), où*
- I est un vecteur dans $[0;1]^Q$, tel que

$$\sum_{p \in Q} I_p = 1;$$

- μ est un morphisme de A^* dans $[0;1]^{Q \times Q}$ tel que, pour tout $i \in Q$,

$$\sum_{j \in Q} \sum_{a \in A} \mu(a)_{i,j} = 1.$$

Pour tout mot w, la probabilité associée à w par cet automate est

$$\mathsf{prob}(w) = \sum_{i \in Q} (I\mu(w))_i.$$

Remarque 3 *Dans un automate stochastique, la somme des poids de toutes les transitions sortant de chaque état est égale à 1.*

Proposition 6 *soit $\mathcal{A} = (I, \mu)$ un automate stochastique . Alors,*
1) pour tout mot w, $\mathsf{prob}(w) \in [0;1]$;
2) pour tout mot w,

$$\mathsf{prob}(w) = \sum_{a \in A} \mathsf{prob}(wa);$$

3) pour tout n dans $\mathbb{N}$,

$$\sum_{w \in A^n} \mathsf{prob}(w) = 1.$$

Preuve. Si w est le mot vide, $I\mu(w) = I$, donc

$$\mathsf{prob}(w) = \sum_{i \in Q} I_i = 1.$$

cela montre 1) pour le mot vide, tout comme 3) pour $n = 0$.

pour tout mot w, on a :

$$\begin{aligned}
\sum_{a \in A} \mathsf{prob}(wa) &= \sum_{a \in A} \sum_{i \in Q} (I\mu(wa))_i \\
&= \sum_{a \in A} \sum_{i,j \in Q} (I\mu(w))_j \mu(a)_{j,i} \\
&= \sum_{j \in Q} (I\mu(w))_j (\sum_{a \in A} \sum_{i \in Q} \mu(a)_{j,i}) \\
&= \sum_{j \in Q} (I\mu(w))_j \\
&= \mathsf{prob}(w).
\end{aligned}$$

□

Remarque 4 *L'interprétation de* $\mathsf{prob}(w)$ *est la probabilité du mot w parmi les mots de longueur n dans un langage. Notez que cette définition diffère de définitions usuelles des automates probabilistes. La définition de Rabin dans ([27]) implique une probabilité d'acceptation pour chaque mot. En ([11],[12]) Mohri définit les automates probabilistes qui définissent une probabilité sur A^* alors que notre définition donne une probabilité sur A^n pour chaque n. Cette différence vient du fait que nous nous intéressons au comportement des mots longs, tandis que le calcul du noyau dépend largement des petits mots.*

Une source Markovienne est décrite par un automate stochastique (I, μ). A chaque étape, l'état dans lequel on se trouve est une variable aléatoire donnée par une probabilité λ (initialisée à I). La lettre a est produite avec probabilité $p_a = \lambda\mu(a)\mathbb{1}$ (où $\mathbb{1}$ est le vecteur d'observation où chaque composante vaut 1), et la probabilité de l'état où on se trouve est $p_a^{-1}\lambda\mu(a)$.

On considérera surtout dans la suite des sources Markoviennes décrites par des automates stochastiques déterministes, l'état initial est alors unique et la connaissance de la lettre produite permet de savoir avec certitude dans quel état se retrouve la source Markovienne.

3.2.2 Calcul de l'entropie d'une source Markovienne

Dans [35], Shannon introduit la notion d'entropie qui permet de mesurer la quantité d'information contenu dans un texte ou, dans notre cas, dans une source produisant aléatoirement des mots. Si à chaque instant la source est déterministe, c'est-à-dire si, à partir des lettres déjà produites, on peut prédire avec certitude quelle sera la lettre suivante, il suffit pour encoder un mot de connaître sa longueur ; d'une certaine manière, la quantité d'information apportée par chaque lettre est nulle. A l'opposé, si chaque lettre est produite avec une probabilité uniforme indépendament des lettres produites précédemment, l'information apportée par chaque lettre est maximale et on ne peut pas décrire d'algorithme pour lequel l'espérance du taux de compression est bonne.

L'entropie d'une source d'information est intuitivement le meilleur taux de compression que l'on peut obtenir pour cette source. On formalisera cette intuition dans l'énoncé du théorème 1.

Définition 19 *Soit s une source Markovienne s définie par un automate stochastique $\mathcal{A} = (I, \mu)$ sur un alphabet A. L'entropie de s est :*

$$E(s) = \lim_{n\to\infty} -\frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w) \log \mathsf{prob}(w).$$

Notons que dans cette définition comme dans toute cette thèse, log désigne le logarithme de base 2. On définit l'efficacité d'un algorithme de compression d'une source Markovienne de la manière suivante.

Définition 20 *Soit A et B deux alphabets.*[1] *Soit φ un algorithme de compression de A^* dans B^* et soit s une source Markovienne sur A. L'efficacité de l'algorithme φ pour compresser la source s, $\mathsf{eff}(\varphi, s)$, est la limite supérieure, quand n tends vers l'infinie, du taux de compression des mots de longueur n :*

$$\begin{aligned}\mathsf{eff}(\varphi, s) &= \limsup_{n\to\infty} \sum_{w\in A^n} \mathsf{prob}(w) \frac{|\varphi(w)|}{|w|} \\ &= \limsup_{n\to\infty} \frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w)|\varphi(w)|.\end{aligned}$$

On montre dans le théorème suivant que l'entropie est une borne inférieure pour l'efficacité de tout algorithme de compression.

Théorème 1 *Soit s une source Markovienne sur A et soit $\varphi : A^* \longrightarrow B^*$ un algorithme de compression. Alors,*

$$\mathsf{eff}(\varphi, s) \log |B| \geqslant E(s).$$

1. On suppose ici et dans toute la suite que les alphabets ont au minimum 2 lettres.

Lemma 1 *Soit φ une fonction de compression de A^* dans B^*. Alors,*

$$\liminf_{n\to\infty} \sum_{w\in A^n} \frac{1}{|B|^{|\varphi(w)|}} \leqslant \frac{\log|A|}{\log|B|}.$$

Preuve. On utilise le lemme de Cesaro pour obtenir le résultat :

$$\liminf_{n\to\infty} \sum_{w\in A^n} \frac{1}{|B|^{|\varphi(w)|}} \leqslant \lim_{n\to\infty} \frac{1}{n} \sum_{w\in A^{<n}} \frac{1}{|B|^{|\varphi(w)|}}.$$

Pour tout n, le nombre de mots plus petits que n est $\frac{|A|^n-1}{|A|-1}$, et pour tout k, le nombre de mots dans B plus petit que k est $\frac{|B|^k-1}{|B|-1}$.

Alors, le plus petit k_n tel que $\{B^{<k_n}\}$ contient le même nombre de mots que $\{A^{<n}\}$ satisfait

$$k_n \geqslant \left\lfloor \frac{\log\left(\frac{|B|-1}{|A|-1}(|A|^n-1)+1\right)}{\log|B|} \right\rfloor.$$

D'où, $\lim_{n\to\infty}(\frac{k_n}{n}) = \frac{\log|A|}{\log|B|}$. Or φ est injective, la meilleure compression de tous les mots plus petits que n utilise les mots dans B plus petits que k_n :

$$\lim_{n\to\infty} \frac{1}{n} \sum_{w\in A^{<n}} \frac{1}{|B|^{|\varphi(w)|}} \leqslant \lim_{n\to\infty} \frac{1}{n} \sum_{u\in|B|^{<k_n}} \frac{1}{|B|^{|u|}} = \lim_{n\to\infty} \frac{k_n}{n} = \frac{\log|A|}{\log|B|}.$$

□

Preuve. [Théorème 1]

$$\begin{aligned}
&E(s) - \mathsf{eff}(\varphi, s)\log|B| \\
&= \liminf_{n\to\infty} \left(-\frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w)\log\mathsf{prob}(w) - \frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w)|\varphi(w)|\log|B| \right) \\
&= \liminf_{n\to\infty} \frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w) \left(\log\frac{1}{\mathsf{prob}(w)} + \log\frac{1}{|B|^{|\varphi(w)|}} + \log\frac{\log|B|}{\log|A|} \right) \\
&= \liminf_{n\to\infty} \frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w)\log\frac{\log|B|}{\mathsf{prob}(w)(\log|A|)|B|^{|\varphi(w)|}} \\
&\leqslant \liminf_{n\to\infty} \frac{1}{n\ln 2} \sum_{w\in A^n} \mathsf{prob}(w) \left(\frac{\log|B|}{\mathsf{prob}(w)(\log|A|)|B|^{|\varphi(w)|}} - 1 \right) \\
&= \liminf_{n\to\infty} \frac{1}{n\ln 2} \sum_{w\in A^n} \left(\frac{\log|B|}{(\log|A|)|B|^{|\varphi(w)|}} - \mathsf{prob}(w) \right) \\
&= \liminf_{n\to\infty} \frac{\log|B|}{n\ln|A|} \left(\Big(\sum_{w\in A^n} \frac{1}{|B|^{|\varphi(w)|}} \Big) - \frac{\log|A|}{\log|B|} \right) \leqslant 0.
\end{aligned}$$

□

3.2.3 Calcul de l'efficacité d'un codeur rationnel

Définition 21 *Un codeur rationnel pour une source est un transducteur séquentiel qui accepte au moins chaque mot que la source peut produire avec une probabilité non nulle.*

Lemma 2 *Soit $\mathcal{A} = (I, \mu)$ un automate stochastique décrivant une source Markovienne, et soit $\mathcal{T}$ un transducteur qui réalise ce codage. L'efficacité de $\mathcal{T}$ pour la source décrite par $\mathcal{A}$ est la limite, quand n tend vers l'infini, de l'espérance du rapport de la longueur des images des mots de longueur n :*

$$\mathsf{eff}(\mathcal{T}) = \lim_{n\to\infty} \frac{1}{n} \sum_{w\in A^n} |\mathcal{A}(w)|.|\mathcal{T}(w)|.$$

Pour étudier l'efficacité d'un tranducteur, on utilise le semi-anneau $\mathbb{K} = \mathbb{R} \times \mathbb{R}_+^* \cup \mathsf{0}$ défini comme suit :

— $\mathsf{0}$ est le zéro du semi-anneau ;

— pour chaque (x_1, p_1) et (x_2, p_2) dans $\mathbb{R} \times \mathbb{R}_+$, la somme ($\oplus$) et le produit ($\otimes$) sont definis comme suit :

$$\begin{aligned}(x_1, p_1) \oplus (x_2, p_2) &= \left(\frac{p_1x_1 + p_2x_2}{p_1 + p_2}, p_1 + p_2\right)\\ (x_1, p_1) \otimes (x_2, p_2) &= (x_1 + x_2, p_1p_2);\end{aligned}$$

L'unité du semi-anneau est donc $(0, 1)$. Intuitivement, une paire (x, p) est une valeur x associée à la probabilité p, et l'élément $\mathsf{0}$ a une valeur indéfinie avec une probabilité nulle.

Soit π (*resp.* κ) la projection canonique de $\mathbb{K}$ dans $\mathbb{R}_+$ (*resp.* $\mathbb{R}$) : $\pi(x, p) = p$ et $\kappa(x, p) = x$. On pose $\pi(\mathsf{0}) = \kappa(\mathsf{0}) = 0$.

On considère le graphe orienté $\mathcal{E}(\mathcal{T}, \mathcal{A})$ pondéré par $\mathbb{K}$ dont chaque sommet est une paire formée d'un état s de $\mathcal{A}$ et d'un état t de $\mathcal{T}$.

Ce graphe simule l'application de $\mathcal{T}$ au fur et à mesure que la source $\mathcal{A}$ produit des lettres. Ainsi, si la source se trouve dans l'état s et le transducteur dans l'état t et que la source produit à cet instant la lettre a et passe dans l'état s' avec la probabilité $\mu(a)_{s,s'}$, le transducteur doit avoir une transition partant de t et d'entrée a ; si la sortie de cette transition est u et qu'elle mène en t', alors, avec probabilité $\mu(a)_{s,s'}$, le système source/transducteur produit $|u|$ lettres avec probabilité $\mu(a)_{s,s'}$.

Finalement, le poids de l'arc (s, t) vers (s', t') est égal à

$$\bigoplus_{\substack{a|u\\ t\longrightarrow t'}} (|u|, \mu(a)_{s,s'}).$$

Exemple. Considérons la source et le transducteur de la figure 3.4. Le graphe $\mathcal{E}(\mathcal{T}, \mathcal{A})$ est présenté figure 3.5. Par exemple, dans le sommet $(1, p)$, la source

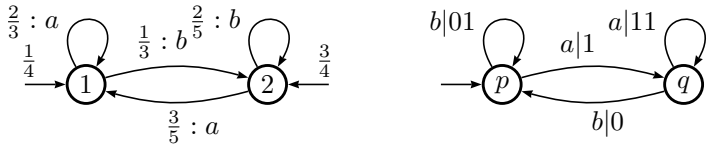

FIGURE 3.4 – Une source Markovienne et un transducteur de codage

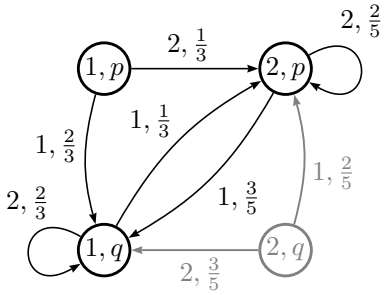

FIGURE 3.5 – Le graphe décrivant l'efficacité du transducteur sur la source de la figure 3.4

produit un a avec probabilité $\frac{2}{3}$ et reste en 1 ; le transducteur produit le mot 1 et passe en q, donc un arc part vers $(1, q)$ avec poids $(1, \frac{2}{3})$. Le sommet $(2, q)$ n'est pas accessible à partir des sommets qui correspondent à des états initiaux.

Le poids le long d'un chemin de ce graphe est le produit des poids, on obtient donc une paire (x, p) où x est la somme des longueurs et p le produit des probabilités.

Lemma 3 *Soit $\mathcal{A} = (I, \mu)$ une source Markovienne et $\mathcal{T}$ un transducteur séquentiel encodant cette source. Soit M la matrice de transition du graphe $\mathcal{E}(\mathcal{T}, \mathcal{A})$. Pour tout $n > 0$, pour toute paire d'états p et q du graphe $\mathcal{E}(\mathcal{T}, \mathcal{A})$, soit $k = M^n_{p,q}$; $\pi(k)$ est la probabilité de passer en q en n étapes si on part de p et $\kappa(k)$ est l'espérance du nombre de lettres produites lors de ce passage.*

La preuve est une récurrence simple sur n.

Notations. Pour toute matrice X à coefficients dans $\mathbb{K}$, on note $\pi(X)$ et $\kappa(X)$ les projections entrée par entrée de X respectivement par π et κ. Pour 2 matrices réelles X et Y de même taille, on note $X \odot Y$ la multiplication composante par composante.
Par ailleurs, on note $\mathbb{1}$ le vecteur colonne dont chaque entrée vaut 1.

Lemma 4 *Soit $\mathcal{A} = (I, \mu)$ une source Markovienne et $\mathcal{T}$ un transducteur séquentiel encodant cette source. Soit i l'état initial de $\mathcal{T}$ et $\mathcal{I}$ le vecteur indicé par les paires d'états de $\mathcal{A}$ et $\mathcal{T}$ tel que $\mathcal{I}_{(s,t)} = I_s$ si $t = i$ et est nul sinon. L'espérance de la longueur du code associée à un mot de longueur n produit par la source est $\mathcal{I}\kappa(M^n)\mathbb{1}$.*

Finalement, l'efficacité de $\mathcal{T}$ pour encoder la source $\mathcal{A}$ est donc

$$\lim_{n\to\infty} \frac{1}{n}\mathcal{I}\kappa(M^n)\mathbb{1}.$$

Pour estimer cette quantité, on considère une probabilité λ invariante pour la matrice stochastique $\pi(M)$: $\lambda\pi(M) = \lambda$ et on regarde l'espérance de la production sous cette probabilité. Formellement, on considère le vecteur propre (à gauche) $(k, \lambda) = (k_i, \lambda_i)_i \in I$ où I est la dimension de M ; ce vecteur propre correspond à une valeur propre $(\alpha, 1)$:

$$(k, \lambda) \otimes M = (k + \alpha, \lambda).$$

La valeur α est donc l'efficacité que l'on recherche. Voyons comment on peut calculer α à partir de la matrice M. Supposons qu'on a calculé λ à partir de $\pi(M)$; alors

$$\begin{aligned}
&\forall j \in I,\ \bigoplus_{i\in I}(k_i, \lambda_i) \otimes (\kappa(M)_{i,j}, \pi(M)_{i,j}) = (\alpha + k_j, \lambda_j),\\
\Rightarrow &\forall j \in I,\ \bigoplus_{i\in I}(k_i + \kappa(M)_{i,j}, \lambda_i\pi(M)_{i,j}) = (\alpha + k_j, \lambda_j)\\
\Rightarrow &\forall j \in I,\ \frac{\sum_{i\in I} \lambda_i\pi(M)_{i,j}(k_i + \kappa(M)_{i,j})}{\lambda_j} = \alpha + k_j\\
\Rightarrow &\forall j \in I,\ \sum_i(\lambda_i\pi(M)_{i,j}\kappa(M)_{i,j}) + \sum_i \lambda_i\pi(M)_{i,j}k_i = \lambda_j\alpha + \lambda_j k_j\\
\Rightarrow &\lambda(\pi(M) \odot \kappa(M)) + (\lambda \odot k)\pi(M) = \alpha\lambda + \lambda \odot k\\
\Rightarrow &\lambda(\pi(M) \odot \kappa(M)) + (\lambda \odot k)(\pi(M) - \mathsf{Id}) = \alpha\lambda\\
\Rightarrow &\lambda(\pi(M) \odot \kappa(M))\mathbb{1} + (\lambda \odot k)(\pi(M) - \mathsf{Id})\mathbb{1} = \alpha\lambda\mathbb{1} = \alpha\\
\Rightarrow &\lambda(\pi(M) \odot \kappa(M))\mathbb{1} = \alpha \qquad \text{car } \pi(M)\mathbb{1} = \mathbb{1}.
\end{aligned}$$

Ce résultat peut être interprété comme suit. La longueur prévue de la sortie d'une transition, en supposant que la transition commence à l'état i, est $\sum_j \pi(M)_{i,j}\kappa(M)_{i,j}$; l'efficacité du transducteur est l'espérance de cette valeur, et la probabilité d'être dans l'état i après un grand nombre d'étapes est de λ_i.

3.3 Codage par blocs de longueur fixe

3.3.1 Algorithme de Huffman local

Pour tirer parti de la connaissance de la source Markovienne, une première idée consiste à adapter le codage selon l'état de la source dans lequel on se trouve.

Soit $\mathcal{A} = (I, \mu)$ une source Markovienne. On suppose que la source Markovienne est déterministe :

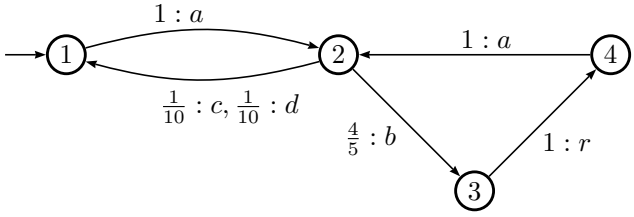

FIGURE 3.6 – L'automate stochastique $\mathcal{A}_1$

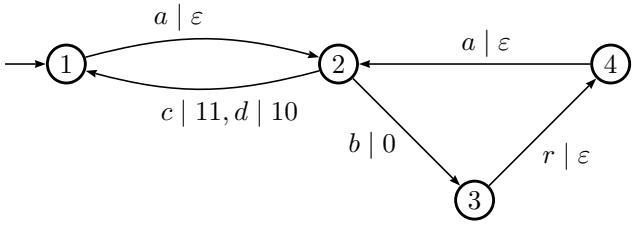

FIGURE 3.7 – Le transducteur sequentiel $\mathcal{T}_1$

— elle a un seul état initial (I est un vecteur caractéristique) ;
— en chaque état, pour chaque lettre, il n'y a au plus une transition sortante portant cette étiquette.

Nous construisons un transducteur séquentiel $\mathcal{T}$ avec le même ensemble d'états Q que $\mathcal{A}$. L'état initial de $\mathcal{T}$ est le même que l'état initial de $\mathcal{A}$.

Pour tout état p de $\mathcal{A}$, l'ensemble des transitions partant de p induit une distribution de probabilité sur l'alphabet : si (p, a, k, q) est une transition, la lettre a a la probabilité k (si la lettre a n'étiquète aucune des transitions partant de p, elle n'est pas encodée). Soit ϕ_p un code de Huffman obtenu à partir de cette distribution de probabilité. Pour toute transition (p, a, k, q) de $\mathcal{A}$, on construit la transition $(p, a, \phi_p(a), q)$ dans $\mathcal{T}$.

Le transducteur obtenu est un transducteur préfixe. La fonction qu'il réalise peut donc être inversée par un transducteur séquentiel.

EXEMPLE. Soit $\mathcal{A}_1$ l'automate de la Figure 3.6. Il y a trois transitions sortantes de l'état 2, qui sont étiquetées respectivement par c, d et b et ont une probabilité respective de $\frac{1}{10}$, $\frac{1}{10}$ et $\frac{4}{5}$. L'algorithme de Huffman appliqué à cet ensemble pondéré renvoie un code à deux lettres pour les deux premières transitions et un code à une lettre pour la dernière. Notez que dans le cas où une seule transition sort d'un état, le code associé est le mot vide. La procédure donne donc le transducteur $\mathcal{T}_1$ de la Figure 3.7.

Si on néglige le problème des marqueurs d'états terminaux, le transducteur décodeur est celui de la figure 3.8. Considérons par exemple le codage et le décodage d'un mot commençant par *abracadabra* :

a	b	r	a	c		a	d		a	b	r	a	$\ldots$
	0			1	1		1	0		0			
	ab			ra	c		a	d		ab	$\ldots$		

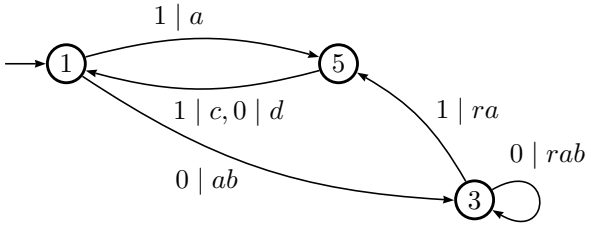

FIGURE 3.8 – Le transducteur décodeur de $\mathcal{T}_1$

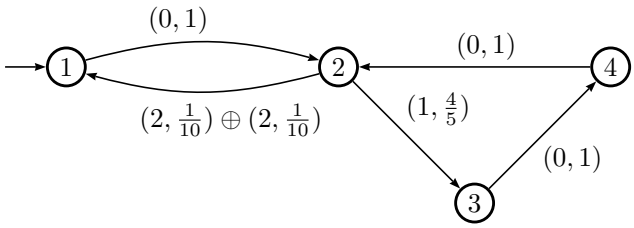

FIGURE 3.9 – Le graphe calculant l'efficacité $\mathcal{T}_1$ de la source $\mathcal{A}_1$

Calculons maintenant l'efficacité de ce codage. Le graphe de la Figure 3.9 permet de la calculer.

Soit M_1 la matrice de transition de ce graphe :

$$M_1 = \begin{bmatrix} 0 & (0,1) & 0 & 0 \\ (2,\frac{1}{5}) & 0 & (1,\frac{4}{5}) & 0 \\ 0 & 0 & 0 & (0,1) \\ 0 & (0,1) & 0 & 0 \end{bmatrix}$$

Une mesure invariante pour $\pi(M_1)$ est $\lambda = [\frac{1}{14}, \frac{5}{14}, \frac{4}{14}, \frac{4}{14}]$. Avec les notations précédentes, on obtient :

$$\lambda(\pi(M_1) \odot \kappa(M_1))\mathbb{1} = \lambda \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{2}{5} & 0 & \frac{4}{5} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbb{1} = \frac{3}{7}.$$

L'espérance de la longueur du code sur l'alphabet $\{0,1\}$ représentant un mot de longueur n produit par la source est donc équivalente à $\frac{3n}{7}$ lorsque n est grand.

3.3.2 Algorithme de Huffman d'ordre k

L'algorithme est le même que celui d'ordre 1 (Huffman local) mais au lieu de considérer les lettres une par une, on les regroupe par blocs de k lettres et on applique le code de Huffman local sur ces blocs de k lettres.

À partir de l'automate $\mathcal{A}$ associée à la source Markovienne, un automate $\mathcal{A}_\mathcal{K}$ est construit selon la règle suivante : les états de $\mathcal{A}_\mathcal{K}$ sont des copies des états de $\mathcal{A}$, et pour tout chemin $(s_0, a_1, p_1, s_1)(s_1, a_2, p_2, s_2) \dots (s_{k-1}, a_k, p_k, s_k)$

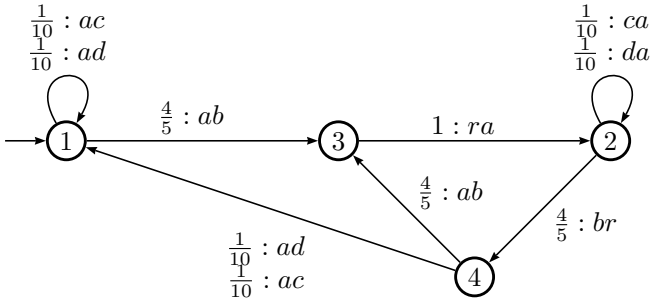

FIGURE 3.10 – Un automate d'ordre 2 decrivant la source Markovienne

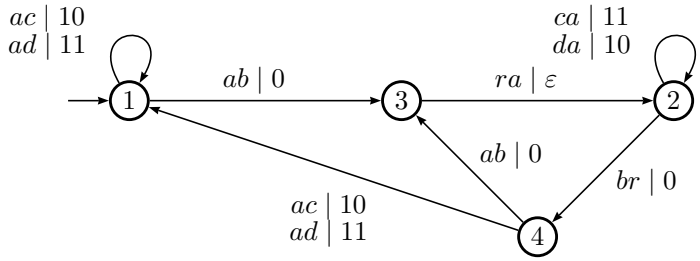

FIGURE 3.11 – Le transducteur d'ordre 2 encodant la source Markovienne.

de longueur k dans $\mathcal{A}$, il y a une transition à partir de s_0 à s_k en $\mathcal{A}_\mathcal{K}$, avec l'étiquette $a_1 a_2 \dots a_k$ et la probabilité $\Pi_{i=1}^{k} p_i$.

EXEMPLE. A partir de la source $\mathcal{A}_1$ de l'exemple précédent, on peut construire une source engendrant des blocs de longueur 2 ; elle est décrite figure 3.10.

Le transducteur encodant la source est dessiné sur la figure 3.11.

Pour calculer l'efficacité du codage, on considère le graphe de la figure 3.12 dont la matrice de transition est

$$M_2 = \begin{bmatrix} (2, \frac{1}{5}) & 0 & (1, \frac{4}{5}) & 0 \\ 0 & (2, \frac{1}{5}) & 0 & (1, \frac{4}{5}) \\ 0 & (0, 1) & 0 & 0 \\ (2, \frac{1}{5}) & 0 & (1, \frac{4}{5}) & 0 \end{bmatrix}$$

On constate que $\pi(M_2) = \pi(M_1)^2$ et que ces matrices admettent donc la même mesure invariante λ.

Finalement, on obtient

$$\alpha_2 = \lambda(\pi(M_2) \odot \kappa(M_2)\mathbb{1} = \lambda \begin{bmatrix} \frac{2}{5} & 0 & \frac{4}{5} & 0 \\ 0 & \frac{2}{5} & 0 & \frac{4}{5} \\ 0 & 0 & 0 & 0 \\ \frac{2}{5} & 0 & \frac{4}{5} & 0 \end{bmatrix} \mathbb{1} = \frac{6}{7}.$$

Ceci est l'efficacité si on considère chaque groupe de 2 lettres comme ayant une longueur égale à 1. Si on considère la longueur réelle, ceci divise l'efficacité par 2 et on retrouve $\frac{3}{7}$. On verra par la suite que pour l'ordre 3, l'efficacité diminue. Notons toutefois, que le transducteur, même s'il n'a pas plus d'états a potentiellement beaucoup plus de transitions.

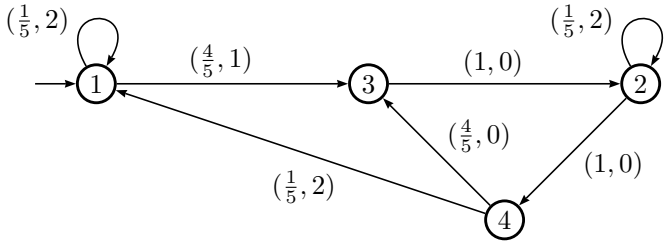

FIGURE 3.12 – L'automate d'ordre 2 calculant l'efficacité (le taux) de compression associé à la chaîne *abracadabra*.

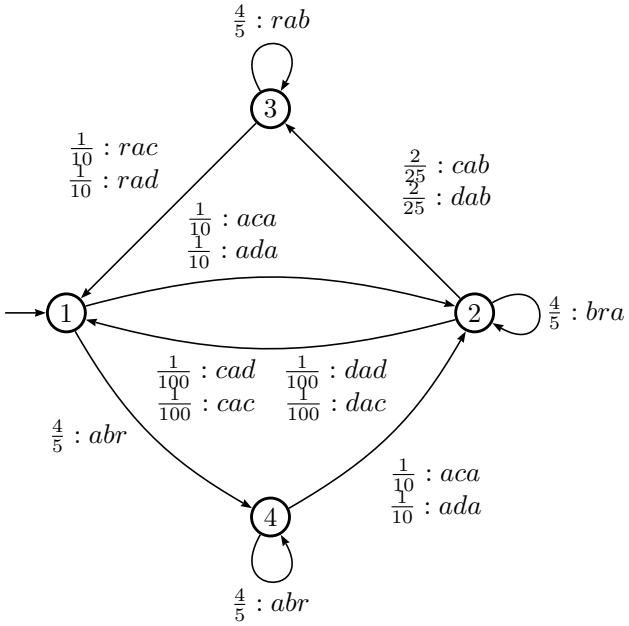

FIGURE 3.13 – Un automate d'ordre-3 décrivant une source Markoviènne

Refaisons les calculs à l'ordre 3. La source Markovienne correspondante est dessinée figure 3.13.

Le transducteur résultat est dessiné dans la Figure 3.14.

La matrice de transition du graphe d'efficacité est :

$$M_3 = \begin{bmatrix} 0 & (2, \frac{1}{5}) & 0 & (1, \frac{4}{5}) \\ (5, \frac{1}{25}) & (1, \frac{4}{5}) & (\frac{5}{2}, \frac{4}{25}) & 0 \\ (2, \frac{1}{5}) & 0 & (1, \frac{4}{5}) & 0 \\ 0 & (2, \frac{1}{5}) & 0 & (1, \frac{4}{5}) \end{bmatrix}$$

La mesure invariante associée à la matrice $\pi(M_3)$ est le même vecteur

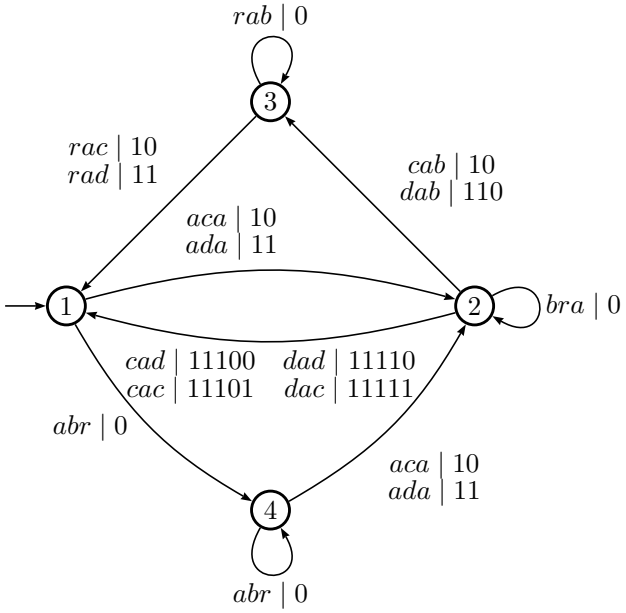

FIGURE 3.14 – Un transducteur d'ordre 3 encodant la source Markovienne

propre. On obtient

$$\alpha = \lambda(\pi(M_3) \odot \kappa(M_2)\mathbb{1} = \lambda \begin{bmatrix} 0 & \frac{2}{5} & 0 & \frac{4}{5} \\ \frac{1}{5} & \frac{4}{5} & \frac{2}{5} & 0 \\ \frac{2}{5} & 0 & \frac{4}{5} & 0 \\ 0 & \frac{2}{5} & 0 & \frac{4}{5} \end{bmatrix} \mathbb{1} = \frac{89}{70}.$$

Sachant qu'on groupe les lettres par 3, l'efficacité est donc $\frac{89}{210}$, ce qui est un gain très faible par rapport à $\frac{3}{7} = \frac{90}{210}$.

3.4 Codage par blocs de longueur variable

3.4.1 Codage parfait

On considère dans cette section des conditions qui permettent d'obtenir un codage optimal, c'est-à-dire dont l'efficacité est égale à l'entropie de la source. Dans toute la suite, A est l'alphabet des mots produits par la source et b est la taille de l'alphabet de sortie.

Définition 22 *Soit $\mathcal{A} = (I, \mu)$ une source Markovienne. Un transducteur T réalise un codage parfait de $\mathcal{A}$ si* $\log_b |A|.\mathsf{eff}(T) = E(\mathcal{A})$.

Intuitivement, si pour tout n suffisament grand, tout mot w de longueur n est encodé par un mot de longueur $-\log_b \mathsf{prob}(w)$ (à une constante près indépendante de n), alors le codage est parfait.

Proposition 7 *Soit (I, μ) une source de Markov. Supposons que pour tout état p, il existe un ensemble fini de mots X tel que :*

- Tout mot de X étiquette un chemin partant de p avec une probabilité puissance de $\frac{1}{b}$ où b est la taille de l'alphabet de sortie.
- Tout chemin partant de p est préfixe d'un chemin étiqueté par un mot de X ou admet un tel chemin comme préfixe.
Alors il existe un codage parfait par un transducteur préfixe pour (I, μ).

Preuve. On considére la suite des mots de X, $(x_1, x_2, \ldots, x_k)$, triés dans l'ordre décroissant de leur probabilité. On prouve par récurrence sur k qu'il existe un codage préfixe φ encodant X tel que la longueur de $\varphi(x_i)$ est $-\log_b \mathsf{prob}(x_i)$. Si $k = 1$, X a un seul mot de probabilité 1 et on a vu que dans ce cas, on encode x_1 par le mot vide. Si $k > 1$, supposons qu'on a montré l'existence d'un codage pour tout ensemble de taille strictement inférieure à k. Comme les probabilités des mots de X sont des puissances de $\frac{1}{b}$ et que la somme des probabilités est 1, les b derniers mots de la suite ont une même probabilité $\frac{1}{b^r}$. Considérons une suite de mots $X' = (x_1, x_2 \ldots x_{k-b}, y)$ dans laquelle chaque x_i a la même probabilité que dans X et y a la probabilité $\frac{1}{b^{r-1}}$.

Par hypothèse de récurrence, il existe un codage φ' préfixe de X' ; on en déduit un codage pour X : $\varphi(x_i) = \varphi'(x_i)$ si $i \leqslant k - b$ et $\varphi(x_i) = \varphi(y)\beta_{i-(k-b)}$ sinon, où β_j est la j-ème lettre de l'alphabet de codage.

Pour chaque état p, on peut donc trouver un code préfixe (noter que ce code est construit par une procédure de Huffman à partir de X et tout mot produit par la source peut être découpé (sauf une partie bornée à la fin du mot) de manière à être encodé par ce code préfixe. Le code associé à un mot est donc, à une constante près, de longueur égale à l'opposé du logarithme de sa probabilité. □

En pratique, les probabilités des facteurs engendrés par la source ne sont pas des puissances de $\frac{1}{b}$; on va donc chercher des facteurs dont la probabilité s'approche de cette probabilité parfaite pour obtenir un bon codage.

3.4.2 Algorithme de Huffman adaptatif

Pour chaque état p de la source, on veut calculer un ensemble de mots X qui étiquètent un ensemble préfixe de tous les chemins partant de p et dont les probabilités sont proches de puissances de $\frac{1}{b}$.

On fixe une erreur $\varepsilon > 0$ et on dira qu'un chemin étiqueté par w a une probabilité π acceptable s'il existe un entier k tel que $|-\log_b \pi - k| < \varepsilon$.

La procédure pour calculer les chemins est donc la suivante :

— on place toutes les transitions partant de p dans une file ;
— tant que la file n'est pas vide, on sort un chemin de la file ; s'il a une probabilité acceptable, on le place dans l'ensemble des chemins acceptés, sinon, on prolonge le chemin de toutes les transitions possibles et on remet les chemins ainsi créés dans la file.

```
HuffmanAdaptatif_1 de A = (I, μ), p état de A et ε > 0
    Résultat := ∅
    f : file vide
    pour toute transition (p, a, k, q)
        f ⟵ (a, k, q)
    tant que f ≠ ∅
        (w, k, q) :=pop(f)
        x := − log_b k ; y := ⌊x⌋
        si x − y < ε|w| ou y + 1 − x < ε|w|
            Résultat ⟵ (a, k, q)
        sinon
            pour toute transition (q, a, l, r)
                f ⟵ (wa, k.l, r)
    retourner Résultat
```

FIGURE 3.15 – Procédure de calcul des facteurs ayant une probabilité acceptable

Noter que si on remplace la probabilité réelle de chaque chemin sélectionné par la puissance de $1/b$ dont elle est proche, on obtient un ensemble de valeurs dont la somme n'est pas forcément égale à 1, comme on va le voir dans l'exemple suivant.

EXEMPLE. Considérons la source de la figure 3.6. À partir de chaque état, on essaie de trouver des chemins dont la probabilité est proche d'une puissance de $\frac{1}{2}$. Pour l'état 1 (ou 3 ou 4), il n'y a pas de problème : la seule transition qui en part a une probabilité égale à 1. Pour l'état 2, avec $\varepsilon = 0,01$, on trouve

les chemins suivants :

mot	état d'arrivée	probabité p	$-\log_2 p$	code
brabrab	3	$\frac{64}{125}$	$0,97$	1
brabrac	1	$\frac{16}{250}$	$3,97$	0010
brabrad	1	”	”	0011
bracab	3	”	”	0100
bradab	3	”	”	0101
cabrab	3	”	”	0110
dabrab	3	”	”	0111
bracac	1	$\frac{4}{500}$	$6,97$	000000
bracad	1	”	”	000001
bradac	1	”	”	000010
bradad	1	”	”	0000110
cabrac	1	”	”	0001000
cabrad	1	”	”	0001001
dabrac	1	”	”	0001010
dabrad	1	”	”	0001011
cacab	3	”	”	0001100
cadab	3	”	”	0001101
dacab	3	”	”	0001110
dadab	3	”	”	0001111
cacac	1	$\frac{1}{1000}$	$9,97$	0000111000
cacad	1	”	”	0000111001
cadac	1	”	”	0000111010
cadad	1	”	”	0000111011
dacac	1	”	”	0000111100
dacad	1	”	”	0000111101
dadac	1	”	”	0000111110
dadad	1	”	”	0000111111

Dans la colonne de droite, on indique le code de Huffman calculé à partir de ces mots en utilisant leur probabilité. Dans cet exemple, les probabilités sont toutes légèrement supérieures à la puissance de 2 la plus proche, donc la somme des approximations par puissances de 2 est strictement inférieure à 1. Par conséquent, le codage de Huffman pour certains mots est plus court que ce que la puissance de 2 approximante pouvait laisser prévoir. Par exemple, la probabilité de *bracac* est $\frac{4}{500}$, proche de $\frac{1}{2^7}$ et le mot est néanmoins codé par un code de longueur 6.

On peut calculer un transducteur de codage construit à partir de $\mathcal{A}$. Chaque transition de probabilité 1 devient une transition dont la sortie est ε ; à partir de l'état 2, pour chaque ligne du tableau ci-dessus, on a un chemin dont l'entrée est le mot encodé, la sortie est le code de Huffman associé et l'état d'arrivé celui indiqué dans le tableau. L'ensemble étant préfixe, en sortant le code sur la dernière lettre du mot encodé, il n'est pas difficile d'effectuer ce codage avec

un transducteur séquentiel.

La procédure décrite figure 3.15 termine. En effet, si les chemins qu'on considère deviennent grands, l'erreur devient supérieure à $0,5$ et toute probabilité est considérée comme acceptable.

3.4.3 Évaluation de l'efficacité du codage

On suppose qu'on a construit un transducteur $\mathcal{T}$ de codage en utilisant l'algorithme de la figure 3.15.

Supposons dans un premier temps que pour tout mot encodé, le code obtenu n'est pas plus long que la valeur donnée par le logarithme de son approximation en puissance de $\frac{1}{2}$ (comme c'est le cas dans l'exemple).

On peut alors majorer l'efficacité de $\mathcal{T}$ par ce qui serait obtenu si le codage avait exactement la longueur qui correspond à l'approximation.

Soit n un entier (grand) et w un mot de longueur n ; le mot w est encodé par bloc par le transducteur ; on peut écrire $w = u_1u_2 \dots u_kv$ (où v est le préfixe d'un mot encodé par le transducteur). On note p_i l'état atteint après avoir lu $u_1 \dots u_{i-1}$, e_i le vecteur caractéristique de p_i et l_i l'entier tel que $|l_i + \log_b(e_i\mu(u_i)\mathbb{1})|$ est plus petit que $\varepsilon|u_i|$. Notons $f(w) = \sum l_i$. A une constante additive près (codage de v et fin du codage) qui ne dépend pas de w, $f(w)$ majore la longueur de l'encodage de w, donc

$$\mathsf{eff}(\mathcal{T}, \mathcal{A}) \leqslant \lim_{n\to\infty} \frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w) f(w).$$

Par ailleurs, à une constante multiplicative près (comprenant les préfixes des mots encodés par le transducteur), $\mathsf{prob}(w) = I\mu(u_1 \dots u_n)$, donc $|f(w) + \log_b \mathsf{prob}(w)|$ est majoré asymptotiquuement par $\varepsilon|w|$.

Finalement, on obtient :

$$\begin{aligned}\mathsf{eff}(\mathcal{T}, \mathcal{A}) \log(b) - E(\mathcal{A}) &\leqslant \lim_{n\to\infty} \frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w)(f(w) + \log_b \mathsf{prob}(w)) \log(b) \\ &\leqslant \lim_{n\to\infty} \frac{1}{n} \sum_{w\in A^n} \mathsf{prob}(w) n\varepsilon \log(b) = \varepsilon \log(b).\end{aligned}$$

Remarquons que l'on peut toujours se placer dans ce cadre. Comme on l'a vu, il suffit d'imposer dans la procédure que $\lceil x \rceil - x$ soit strictement inférieur à $\varepsilon|w|$ (avec les notations de la figure 3.15. La contrepartie de cette contrainte est qu'on risque d'obtenir des chemins beaucoup plus long, donc un transducteur de codage (et de décodage) plus gros.

Théorème 2 *Pour toute source Markovienne déterministe et tout $\varepsilon > 0$, on peut calculer un transducteur séquentiel qui réalise l'encodage de cette source avec une efficacité égale à ε près à l'entropie de la source.*

Chapitre 4

Noyaux et signatures de langages

Dans ce chapitre nous abordons l'étude des noyaux. Les noyaux sont un moyen pour classifier des mots ou des langages par rapport à un ensemble de témoins connus. Une fonction est appliquée à chaque mot à tester ainsi qu'à chaque témoin ; elle produit un scalaire qui donne une mesure de la proximité entre le mot à tester et le témoin. La notion de proximité dépend évidemment de la fonction appliquée. Une façon de calculer un noyau est de transformer chaque mot en une signature qui est une combinaison linéaire de mots ; le noyau est alors le produit scalaire des signatures des deux mots comparés (voir [22] ou [23]).

Dans [9] sont introduits les noyaux et signatures rationnels qui peuvent être calculés par des transducteurs. Ce chapitre présente l'extension de ces méthodes, puisque nous appliquons les signatures et noyaux non pas à des mots, mais à des langages. Ceci peut être appliqué pour la classification de textes par domaines ou la comparaison entre langues naturelles.

4.1 Noyaux et signatures de langages

On définit formellement les signatures et noyaux de langages dans cette section. En réalité, on s'intéressera aux noyaux rationnels, la plupart des fonctions que l'on définit dans cette section seront donc dans la suite réalisées par automates (ou transducteurs).

Définition 23 *Soit A un alphabet. Un* noyau *sur A^* est une application de $A^* \times A^*$ dans $\mathbb{R}$.*

Cette application est utilisée pour mesurer la proximité des mots.

Un des premiers noyaux utilisés est le noyau de Lohdi ([23]) qui compte le nombre de sous-mots communs entre deux mots.

Nous nous concentrerons dans ce chapitre sur les noyaux rationnels symétriques définis positifs (*cf.* [10]), qui peuvent s'exprimer comme produit scalaire des signatures de deux mots.

Définition 24 *Soit A un alphabet. Une* signature *pour les mots de A^* est une application de A^* dans $\mathbb{R}_+\langle B^*\rangle$, les polynômes sur un alphabet B de variables non commutatives à coefficients réels positifs.*

On pourra avoir $B = A$.

On le verra, en pratique, une signature peut par exemple associer à chaque mot le polynôme de ses facteurs, avec pour chaque facteur un coefficient égal à son nombre d'occurences.

Une telle signature peut être utilisée pour comparer des mots ; deux mots seront proches s'ils ont une signature similaire.

Plus formellement, à partir d'une signature f, on peut définir le *polynôme indicateur* de deux mots comme le produit d'Hadamard des signatures de ces mots :

$$P_f(u, v) = \sum_{w \in B^*} \langle f(u), w\rangle . \langle f(v), w\rangle w.$$

Le *noyau* de deux mots est défini comme le produit scalaire de leurs signatures, c'est-à-dire la somme des coefficient du polynôme indicateur :

$$K(u, v) = \sum_{w \in B^*} \langle f(u), w\rangle . \langle f(v), w\rangle = \sum_{w \in B^*} \langle P_f(u, v), w\rangle .$$

Par exemple, si la signature associée à chaque mot est la liste de ses facteurs (sans tenir compte du nombre d'occurences), le noyau donne le nombre de facteurs en commun.

Le noyau de rang k se concentre sur les termes de la signatures correspondant à des mots de longueur k :

$$K_k(u, v) = \sum_{w \in B^k} \langle f(u), w\rangle . \langle f(v), w\rangle = \sum_{w \in B^k} \langle P_f(u, v), w\rangle .$$

Exemple. Considérons la signature de sous-mots "booléenne" introduite par Lodhi dans [23]. On note $u \sqsubseteq x$ si le mot u est un sous-mot de x et pour tout mot x, on définit $\zeta(x)$ par

$$\forall u \in A^*,\ \langle \zeta(x), u\rangle = \begin{cases} 1 & \text{si } u \sqsubseteq x, \\ 0 & \text{sinon.} \end{cases}$$

Le noyau de Lodhi est alors défini par

$$K(x, y) = \langle \zeta(x), \zeta(y)\rangle = \sum_{u \in A^*} \mathbb{1}_{u \sqsubseteq x} . \mathbb{1}_{u \sqsubseteq y}.$$

L'objet de chapitre est l'extension de ces notions aux langages.

Définition 25 *Soit A un alphabet et μ une variable (qui commute avec les lettres de A). Pour toute signature f,pour tout langage L de A^*, la signature de L de paramètre μ est définie par :*

$$f_\mu(L) = \sum_{w \in L} f(w)\mu^{|w|}.$$

La série indicatrice *de deux langages H est L est définie par :*

$$S_\mu(H, L) = \sum_{w \in B^*} \langle f_\mu(H), w\rangle . \langle f_\mu(L), w\rangle.$$

Le noyau de deux langages $K_\mu(H, L)$ est la somme des coefficients de la série indicatrice.

Comme le nombre de mots d'une longueur fixée est fini, le noyau est bien défini lorsque le paramètre μ est formel, il s'agit d'une série formelle. Pour les applications, il peut être intéressant d'évaluer cette série. Pour cela, il est nécessaire d'étudier son rayon de convergence, qui dépend de la nature du noyau. Nous étudierons les cas particuliers des facteurs et des sous-mots.

S'il existe une valeur de μ pour laquelle le noyau est défini pour toute paire de langage, on peut définir une norme associée à ce noyau :

$$||L||_\mu = \sqrt{K_\mu(L, L)}.$$

4.2 Noyau rationnel

Dans [9], les noyaux rationnels de mots sont définis de la façon suivante :

Définition 26 ([9]) *Soit $\mathbb{K}$ un semi-anneau, soit $\mathcal{T}$ un $\mathbb{K}$-transducteur dans $A^* \times A^*$ et $f : \mathbb{K} \to \mathbb{R}$ une application. Le noyau rationnel K induit par $(\mathcal{T}, f)$ est l'application de $A^* \times A^*$ dans $\mathbb{R}$ définie par $K(u, v) = f \circ \mathcal{T}(u, v)$, pour chaque paire de mots u et v.*

Comme on s'intéresse aux noyaux obtenus par produit scalaire, on définit les signatures rationnelles. Par ailleurs, nous supposons que dans la définition 26, f est un morphisme (de semi-anneaux) et qu'on peut l'appliquer à chaque transition du transducteur de sorte à obtenir un $\mathbb{R}$-transducteur.

Définition 27 *Soit A et B deux alphabets. Soit $\mathcal{T}$ un $\mathbb{R}$-transducteur sur $A^* \times B^*$.*[1] *La $\mathcal{T}$-signature d'un mot w dans A^* est $\mathcal{T}(w)$. La $\mathcal{T}$-signature de paramètre μ d'un langage L inclus dans A^* est :*

$$\sigma_\mu(L) = \sum_{w \in L} \mathcal{T}(w)\mu^{|w|}.$$

1. Vu comme $\mathbb{R}\mathsf{Rat}B^*$-automate sur A^*.

Proposition 8 *Si $\mathcal{T}$ est un $\mathbb{R}$-transducteur, la $\mathcal{T}$-signature de paramètre μ d'un langage est réalisée par un transducteur $\mathcal{T}_\mu$.*

Preuve. Soit $\mathcal{T}_\mu$ le transducteur obtenu à partir de $\mathcal{T}$ en multipliant le poids de chaque transition par μ (si l'étiquette est une paire (a, b) où a est une lettre). L'image d'un mot u par $\mathcal{T}$ est un polynôme ; pour tout calcul de $\mathcal{T}$ étiqueté par u, le poids de la sortie est multiplié par $\mu^{|u|}$. L'image de u par $\mathcal{T}_\mu$ est donc celle par $\mathcal{T}$ multipliée par μ. Par linéarité, on obtient donc la proposition. □

Définition 28 *Le $\mathcal{T}$-noyau de rang k des deux langages L et H est défini par*

$$K_\mu^{(k)}(L, H) = \sum_{w \in B^k} \langle \sigma_\mu(L), w\rangle \langle \sigma_\mu(H), w\rangle.$$

Le $\mathcal{T}$-noyau de paramètre μ de deux langages L et H est

$$K_\mu(L, H) = \sum_{k=0}^{\infty} K_\mu^{(k)}(L, H) = \sum_{w \in B^*} \langle \sigma_\mu(L), w\rangle \langle \sigma_\mu(H), w\rangle.$$

Proposition 9 *Soit $\mathcal{T}$ un $\mathbb{R}$-transducteur sur $A^* \times B^*$, et soit $\mathcal{T}_\mu$ le transducteur qui réalise la $\mathcal{T}$-signature de paramètre μ. Alors, le $\mathcal{T}$-noyau de paramètre μ est réalisée par le transducteur $\mathcal{T}_\mu^{-1} \circ \mathcal{T}_\mu$.*

Preuve. En effet, pour toute paire de mots (u, v) dans $A^* \times A^*$,

$$K_\mu(u, v) = \mathcal{T}_\mu(u).\mathcal{T}_\mu(v) = u[\mathcal{T}_\mu^{-1} \circ \mathcal{T}_\mu]v.$$

Le résultat pour les langages suit par linéarité. □

4.3 Signatures de facteur

4.3.1 Signature de facteurs de mots

Deux mots peuvent être comparés par rapport à leurs contenus communs. À cette fin, nous définissons la *signature* facteur d'un mot.

Soit w un mot de longueur n dans A^*. Chaque paire (i, l) avec i dans $[0; |w| - 1]$ et l dans $[1; |w| - i]$ correspond au mot $v = v_1 \dots v_l$ de longueur l de telle sorte que, pour chaque r dans $[1; l]$, $v_r = w_{r+i}$: c'est le facteur de w d'indice i et de longueur l, on le note $f(w, i, l)$. Soit $\mathsf{fact}(w)$ l'ensemble des couples admissibles (i, l) décrivant un facteur de w.

Définition 29 *Soit w un mot. La signature des facteurs de w est la combinaison linéaire définie par :*

$$\chi(w) = \sum_{(i,l) \in \mathsf{fac}(w)} f(w, i, l).$$

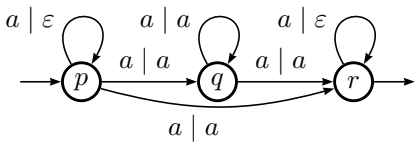

FIGURE 4.1 – Le transducteur calculant la signature des facteurs χ

Pour chaque mot v dans A^+, le coefficient de v dans $\chi(w)$ est le nombre d'occurences de v dans w.

La norme des facteurs d'un mot w est la racine du produit scalaire de sa signature avec elle même, $||w|| = \sqrt{\langle \chi(w), \chi(w) \rangle}$.

EXEMPLE. Soit $w = abacab$. $f(w, 2, 3) = aca$, et $\chi(w) = 3a + 2b + c + 2ab + ac + ba + ca + aba + aca + bac + cab + abac + acab + baca + abaca + bacab + abacab$. Alors, $||w|| = \sqrt{9 + 4 + 1 + 4 + 13} = \sqrt{31}$.

Définition 30 *Soit u et v deux mots dans A^*. Le noyau des facteurs de u et v est $X(u, v) = \langle \chi(u), \chi(v) \rangle$. Le noyau des facteurs normalisé de u et v est $\frac{X(u,v)}{||u||.||v||}$.*

EXEMPLE. Soit $u = tree$ et $v = troe$. $\chi(u) = t + r + 2e + tr + re + ee + tre + ree + tree$ et $\chi(v) = t + r + o + e + tr + ro + oe + roe + troe$. $||u|| = \sqrt{12}$, $||v|| = \sqrt{10}$ et $X(u, v) = 5$. Alors, $\frac{X(u,v)}{||u||.||v||} = \frac{1}{4}\sqrt{\frac{10}{3}}$.

Proposition 10 *Pour tout alphabet A, il existe un $\mathbb{N}$-transducteur à trois états et $6|A|$ transitions qui réalise la signature des facteurs.*

Preuve. Soit $\mathcal{T}$ le transducteur de la figure 4.1. Chaque transition $p \xrightarrow{a|\varepsilon} q$ est une transition qui peut être empruntée par toute lettre et qui efface la lettre en entrée ; chaque transition $p \xrightarrow{a|a} q$ est une transition qui peut être empruntée par toute lettre et qui recopie la lettre en entrée. Le poids de chaque transition de $\mathcal{T}$ est 1, donc le poids de (w, v) dans ce transducteur est égale au nombre de chemins avec l'entrée w et la sortie v ; Ce poids est égal au nombre d'occurrences de v dans w comme facteur.

Pour chaque mot v dans A^+, le coefficient de v dans $\chi(w)$ est le nombre d'occurences de v dans w. □

4.3.2 Signature des facteurs de langages

La signature des facteurs d'un langage L de paramètre μ est définie comme la série

$$\Psi_\mu(L) = \sum_{w \in L} \chi(w) \mu^{|w|}.$$

La signature Ψ_μ est réalisée par le transducteur de la figure 4.2. Dans la suite, ce transducteur est appelée Ψ_μ.

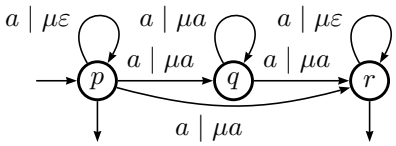

FIGURE 4.2 – Transducteur calculant Ψ_μ qui extrait les facteurs. Chaque transition est validée par une lettre d'entrée : les transition avec etiquette $a \mid a$ copie l'entrée vers la sortie, les transitions avec étiquette $a \mid \varepsilon$ n'ont pas de sorties

EXEMPLE. Considérons deux langages $L = \{abab, ab\}$ et $H = \{bab, bb\}$; alors

$$\psi_\mu(L) = (2\mu^4+\mu^2)a+(2\mu^4+\mu^2)b+(2\mu^4+\mu^2)ab+\mu^4ba+\mu^4aba+\mu^4bab+\mu^4abab.$$

et

$$\psi_\mu(H) = \mu^3a + (2\mu^3 + 2\mu^2)b + \mu^3ab + \mu^3ba + \mu^2bb + \mu^3bab.$$

Définition 31 *Si L est un langage, soit $\mathsf{Fact}_n(L)$ est le nombre de facteurs de longueur n dans L, nous considérons l'entropie de L :*

$$\mathsf{E}(L) = \limsup_{n\to\infty} \frac{\log_2(\mathsf{Fact}_n(L))}{n}.$$

Proposition 11 *Soit L un langage rationnel. Si μ appartient à $[0; \frac{1}{2^{\mathsf{E}(L)}}[$, la signature des facteurs $\Psi_\mu(L)$ est définie.*

Preuve. Considérons β dans $]2^{\mathsf{E}(L)}; 1/\mu[$. Il existe un N tel que pour chaque $n \geqslant N$, $\mathsf{Fact}_n(L) < \beta^n$.

Pour chaque mot v dans A^+,

$$\begin{aligned}
\langle \Psi_\mu(L), v\rangle &= \sum_{w\in L} \langle \chi(w), v\rangle \mu^{|w|} = \sum_{x,y\in A^* \mid xvy\in L} \mu^{|xvy|} \\
&\leqslant \left(\sum_{k\geqslant 0} \mathsf{Fact}_k(L)\mu^k\right) \mu^{|v|} \left(\sum_{k\geqslant 0} \mathsf{Fact}_k(L)\mu^k\right) \\
&< \left(\sum_{k=0}^{N-1} (|A|\mu)^k + \sum_{k\geqslant N} (\beta\mu)^k\right)^2 \mu^{|v|} \\
&< \left(\sum_{k=0}^{N-1} (|A|\mu)^k + \frac{(\beta\mu)^N}{1-\beta\mu}\right)^2 \mu^{|v|}.
\end{aligned}$$

□

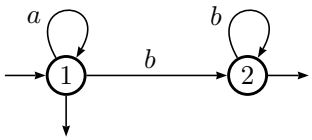

FIGURE 4.3 – L'automate minimal de L_2

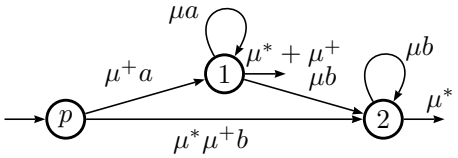

FIGURE 4.4 – Un automate realisant $\Psi_\mu(L_2)$,la signature de facteurs de $L_2 = a^*b^*$.

Notez que $2^{\mathsf{E}(L)}$ est dans $[1; |A|]$, où A est l'alphabet ; Par conséquent, si μ est inférieur à $1/|A|$, $\Psi_\mu(L)$ est définie.

EXEMPLE. Le langage $L_2 = a^*b^*$ est reconnu par l'automate déterministe $\mathcal{A}_2$ de la figure 4.3. L'application de Ψ_μ de $\mathcal{A}_2$, suivie par la suppression des ε-transitions, donne l'automate de la figure 4.4 . L'entropie de L_2 est nulle, donc Ψ_μ peut être évaluée pour tout $\mu < 1$.

Proposition 12 *Si μ est dans $[0, \frac{1}{|A|}[$, où A est l'alphabet, pour chaque paire de langages L et H, le noyau de facteur $K_\mu(L, H)$ est définie.*

Preuve. La norme de chaque langage est plus petite que la norme de A^* et on a

$$\forall w \in A^*,\ \langle \Psi_\mu(A^*), w\rangle = \frac{\mu^{|w|}}{(1-|A|\mu)^2}.$$

Donc,

$$||A^*||_\mu^2 = \sum_{w\in A^*} \langle \Psi_\mu(A^*), w\rangle^2 = \sum_{n=1}^{\infty} \sum_{w\in A^n} \frac{\mu^{2n}}{(1-|A|\mu)^4}$$
$$= \sum_{n=1}^{\infty} \frac{(|A|\mu^2)^n}{(1-|A|\mu)^4} = \frac{|A|\mu^2}{(1-|A|\mu)^4(1-|A|\mu^2)}.$$

□

EXEMPLE. Si $L = a^*b^*$, la série indicatrice des facteurs $I_\mu(L_2, L_2)$ est réalisé par l'automate de la figure 4.5.

Le noyaux des facteurs de deux langages est réalisé par le transducteur $\Psi_\mu \circ (\Psi_\mu)^{-1}$ dessiné figure 4.6

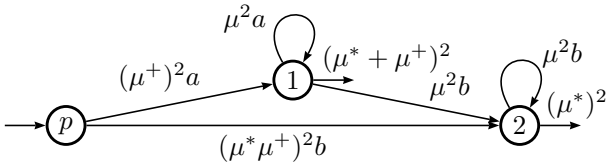

FIGURE 4.5 – La serie indicatrice des facteurs $I_\mu(L_2, L_2)$.

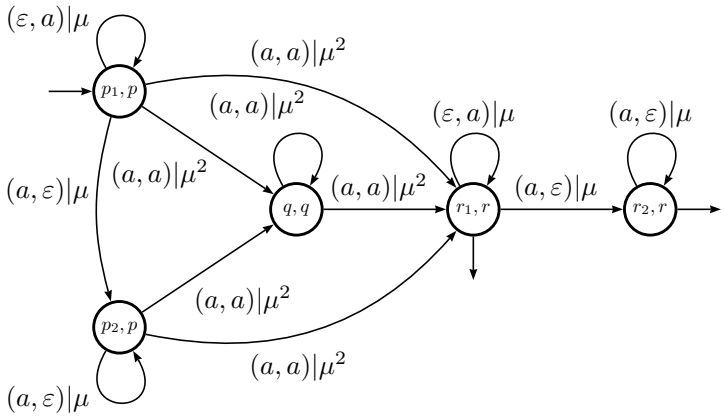

FIGURE 4.6 – Le transducteur qui réalise le noyau des facteurs.

4.4 Signature des sous-mots et noyaux

4.4.1 Signature des sous-mots de mots

Une autre façon de comparer deux mots consiste à considérer leurs sous-mots communs. Comme pour les facteurs, la signature des sous-mots d'un mot peut être définie. De même que la signature des facteurs, la signature des sous-mots de deux mots est paramétrée. En fait, nous ne voulons pas seulement savoir si un mot donné apparaît comme sous-mot, mais également si ses occurrences sont compactes ou dispersées.

Définition 32 *Soit w un mot dans A^*, pour chaque k dans $[1, n]$, toute suite croissante s de longueur k à valeurs dans $[1; |w|]$ définit un mot v de longueur k tel que $v_i = w_{s_i}$: v est le sous-mot de w indexée par s et est notée $\sigma(w, s)$. On note la longueur de s par $|s|$ et nous définissons la largeur de s par $\ell(s) = i_k - i_1 + 1$.*

Soit $S(w)$ l'ensemble des suites strictement croissantes (non vides) dans $[1; |w|]$.

Définition 33 *Soit w un mot. La signature des sous-mots de w de paramètre λ est la combinaison linéaire définie comme suit :*

$$\varphi_\lambda(w) = \sum_{s \in S(w)} \lambda^{\ell(s)-|s|} \sigma(w, s).$$

La signature $\varphi_\lambda(w)$ peut être évaluée pour toute valeur de λ dans $]0,1]$. Si $\lambda = 1$, le coefficient d'un mot v dans $\varphi_\lambda(w)$ est le nombre d'occurrences comme sous-mot de v dans w, et $\varphi_1(\lambda)$ est connue comme la transformation de Magnus (*cf.* [32]) de w, si $\lambda = 0$ chaque séquence avec un trou est éliminée, et $\varphi_\lambda(w) = \chi(w)$.

Le noyau des sous-mots de u et v est

$$K_\lambda(u,v) = \langle \varphi_\lambda(u), \varphi_\lambda(v) \rangle = \sum_{w \in A^*} \langle \varphi_\lambda(u), w \rangle \langle \varphi_\lambda(v), w \rangle.$$

Le noyau d'ordre k de u et v est

$$K_\lambda^k(u,v) = \sum_{w \in A^k} \langle \varphi_\lambda(u), w \rangle \langle \varphi_\lambda(v), w \rangle.$$

La norme des sous-mots de u est $||u||_\lambda = \sqrt{K_\lambda(u,u)}$. Le noyau des sous-mots normalisé de u et v est

$$\frac{K_\lambda(u,v)}{||u||_\lambda . ||v||_\lambda}.$$

Exemple. On considère les mots suivants : *cat*, *car*, *bat*, *bar*. Le tableau suivant montre les coefficients des sous-mots de longueur 2 dans φ_λ pour chaque mot.

	ca	*ct*	*at*	*ba*	*bt*	*cr*	*ar*	*br*
$\varphi_\lambda(cat)$	1	λ	1	0	0	0	0	0
$\varphi_\lambda(car)$	1	0	0	0	0	λ	1	0
$\varphi_\lambda(bat)$	0	0	1	1	λ	0	0	0
$\varphi_\lambda(bar)$	0	0	0	1	0	0	1	λ

Le noyau des sous-mots d'ordre 2 entre *cat* et *car* est $K_\lambda(cat, car) = 1$, et puisque $||cat||_\lambda^{(2)} = ||car||_\lambda^{(2)} = \sqrt{2+\lambda^2}$, le noyau de sous-mots normalisé d'ordre 2 est $\frac{1}{2+\lambda^2}$.

Proposition 13 *Il existe une transduction rationnelle τ_λ pondérée dans $\mathbb{N}[\lambda]$ telle que $\varphi_\lambda = \tau_\lambda \circ \chi$.*

Preuve. Considérons le transducteur à multiplicité de la figure 4.7. Ce transducteur réalise une relation rationnelle τ_λ. Pour chaque mot d'entrée w, il produit l'ensemble des mots qu'on peut obtenir en effaçant des lettres, hormis la première et la dernière, de w ; le coup de chaque effacement est λ. Appelons sous-mot de bord le sous-ensemble des sous-mots qui contiennent la première et la dernière lettre du mot. Finalement, pour chaque mot w, $\tau_\lambda(w)$ est une combinaison linéaire de termes, telle que le coefficient d'un mot u dans cette combinaison est de $k\lambda^{|w|-|u|}$ si u apparait k fois comme sous-mot de bord de w. Chaque sous-mot de w est uniquement décrit comme un sous-mot de bord d'un facteur de w. Par conséquent, $\varphi_\lambda = \tau_\lambda \circ \chi$. □

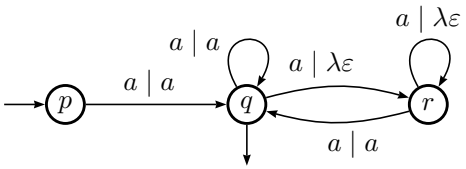

FIGURE 4.7 – Un transducteur effaçant les lettres d'entrées. Chaque transition est étiquetée par une lettre d'entrée : Les transitions d'étiquette $a \mid a$ copient l'entrée vers la sortie, les transitions d'étiquette $a \mid \varepsilon$ efface l'entrée.

Par conséquent φ_λ est la fonction $\mathbb{N}[\lambda]$-rationnelle réalisée par le transducteur de la figure 4.8, simplification de la composition des transducteurs de la figure 4.1 et la figure 4.7.

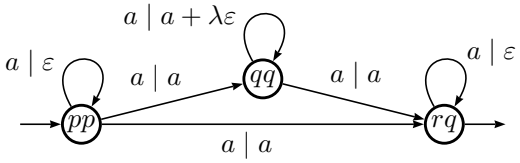

FIGURE 4.8 – Un transducteur calculant la signature de sous-mots d'un mot.

Remarque 5 *Dans [10], les lettres qui participent aux sous-mots sont elles aussi pénalisées d'un facteur λ. La signature ainsi obtenue est donc légèrement différente de celle que nous considérons ici.*

4.4.2 Calculs effectifs de noyaux de sous-mots

On s'intéresse dans cette partie et la suivante au calcul effectif des noyaux de sous-mots. On commence par le cas du noyau booléen puis on présentera le calcul du noyau pondéré.

Noyaux de sous-mots booléen

On présente dans ce paragraphe une construction directe pour le calcul du noyau de sous-mots booléens, c'est-à-dire le nombre de sous-mots communs à deux mots (sans tenir compte de la multiplicité des sous-mots à l'intérieur de chacun des mots),

Pour tout mot w, on définit l'automate $A_S(w)$ qui reconnaît les sous-mots de w. Définissons d'abord la fonction f qui indique la première position supérieure à i dans le mot w où l'on trouve une lettre donnée :

$$\begin{aligned} A^* \times A \times \mathbb{N} &\longrightarrow \mathbb{N} \cup \{\infty\} \\ f(w, a, i) &\longmapsto \min\{j \in]i; |w|] \mid w_j = a\}; \end{aligned}$$

si une telle lettre n'existe pas, on pose $(f, w, a) = \infty$.
Pour tout mot w, l'automate $\mathcal{A}_S(w)$ est alors égal à $([0; |w|], A, E, \{0\}, [0; |w|])$, avec

$$E = \{i \xrightarrow{a} j \in [0; |w|] \times A \times [0; |w|] \mid j = f(w, a, i)\}.$$

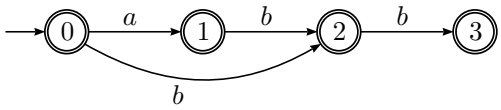

FIGURE 4.9 – L'automate $\mathcal{A}_{S(abb)}$.

Proposition 14 *L'automate $\mathcal{A}_S(w)$ est l'automate déterministe minimal qui reconnaît les sous-mots de w.*

Preuve. Soit u un sous-mot de w et $i_1, i_2, \ldots, i_{|u|}$ la séquence strictement croissante minimale (pour l'ordre lexicographique) telle que pour tout k dans $[1; |u|]$, $u_k = w_{i_k}$. Par minimalité, $i_1 = f(w, u_1, 0)$ et, de même, pour tout k dans $[1; |u|[$, $i_{k+1} = f(w, u_k, i_k)$. Donc $0 \xrightarrow{u_1} i_1 \xrightarrow{u_2} \ldots i_{|u|}$ est un calcul et u est accepté par $\mathcal{A}_S(w)$.

Par ailleurs, si $i \xrightarrow{a} j$ est une transition, par définition $i < j$; donc pour tout calcul $0 \xrightarrow{a_1} i_1 \xrightarrow{a_2} \ldots i_r$, $i_1, i_2, \ldots, i_r$ est une séquence strictement croissante et $a_1 a_2 \ldots a_r$ est un sous-mot tel que $a_k = w_{i_k}$ pour tout k dans $[1; r]$.

Par construction, l'automate est déterministe et acyclique ; comme il accepte w, son nombre d'états est minimal ; c'est donc l'automate minimal. □

EXEMPLE. La figure 4.9 montre l'automate $\mathcal{A}_{S(abb)}$ obtenu par cette construction.

Remarque 6 *L'automate $\mathcal{A}_{S(w)}$ peut être construit en temps linéaire dans la taille du résultat (bornée par $O(|A|.|w|)$). Pour celà, on utilise une liste creuse t (sparse list) indicée par les lettres. On peut décider en temps constant si $t[a]$ est défini et le cas échéant y accéder ; on peut modifier sa valeur et on peut parcourir le support de t en temps proportionnel à sa taille. Dans l'algorithme de la figure 4.10, les entiers de 0 à n représentent les états.*

L'automate reconnaissant le noyau de sous-mots de deux mots est obtenu par produit des automates reconnaissant les signatures de ces mots. Sur ce produit, on peut directement calculer la valeur du noyau (nombre de sous-mots communs) de la manière suivante (bottom-up) :

— on assigne à chaque feuille un poids égal à 1 ;
— on assigne à nœud un poids égal à la somme des poids de ses feuilles plus 1.

Proposition 15 *Le poids de l'état initial du produit des automates $\mathcal{A}_S(u)$ et $\mathcal{A}_S(v)$ est le nombre de sous-mots communs à u et v.*

Preuve. Comme tout état est final, le poids de chaque état est simplement le nombre de mots acceptés à partir de cet état. □

```
Sous-mots de w
    n :=|w|
    A : automate avec n + 1 états terminaux [0; n]
    Rendre 0 initial
    t liste creuse indicée par l'alphabet
    pour k de n à 0
        si k < n
            pour tout a dans support(t)
                créer k -a-> t[a]
        si k > 0
            t[w_k] :=k
    retourner A
```

FIGURE 4.10 – Calcul de l'automate des sous-mots d'un mot

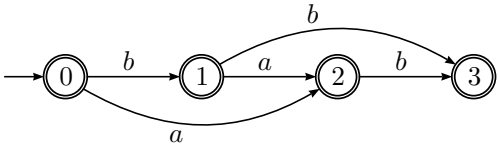

FIGURE 4.11 – L'automate $\mathcal{A}_{S(bab)}$.

EXEMPLE. Considérons l'automate de la figure 4.11 qui reconnaît les sous-mots de *bab*. Son intersection avec l'automate de la figure 4.9, qui reconnaît les sous-mots de *abb*, donne l'automate de la figure 4.12 sur laquelle les poids assignés aux états nous indiquent que le nombre $K(abb, bab)$ de sous-mots communs est 5.

Noyaux avec multiplicité

On propose ici une construction directe d'un automate $\mathcal{A}_{\varphi_\lambda}(w)$ réalisant $\varphi_\lambda(w)$ à partir d'un mot w. Comme pour le cas booléen, cet automate a $|w|+1$ états numérotés de 0 à $|w|$; 0 est initial et tous les autres états sont terminaux. Pour tout i et j dans $[1, |w|]$, avec $i < j$, on a une transition $i \xrightarrow{w_j|\lambda^{j-i-1}} j$. Pour tout j dans $[1, |w|]$, on a une transition $i \xrightarrow{w_j|1} j$.

Proposition 16 *Pour tout mot w, l'automate $\mathcal{A}_{\varphi_\lambda}(w)$ réalise $\varphi_\lambda(w)$.*

Preuve. On a une bijection entre les calculs de $\mathcal{A}_{\varphi_\lambda}(w)$ et $S(w)$: si on excepte l'état initial 0, la suite des numéros des états $i_1, i_2, \ldots, i_k$ d'un calcul forme une suite s de $S(w)$. Avec les notations de la définition 32, il vient immédiatement que l'étiquette de ce calcul est $\sigma(s, w)$. Le poids de ce calcul est

$$\lambda^{1+(i_2-i_1-1)+\ldots(i_k-i_{k-1}-1)} = \lambda^{i_k-i_1+1-(k-1)} = \lambda^{\ell(s)-|s|}.$$

Finalement, d'après la définition 33 l'automate réalise donc $\varphi_\lambda(w)$. □

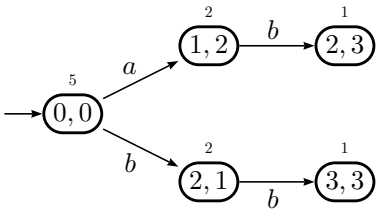

FIGURE 4.12 – L'automate reconnaissant les sous-mots communs de *abb* et *bab*.

EXEMPLE. L'automate de la figure 4.13 réalise $\varphi_\lambda(ctct)$.

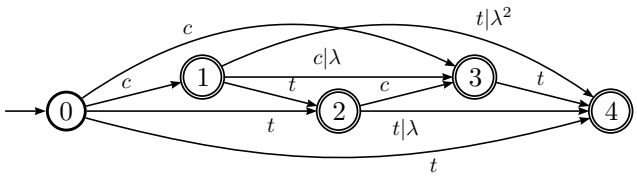

FIGURE 4.13 – L'automate $\mathcal{A}_{\varphi_\lambda}(ctct)$.

Remarque 7 *L'automate $\mathcal{A}_{\varphi_\lambda}(w)$ a $|w|(|w|+1)/2$ transitions et se construit en temps proportionnel à ce nombre de transitions.*

Pour évaluer le noyau de sous-mots de deux mots u et v, on construit $\mathcal{A}_{\varphi_\lambda}(u)$ et $\mathcal{A}_{\varphi_\lambda}(v)$, puis le produit de ces deux automates. On peut ensuite pondérer chaque nœud de la manière suivante, en remontant à partir des feuilles :

— chaque feuille a un poids 1 ;
— Le poids de chaque état est la somme des poids de chacun des successeur, chaque terme étant multiplié par le poids de la transition correspondante, cette somme est augmentée de 1 sauf pour l'état initial.

Comme pour le cas booléen, le poids ainsi calculé correspond aux poids total des calculs partant de chaque état.

EXEMPLE. Le produit de $\mathcal{A}_{\varphi_\lambda}(ctct)$ avec lui-même donne l'automate de la figure 4.14. On assigne des poids comme indiqué et on obtient à l'état initial $16 + 6\lambda^2 + \lambda^4$ qui est donc le noyau de *ctct* avec lui-même.

4.4.3 Signature des sous-mots des langages rationnels

À partir de la signature de sous-mots de mots à un paramètre, on définit la signature de sous-mots de langages. Comme pour la signature de facteurs, l'extension aux langages nécessite l'introduction d'un paramètre supplémentaire.

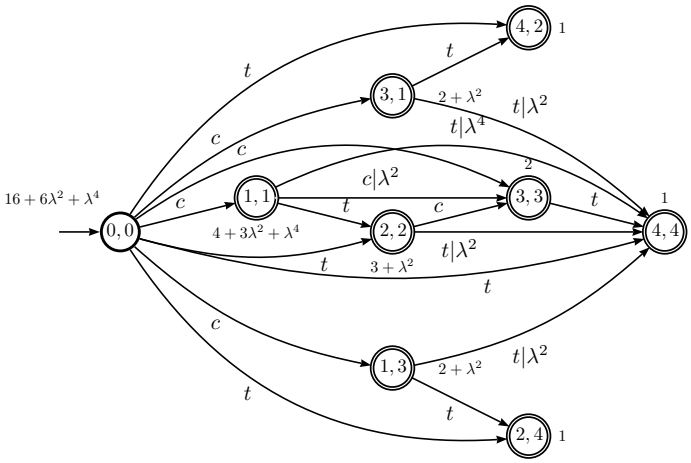

FIGURE 4.14 – L'automate $\mathcal{A}_{\varphi_\lambda}(ctct)^2$.

La signature de sous-mots d'un langage L est donc une série bivariée :

$$\Phi_{\lambda,\mu}(L) = \sum_{w\in L} \varphi_\lambda(w)\mu^{|w|}.$$

Comme pour les facteurs, le paramètre μ pénalise la prise en compte des mots longs.

EXEMPLE. Soit $L = \{abab, ab\}$ et $H = \{bab, bb\}$ deux langages rationnels.

$$\begin{aligned}\Phi_{\lambda,\mu}(L) =&\varphi_\lambda(abab)\mu^4 + \varphi_\lambda(ab)\mu^2\\ =&(2a + 2b + \lambda aa + (2+\lambda^2)ab + ba + \lambda bb + \lambda aab + aba + \lambda abb\\ &+ bab + abab)\mu^4 + (a + b + ab)\mu^2\\ =&(2\mu^4 + \mu^2)a + (2\mu^4 + \mu^2)b + \mu^4\lambda aa + (2\mu^4 + \mu^2 + \mu^4\lambda^2)ab\\ &+ \mu^4 ba + \mu^4\lambda bb + \mu^4\lambda aab + \mu^4 aba + \mu^4\lambda abb + \mu^4 bab + \mu^4 abab;\end{aligned}$$

$$\begin{aligned}\Phi_{\lambda,\mu}(H) =&\varphi_\lambda(bab)\mu^3 + \varphi_\lambda(bb)\mu^2\\ =&(a + 2b + ab + ba + \lambda bb + bab)\mu^3 + (2b + bb)\mu^2\\ =&\mu^3 a + (2\mu^3 + 2\mu^2)b + \mu^3 ab + \mu^3 ba + (\mu^3\lambda + \mu^2)bb + \mu^3 bab.\end{aligned}$$

La série indicatrice de sous-mots de ces deux langages est donc :

$$\begin{aligned}I_{\lambda,\mu}(L,H) =&\Phi_{\lambda,\mu}(L) \odot \Phi_{\lambda,\mu}(H)\\ =&(2\mu^7 + \mu^5)a + (4\mu^7 + 4\mu^6 + 2\mu^5 + 2\mu^4)b + (2\mu^7 + \mu^5 + \mu^7\lambda^2)ab\\ &+ \mu^7 ba + (\mu^7\lambda^2 + \mu^6\lambda)bb + \mu^7 bab.\end{aligned}$$

Le noyau de ces deux langages est donc

$$K_{\lambda,\mu}(L,H) = (10 + \lambda^2)\mu^7 + (4+\lambda)\mu^6 + 4\mu^5 + 2\mu^4.$$

La proposition suivante donne une condition suffisante sur les valeurs de μ et λ pour que la signature d'un langage L soit définie.

Proposition 17 *Soit L un langage rationnel. Pour μ dans $[0; \frac{1}{2^{\mathsf{E}(L)}}[$ et λ dans $[0; \frac{1}{\mu 2^{\mathsf{E}(L)}}[$, $\Phi_{\lambda,\mu}(L)$ est définie.*

Preuve. On prouve que les coefficients de $\Phi_{\lambda,\mu}(L)$ sont tous définis pour ces valeurs.

Soit β dans $]2^{\mathsf{E}(L)}; 1/\mu[$. Il existe N telle que pour chaque $n \geqslant N$, $\mathsf{Fact}_n(L) < \beta^n$.
Soit w un mot de longueur $n \geqslant 2$ (le cas des lettres est facile à résoudre). Soit v un mot de L qui contient une occurence de w comme sous mots : $v = u_0 w_1 u_1 w_2 \dots w_n u_n$, et la participation de cette occurence dans $\langle \Phi_{\lambda,\mu}(L), w\rangle$ est $\mu^{|u_0|}\mu^{|u_n|}\mu^n \prod_{i=1}^{n-1}(\mu\lambda)^{|u_i|}$.

Pour chaque longueur k, pour chaque i dans $[0; n]$, le nombre de mots u_i de L longueur k est plus petit que $|A|^k$ si k est plus petit que N, et est borné par pour β^k pour k plus grand que N. Donc,

$$\langle \Phi_{\lambda,\mu}(L), w\rangle \leqslant \left(\sum_{k=0}^{N-1}(|A|\mu)^k + \frac{(\beta\mu)^N}{1-\beta\mu}\right)^2 \left(\sum_{k=0}^{N-1}(\lambda|A|\mu)^k + \frac{(\lambda\beta\mu)^N}{1-\lambda\beta\mu}\right)^{n-1} \mu^n.$$

Notez que si $L = A^*$, le même argument donne :

$$\langle \Phi_{\lambda,\mu}(A^*), w\rangle = \frac{\mu^n}{(1-|A|\mu)^2(1-\lambda|A|\mu)^{n-1}}.$$

□

Proposition 18 *La fonction $\Phi_{\lambda,\mu}$ qui envoie un langage sur sa signature de sous-mots est une fonction rationnelle.*

Preuve. Pour chaque mot w, le transducteur Ψ_μ de la figure 4.2 donne la série $\chi(w)\mu^{|w|}$. Par la proposition 13, le transducteur τ_λ de la figure 4.7 applique $\chi(w)\mu^{|w|}$ dans $\varphi_\lambda(w)\mu^{|w|}$. par linéarité , $\Phi_{\lambda,\mu}$ est alors égale à $\tau_\lambda \circ \Psi_\mu$. Le transducteur obtenu par la composition de τ_λ et Ψ_μ est dessiné dans la figure 4.15.
□

Exemple. Soit $L_2 = a^*b^*$. L'automate minimal de L_2 est donné figure 4.3. La signature de sous-mots de L_2 est obtenue par l'application de $\Phi_{\lambda,\mu}$ sur cet automate, ce qui donne l'automate de la figure 4.16. Cet automate est en fait équivalent à l'automate de la figure 4.17. On verra plus loin comment calculer cet automate plus simplement.

La signature de sous-mots permet de définir une norme paramétrée pour les langages rationnels. Cette norme est un indicateur de la richesse de sous-mots

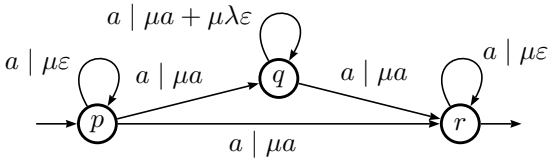

FIGURE 4.15 – Le transducteur $\Phi_{\lambda,\mu} = \tau_\lambda \circ \Psi_\mu$ calculant la signature de sous-mots d'un language.

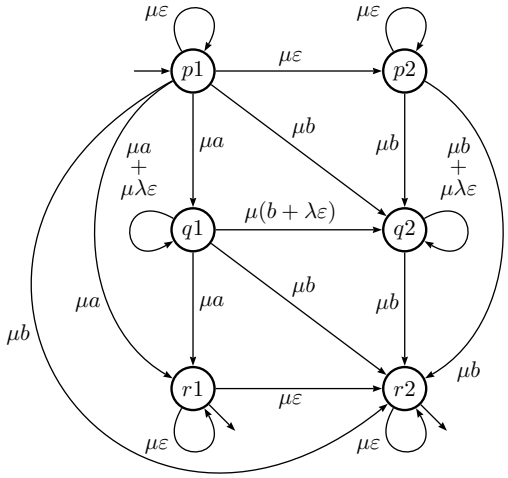

FIGURE 4.16 – La signature de sous-mots de L_2 (Avant suppression des ε-transitions).

dans le langage. La norme des sous-mots de L avec le paramètre (λ, μ) est la suivante :

$$\|L\|_{\mu,\lambda} = \sqrt{\sum_{u \in A^*} \langle \Phi_{\lambda,\mu}(L), u \rangle^2}.$$

Proposition 19 *Soit L un langage rationnel sur un alphabet A. Si λ et μ sont deux réels positifs tels que μ est dans $[0, \frac{1}{|A|}[$ et λ est dans $[0, \frac{1}{\mu|A|} - 1[$, alors $\|L\|_{\mu,\lambda}$ est défini.*

Preuve.

$$\|A^*\|^2_{\lambda,\mu} = \sum_{w \in A^+} \langle \Phi_{\lambda,\mu}(A^*), w \rangle^2 \leqslant \left(\sum_{w \in A^+} \langle \Phi_{\lambda,\mu}(A^*), w \rangle \right)^2,$$

$$\|A^*\|_{\lambda,\mu} \leqslant \sum_{n=1}^{\infty} \frac{1 - \lambda|A|\mu}{(1 - |A|\mu)^2} \left(\frac{|A|\mu}{1 - \lambda|A|\mu} \right)^n.$$

□

À partir du transcteur $\Phi_{\lambda,\mu}$, on peut calculer le transducteur $\Phi_{\lambda,\mu} \circ \Phi^{-1}_{\lambda,\mu}$ (figure 4.18) qui accepte des paires de mots et que l'on peut appliquer à une paire d'automates minimaux de deux langages rationnels.

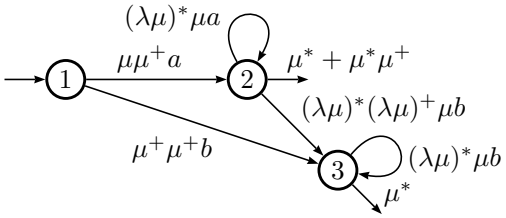

FIGURE 4.17 – Automate reconnaissant les sous-mots de $L_2 = a^*b^*$.

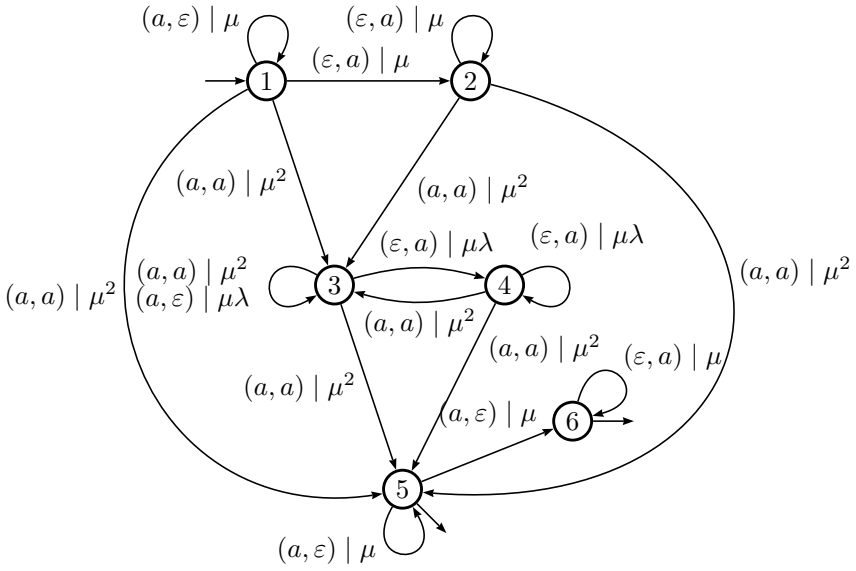

FIGURE 4.18 – Le transducteur réalisant le noyau de sous-mots.

4.4.4 Calcul de la signature des sous-mots

À partir de l'automate minimal $\mathcal{A}$ d'un langage L, un automate qui réalise la signature de sous-mots peut être obtenu en appliquant le transducteur $\Phi_{\lambda,\mu}$.

On peut faire une description du résultat. Il est composé de trois copies $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ de $\mathcal{A}$. Seul l'état initial de $\mathcal{A}_1$ demeure initial et seuls les états terminaux de $\mathcal{A}_3$ demeurent terminaux. Dans $\mathcal{A}_1$ et $\mathcal{A}_3$, chaque transition est remplacée par une ε-transition de poids μ. Dans $\mathcal{A}_2$ chaque transition est pondérée par μ et doublée d'une ε-transition de mêmes extrémités de poids $\mu\lambda$. Pour tout état p de $\mathcal{A}$, notons p_i l'état correspondant dans $\mathcal{A}_i$. Pour toute transition $p \xrightarrow{a} q$, on crée les transitions $p_1 \xrightarrow{a} q_2$, $p_2 \xrightarrow{a} q_3$ et $p_1 \xrightarrow{a} q_3$.

EXEMPLE. L'automate réalisant la signature de $L_2 = a^*b^*$ est dessiné figure 4.16.

Pour pouvoir calculer efficacement avec un tel automate, en particulier pour pouvoir calculer le produit qui permet d'obtenir l'automate qui réalise la série indicatrice de deux langages, il faut supprimer les ε-transitions.

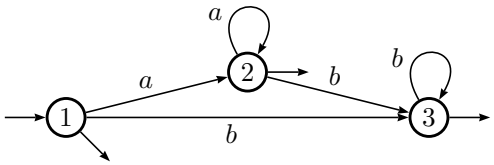

FIGURE 4.19 – Un automate déterministe et standard reconnaissant L_2.

Nous décrivons maintenant une construction qui permet de calculer directement un automate sans ε-transition qui réalise la signature de sous-mots.

Tout d'abord, on suppose que le langage L est reconnu par un automate non ambigu standard $\mathcal{A}$, c'est à dire avec un seul état initial sur lequel n'arrive aucune transition. Un tel automate est facilement calculable à partir de l'automate minimal. Il faut si besoin dupliquer l'état initial, une copie restant initiale, l'autre accueillant les transitions entrantes, et les deux copies conservants leurs transitions sortantes ou flèches finales.

EXEMPLE. Le langage $L_2 = a^*b^*$ est reconnu par l'automate déterministe standard de la figure 4.19.

Soit Q l'ensemble des états de $\mathcal{A}$. Nous définissons dans $Q \times Q$ la matrice $E(x)$, comme suit. Pour chaque paire d'états (p, q),
– $E_{p,q}(x) = kx$, où k est le nombre de transitions de p à q ;
– $E^n_{p,q}(x) = kx^n$ s'il y a exactement k chemins de longueur n de p à q ;
– $E^*_{p,q}(x)$ est la série qui compte le nombre de chemins de chaque longueur entre p et q.

Proposition 20 *Soit $\mathcal{A}$ un automate standard déterministe qui accepte un langage L. Soit Q l'ensemble des états de $\mathcal{A}$, et soit $E(x)$ la matrice paramétrée définie comme ci-dessus. La signature de sous-mots de L de paramètres λ et μ est réalisée par l'automate $\mathcal{B}$ avec le même ensemble d'états Q que $\mathcal{A}$, défini comme suit.*

— *L'état initial i de $\mathcal{B}$ est l'état initial de $\mathcal{A}$.*

— *Pour chaque état $p \neq i$, p est final dans $\mathcal{B}$ avec poids*

$$k = \sum_{q \in T} E^*(\mu)_{p,q} \qquad (si\ k \neq 0).$$

— *Pour chaque état r, pour chaque lettre a dans A, il y a une transition de i à r étiquetée par a et de poids*

$$k = \sum_{q \in Q} E^*(\mu)_{i,q} \mu\delta(a)_{q,r} \qquad (si\ k \neq 0).$$

— *Pour chaque état $p \neq i$, chaque état $r \neq i$, pour chaque lettre $a \in A$, il y a une transition de p à r étiquetée par a de poids :*

$$k = \sum_{q \in Q} E^*(\lambda\mu)_{p,q} \mu\delta(a)_{q,r} \qquad (si\ k \neq 0).$$

Preuve. L'automate $\mathcal{A}$ est caractérisé par le triplet (I, M, T), où I est le vecteur caractéristique des états initiaux et T est le vecteur caractéristique des états terminaux ; M est la matrice de transition de $\mathcal{A}$: si Q est l'ensemble des états de $\mathcal{A}$, M est une matrice $Q \times Q$ telle que pour chaque paire d'états (p, q), $M_{p,q}$ est la somme des lettres qui étiquettent une transition à partir de p et q. Soit i l'état initial de $\mathcal{A}$, puisque $\mathcal{A}$ est standard, $M_{p,i} = 0$ pour chaque état p.

Pour chaque lettre a, la sortie du transducteur $\Phi_{\lambda,\mu}$ sur toutes les transitions est décrite par la matrice suivante :

$$\begin{bmatrix} \mu\varepsilon & \mu a & \mu a \\ 0 & \mu(a + \lambda\varepsilon) & \mu a \\ 0 & 0 & \mu\varepsilon \end{bmatrix}.$$

L'application de $\Phi_{\lambda,\mu}$ dans $\mathcal{A}$ est donc un automate à $3|Q|$ états avec ε-transitions caracterisée par :

$$(J, \Delta, U) = \left(\begin{bmatrix} I & 0 & 0 \end{bmatrix}, \begin{bmatrix} E(\mu) & \mu M & \mu M \\ 0 & \mu M + E(\lambda\mu) & \mu M \\ 0 & 0 & E(\mu) \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \right).$$

Par conséquent, nous avons $\Phi_{\lambda,\mu}(L) = J\Delta^* U$. On peut isoler les ε-transitions, nous obtenons $\Delta = P + F$, où

$$P = \begin{bmatrix} 0 & \mu M & \mu M \\ 0 & \mu M & \mu M \\ 0 & 0 & 0 \end{bmatrix} \text{ and } F = \begin{bmatrix} E(\mu) & 0 & 0 \\ 0 & E(\lambda\mu) & 0 \\ 0 & 0 & E(\mu) \end{bmatrix}.$$

Il vient $\Phi_{\lambda,\mu}(L) = J(P + F)^* U = J(F^* P)^* F^* U$. Par conséquent, $\Phi_{\lambda,\mu}(L)$ est réalisé par l'automate suivant sans ε-transitions :

$$\left(\begin{bmatrix} I & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & \mu E^*(\mu) M & \mu E^*(\mu) M \\ 0 & \mu E^*(\lambda\mu) M & \mu E^*(\lambda\mu) M \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ E^*(\mu) T \end{bmatrix} \right).$$

Cet automate a $3|Q|$ états, mais pour chaque k dans $[1; |Q|]$, les colonnes $k+|Q|$ et $k+2|Q|$ sont égales à la fois dans la matrice de transition et dans le vecteur initial. L'état correspondant peut donc être fusionné pour obtenir un automate à $2|Q|$ états :

$$\left(\begin{bmatrix} I & 0 \end{bmatrix}, \begin{bmatrix} 0 & \mu E^*(\mu) M \\ 0 & \mu E^*(\lambda\mu) M \end{bmatrix}, \begin{bmatrix} 0 \\ E^*(\mu) T \end{bmatrix} \right).$$

Puisque $\mathcal{A}$ est standard (il n'y a pas de transition entrante dans l'état initial), la première colonne de M est nulle, donc, pour chaque k dans $[2; |Q| + 1]$, la colonne k est nulle dans la matrice de transition et dans le vecteur initial,

les états correspondants ne sont pas accessibles, et $\Phi_{\lambda,\mu}(L)$ est réalisé par un automate à $|Q|$ états

$$\left([\ 1 \quad 0\ ], \begin{bmatrix} 0 & (\mu E^*(\mu)M)_{1,2..|Q|} \\ 0 & (\mu E^*(\lambda\mu)M)_{2..|Q|,2..|Q|} \end{bmatrix}, \begin{bmatrix} 0 \\ E^*(\mu)T \end{bmatrix} \right).$$

□

Exemple. Nous considérons l'automate de la figure 4.19. La matrice de transition, la projection et l'etoile de la projection sont :

$$(I, M, T) = \left([\ 1 \quad 0 \quad \cdots \quad 0\ ], \begin{bmatrix} 0 & a & b \\ 0 & a & b \\ 0 & 0 & b \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right),$$

$$E(x) = \begin{bmatrix} 0 & x & x \\ 0 & x & x \\ 0 & 0 & x \end{bmatrix},$$

$$E(x)^* = \begin{bmatrix} 1 & x^+ & x^*x^+ \\ 0 & x^* & x^*x^+ \\ 0 & 0 & x^* \end{bmatrix}.$$

L'automate réalisant $\Phi(L_2)$ est donc caractérisé par :

1)
$$(\mu E^*(\mu)M)_{1,2..3} = [\ \mu\mu^+a \qquad \mu^+\mu^+b\ ].$$

2)
$$(\mu E^*(\lambda\mu)M)_{2..3,2..3} = \begin{bmatrix} (\lambda\mu)^*\mu a & (\lambda\mu)^*(\lambda\mu)^+\mu b \\ 0 & (\lambda\mu)^*\mu b \end{bmatrix}.$$

3)
$$(E^*(\mu)T)_{2..3} = \begin{bmatrix} \mu^* + \mu^*\mu^+ \\ \mu^* \end{bmatrix}.$$

4)

$$(I, M, T) = \left([\ 1 \quad 0 \quad 0\ ], \begin{bmatrix} 0 & \mu\mu^+a & \mu^+\mu^+b \\ 0 & (\lambda\mu)^*\mu a & (\lambda\mu)^*(\lambda\mu)^+\mu b \\ 0 & 0 & (\lambda\mu)^*\mu b \end{bmatrix}, \begin{bmatrix} 0 \\ \mu^* + \mu^*\mu^+ \\ \mu^* \end{bmatrix} \right).$$

L'automate obtenu est celui dessiné sur la figure 4.17.

4.4.5 Les signatures caractérisent les langages

Pour la signature de facteurs comme pour la signatures de sous-mots, la signature d'un langage caractérise le langage et on peut assez aisément retrouver le langage si on dispose de la signature sous sa forme formelle (μ non évalué).

En effet, pour chaque mot w de la signature de L, $\mu^{|w|}$ apparait dans le coefficient de w si et seulement si w est dans L.

À partir d'un automate sans ε-transition qui réalise la signature, il suffit de multiplier chaque transition par μ, puis d'évaluer les coefficients en $\mu = 0$ (et $\lambda = 0$ le cas échéant) pour obtenir un automate qui reconnaît le langage.

Noyau de facteur	football	tennis	ordinateur	carte mère
	$\mu = 0.1$			
football	0.000687522	0.000199493	0.000476418	0.000257754
tennis		0.000890249	0.000573151	0.000241052
ordinateur			0.00954577	0.000971686
carte mère				0.000703869
	$\mu = 0.3$			
football	0.866343	0.495042	0.709353	0.405438
tennis		1.05369		
ordinateur			2.56441	
carte mère				0.818491
	$\mu = 0.5$			
football	60.9736	47.0735	50.5925	32.171
tennis		62.8414	44.031	29.741
ordinateur			94.7652	46.3942
carte mère				42.6872
	$\mu = 0.7$			
football	2717.37	2234.31	2218.26	1764.8
tennis		2459.58		
ordinateur			2753.38	
carte mère				1806.3
	$\mu = 0.9$			
football	141795	128384	131232	166330
tennis		150459	134798	166772
ordinateur			163479	187019
carte mère				268241

FIGURE 4.20 – Comparaison par le noyau des facteur.

4.5 Tests et applications des noyaux de langages rationnels

Dans cette partie on s'intéresse aux tests et applications, pour la classification des documents.

On considère quatre corpus : football, tennis, ordinateur et carte-mère, d'environ 80 mots clés chacun, qu'on compare deux à deux. Pour ces tests on utilise la signature réalisée par le transducteur de la figure 4.22 qui a un paramètre en plus ρ qui nous permet de pénaliser différemment les lettres qui coïncident.

Ainsi, si $\lambda = 0$ et $\rho = \mu$, on a la signature de facteurs de paramètre μ. Si $\lambda = l.\mu$ et $\rho = \mu$, on obtient la signature de sous-mots de paramètre (μ, l) où l est la longueur du sous-mot effacé entre deux lettres qui coïncident.

Remarque 8 *On peut comparer aussi des langages naturels comme le français*

Noyau de facteur	football	tennis	ordinateur	carte mère
	$\mu = 0.1$			
football	1	0, 254993209	0, 185968496	0, 370523873
tennis		1	0, 196611005	0, 304515268
ordinateur			1	0, 374864698
carte mère				1
	$\mu = 0.3$			
football	1	0, 518132318	0, 475908809	0, 481473691
tennis		1		
ordinateur			1	
carte mère				1
	$\mu = 0.5$			
football	1	0, 760470979	0, 665565466	0, 630586701
tennis		1	0, 570572954	0, 574227261
ordinateur			1	0, 72944141
carte mère				1
	$\mu = 0.7$			
football	1	0, 864248361	0, 810970363	0, 796574071
tennis		1		
ordinateur			1	
carte mère				1
	$\mu = 0.9$			
football	1	0, 878964548	0, 861942515	0, 852859614
tennis		1	0, 859495044	0, 830140251
ordinateur			1	0, 893083703
carte mère				1

FIGURE 4.21 – Comparaison par le noyau des facteur normalisé.

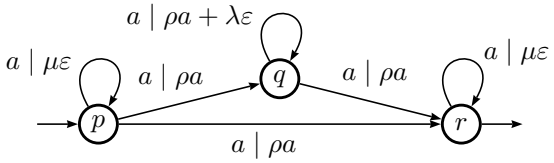

FIGURE 4.22 – Le transducteur $\Phi_{\lambda,\rho,\mu}u$ calculant la signature de sous-mots d'un langage.

par rapport à l'italien et l'anglais par rapport à l'allemand. Malheureusement on n'a pas réussi à obtenir les résultats de comparaison de ces corpus de langues naturelle à cause de leurs tailles importante.

Le tableau de la figure 4.20 présente les résultats obtenus pour le noyau des facteurs avec pénalisation des lettres qui coïncident ($\rho = \mu$). La figure 4.20 présente les mêmes résultats après normalisation.

Dans le graphe de la figure 4.23 on voit les résultats de la comparaison du corpus football avec les autres corpus (football, tennis, ordinateur et carte mère). La courbe en bleu représente le noyaux K(football, football) normalisé. On a bien sûr une courbe constante égale à 1. Ensuite le tennis (en rouge) semble le plus proche du football et les domaines techniques (ordinateur et plus specialement carte mère) ressemblent moins au football.

Remarquons qu'avec un paramètre μ proche de zéro, on a l'impression que

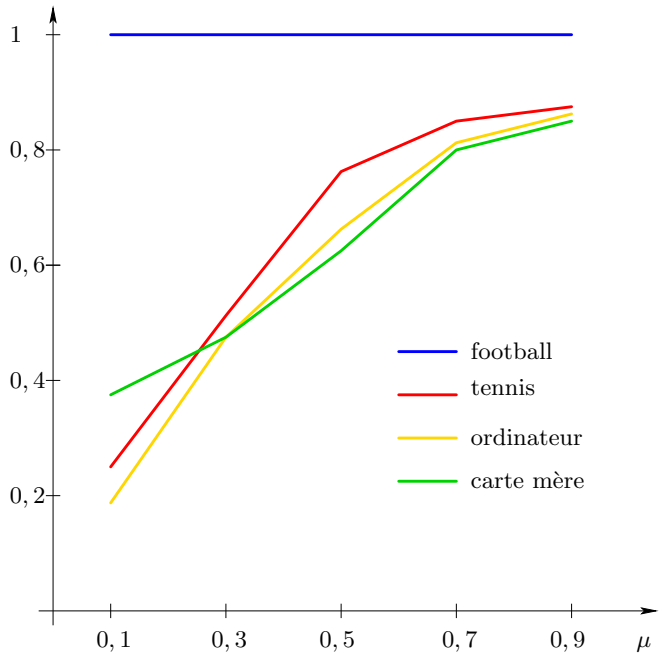

FIGURE 4.23 – Comparaison avec le corpus football

"carte mère" ressemble plus au football que les autres. Cela vient du fait que ce corpus contient des mots plus longs (beaucoup de mots composés), ce qui augmente artificiellement le nombre de facteurs communs si on ne pénalise pas suffisament les mots plus longs.

Globalement, on n'obtient pas de différence très importante car tous ces corpus contiennent des mots de la même langue, le français.

La figure 4.24 présente les résultats obtenus pour le noyau des facteurs avec pénalisation des trous préfixes et suffixes du facteur ($\rho = 1$). La figure 4.25 présente les mêmes résultats après normalisation.

4.6 Extensions et perspectives

4.6.1 Opérations asymétrique et noyaux hybrides

Définition 34 *Un noyau hybride est obtenu par la composition de deux signatures distinctes* $\sigma_1 \circ \sigma_2^{-1}$.

Un noyau hybride n'est donc pas symétrique. Il peut être utilisé si on confronte par exemple des mots obtenus à partir d'un signal audio (considérer les sous-mots permet de prendre en compte le bruit parasite) avec un dictionnaire (où la signature des facteurs est plus pertinente).

Par ailleurs, un noyau hybride peut permettre de comparer des langages sur des alphabets différents.

Noyau de facteur	football	tennis	ordinateur	carte mère
	$\mu = 0.1$			
football	2095.75	108.918	0.391576	0.287699
tennis	0	2096.3	0.0965716	0.106392
ordinateur	0	0	2282.38	188.95
carte mère	0	0	0	2919.61
	$\mu = 0.2$			
football	2229.73	120.49	2.86867	1.68344
tennis	0	2232.3	1.43482	1.27672
ordinateur	0	0	2433.77	207.943
carte mère	0	0	0	3140.83
	$\mu = 0.5$			
football	3887.08	464.824	251.603	153.008
tennis	0	3914.59	205.958	142.77
ordinateur	0	0	4381.65	609.8
carte mère	0	0	0	5599.71
	$\mu = 0.9$			
football	197125	162526	164833	209066
tennis	0	214964	170278	210658
ordinateur	0	0	237204	246564
carte mère	0	0	0	419239

FIGURE 4.24 – Résultat du noyau de facteur

Noyau de facteur	football	tennis	ordinateur	carte mère
	$\mu = 0.1$			
football	1	$0,051964075$	$0,000179041$	$0,000116307$
tennis		1	$0,00004415$	$0,000482167$
ordinateur			1	$0,073196531$
carte mère				1
	$\mu = 0.2$			
football	1	$0,054006818$	$0,001231444$	$0,000636135$
tennis		1	$0,000615576$	$0,000482167$
ordinateur			1	$0,075211166$
carte mère				1
	$\mu = 0.5$			
football	1	$0,119160869$	$0,060965654$	$0,032795906$
tennis		1	$0,049727865$	
ordinateur			1	
carte mère				1
	$\mu = 0.9$			
football	1	$0,78953089$	$0,762276104$	$0,727246632$
tennis	0	1	$0,754075219$	$0,70172064$
ordinateur	0	0	1	$0,781875896$
carte mère	0	0	0	1

FIGURE 4.25 – Noyau de facteur normalisé

4.6.2 Algèbre des noyaux rationnels

Considérons K_1 et K_2 deux noyaux sur $A_1^* \times A_2^*$. On peut définir différentes opérations sur les noyaux (voir [9]). Dans ce qui suit, (x, y) appartient à $A_1^* \times A_2^*$.

Somme :

$$(K_1 + K_2)(x, y) = K_1(x, y) + K_2(x, y).$$

Produit (de Cauchy) :

$$(K_1 \times K_2)(x, y) = \sum_{x=x_1x_2, y=y_1y_2} K_1(x_1, y_1).K_2(x_2, y_2).$$

Proposition 21 *Si les noyaux K_1 et K_2 sont rationnels, alors $K_1 + K_2$ et $K_1 \times K_2$ aussi.*

Preuve. $K_1 + K_2$ est obtenu par union des transducteurs réalisant K_1 et K_2, et $K_1 \times K_2$ par concaténation de ces transducteurs. □

Proposition 22 *Si les noyaux K_1 et K_2 sont obtenus par composition de signatures, alors $K_1 + K_2$ et $K_1 \times K_2$ aussi.*

Preuve. Soit $K_1 = \sigma_{1,1} \circ \sigma_{1,2}^{-1}$ et $K_2 = \sigma_{2,1} \circ \sigma_{2,2}^{-1}$. Alors on a $\sigma_{1,1} : A_1^* \longrightarrow \mathbb{R}\langle B_1^* \rangle$, $\sigma_{1,2} : A_2^* \longrightarrow \mathbb{R}\langle B_1^* \rangle$, $\sigma_{2,1} : A_1^* \longrightarrow \mathbb{R}\langle B_2^* \rangle$ et $\sigma_{2,2} : A_2^* \longrightarrow \mathbb{R}\langle B_2^* \rangle$. On peut supposer que B_1 et B_2 sont deux alphabets disjoints. On peut alors définir, pour i dans $\{1, 2\}$, $\sigma_i : A_i^* \longrightarrow \mathbb{R}\langle (B_1 \cup B_2)^* \rangle$ tel que pour tout mot u de A_i^*, $\sigma_i(u) = \sigma_{i,1}(u) + \sigma_{i,2}(u)$. Comme les alphabets B_1 et B_2 sont disjoints, on obtient :

$$\sigma_1 \circ \sigma_2^{-1} = \sigma_{1,1} \circ \sigma_{1,2}^{-1} + \sigma_{2,1} \circ \sigma_{2,2}^{-1} = K_1 + K_2.$$

De même, soit b une lettre qui n'est pas dans $B_1 \cup B_2$; on définit, pour i dans $\{1, 2\}$, $\pi_i : A_i^* \longrightarrow \mathbb{R}\langle (b \cup B_1 \cup B_2)^* \rangle$ tel que pour tout mot u de A_i^*, $\pi_i(u) = \sum_{xy=u} \sigma_{i,1}(x) b \sigma_{i,2}(v)$. Grâce au marqueur b, on obtient

$$\pi_1 \circ \pi_2^{-1} = K_1 \times K_2.$$

□

Remarque 9 *Si K_1 et K_2 sont rationnels et obtenus par composition de signatures, les opérations décrites dans cette preuve conservent la rationalité.*

Ces opérations s'étendent directement aux noyaux de langages. Par ailleurs, il n'est pas difficile de vérifier que les axiomes de semi-anneaux sont vérifiés en prenant pour noyau unité le noyau caractéristique de $(\varepsilon, \varepsilon)$. Comme on peut facilement obtenir un noyau opposé à un noyau donné, l'ensemble des noyaux forme un anneau. D'après les propositions ci-dessus, c'est aussi vrai des noyaux rationnels, ou des noyaux définis positifs.

4.6.3 Noyaux rationnels n-aires

À partir des signatures, on peut définir des noyaux n-aires.

Définition 35 *Soit $n > 0$. Considérons n signatures : pour tout i dans $[1; n]$, $f_i : A_i^* \longrightarrow \mathbb{R}\langle B\rangle$. Ces signatures induisent un noyau sur $A_1^* \times A_2^* \times \cdots \times A_n^*$:*

$$K(u_1, u_2, \ldots, u_n) = \sum_{w \in B^*} \prod_{i=1}^{n} \langle f_i, w\rangle.$$

On peut de même définir la série indicatrice par

$$K(u_1, u_2, \ldots, u_n) = \sum_{w \in B^*} \left(\prod_{i=1}^{n} \langle f_i, w\rangle \right) w.$$

Table des figures

Index

Bibliographie

[1] Becker, S., Thrun, S., and Obermayer, K., Eds. *Advances in Neural Information Processing Systems 15* (2003), MIT Press.

[2] Berstel, J., and Reutenauer, C. *Rational series and their languages*, vol. 12 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[3] Brookshear, J. G. *Formal languages, automata, and complexity - theory of computation*. Benjamin/Cummings, 1989.

[4] Burges, C. A tutorial on support vector machines for pattern recognition. *Data Mining Knowledge Discovery 2*, 2 (1998), 121–167.

[5] Carroll, J., and Long, D. D. E. *Theory of finite automata with an introduction to formal languages*. Prentice Hall, 1989.

[6] Claveirole, T., Lombardy, S., O'Connor, S., Pouchet, L.-N., and Sakarovitch, J. Inside vaucanson. In *CIAA 2005* (2005), vol. 3845 of *Lecture Notes in Computer Science*, Springer.

[7] Collins, M., and Duffy, N. Convolution kernels for natural language. In *NIPS 2001* (2002), vol. 14, MIT Press, pp. 625–632.

[8] Cormack, G. V., and Horspool, R. N. Data compression using dynamic markov modelling. *Comput. J. 30*, 6 (1987), 541–550.

[9] Cortes, C., Haffner, P., and Mohri, M. Rational kernels. In *NIPS 2002* (2002), vol. 15, MIT Press, pp. 601–608.

[10] Cortes, C., Haffner, P., and Mohri, M. Rational kernels : Theory and algorithms. *Journal of Machine Learning Research 5* (2004), 1035–1062.

[11] Cortes, C., Mohri, M., and Rastogi, A. Distance and equivalence of probabilistic automata. *Int. J. Found. Comput. Sci. 18*, 4 (2007), 761–779.

[12] Cortes, C., Mohri, M., Rastogi, A., and Riley, M. Efficient computation of the relative entropy of probabilistic automata. In *LATIN 2006* (2006), vol. 3887 of *Lecture Notes in Computer Science*, Springer, pp. 323–336.

[13] David, A. H. A method for the construction of minimum redundancy codes. *Institute of Radio Engineers 40*, 9 (1952), 1098–1101.

[14] Droste, M., Kuich, W., and Vogler, H. *Handbook of Weighted Automata*. Springer, 2009.

[15] Eilenberg, S. *Automata, Languages, and Machines, volume A*. Academic Press, 1974.

[16] Herbrich, R. *Learning Kernel Classifiers : Theory and Algorithms*. MIT Press, 2002.

[17] Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel methods in machine learning. *Annals of Statistics 36* (2008), 1171–1220.

[18] Jones, C. B. An efficient coding system for long source sequences. *IEEE Transactions on Information Theory 27*, 3 (1981), 280–291.

[19] Kontorovich, L., Cortes, C., and Mohri, M. Kernel methods for learning languages. *Theoretical Computer Science 405*, 3 (2008), 223–236.

[20] Kuich, W., and Salomaa, A. *Semirings, Automata, Languages*, vol. 5 of *Monographs in Theoretical Computer Science*. Springer, 1986.

[21] Llewellyn, J. A. Data compression for a source with markov characteristics. *The Computer Journal 30*, 2 (1987), 149–156.

[22] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. Text classification using string kernels. *Journal of Machine Learning Research 2* (2002), 419–444.

[23] Lodhi, H., Shawe-Taylor, J., Cristianini, N., and Watkins, C. J. C. H. Text classification using string kernels. In *NIPS 2000* (2001), vol. 13, MIT Press, pp. 563–569.

[24] Lombardy, S., and Sakarovitch, J. Derivatives of rational expressions with multiplicity. *Theoretical Computer Science 332* (2005), 141–177.

[25] Lombardy, S., and Sakarovitch, J. The validity of weighted automata. *International Journal of Algebra and Computation 23* (2013), 863–913.

[26] Lothaire. *Combinatorics on Words*, vol. 17 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, Reading, MA, 1983.

[27] Michael, O. R. Probabilistic automata. *Information and Control 6*, 3 (1963), 230–245.

[28] Mohri, M. Weighted automata algorithms. In *Handbook of Weighted Automata*. Springer, 2009, ch. 6, pp. 213–254.

[29] Pereira, F. C. N., and Riley, M. Speech recognition by composition of weighted finite automata. *CoRR cmp-lg/9603001* (1996).

[30] Sakarovitch, J. *Elements of Automata Theory*. Cambridge University Press, 2009.

[31] Sakarovitch, J. Rational and recognisable power series. In *Handbook of Weighted Automata*. Springer, 2009, ch. 4, pp. 105–174.

[32] Sakarovitch, J., and Simon, I. Subwords. In *Combinatorics on words*, vol. 17 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 83, pp. 104–144.

[33] Scholkopf, B., and Smola, A. J. *Learning with Kernels : Support Vector Machines, Regularization Optimization and Beyond*. MIT Press, 2002.

[34] Schölkopf, B., and Warmuth, M. K., Eds. *16th Annual Conference on Computational Learning Theory, Proceedings* (2003), vol. 2777 of *Lecture Notes in Computer Science*, Springer.

[35] Shannon, C. E. A mathematical theory of communication. *The Bell System Technica 27* (1948), 379–423, 623–656.

[36] Thomas, W. Languages, automata and logics. In *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds., vol. 3. Springer-Verlag, 1997, pp. 389–455.

[37] Zhou, J., Au, O. C., Fan, X., and Wong, P. H.-W. Secure Lempel-Ziv-Welch (LZW) algorithm with random dictionary insertion and permutation. In *ICME 2008* (2008), IEEE, pp. 245–248.

[38] Ziv, J., and Lempel, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory 23*, 3 (1977), 337–343.