Analyse statistique de la structure des automates représentant des dictionnaires électroniques

Lamia Tounsi, Christophe Lenté, Denis Maurel

Université François-Rabelais de Tours, Laboratoire d'Informatique

64, Avenue Jean Portalis – 37200 Tours – France

Abstract

Finite-State Automata are used in many areas of computer science Bioinformatics, Compilers, Natural Language Processing, etc. in order to reduce the memory used by an automata, a search for repetitive structures is performed.

In this paper we present the results of a statistical computation applied to the dictionaries automata.

Résumé

Les automates à nombre fini d'états sont utilisés dans de nombreux domaines tel que le traitement du langage naturel, la bioinformatique, la compilation, etc. Dans l'objectif de réduire la mémoire utilisée par des automates, une recherche de structures répétitives a été menée.

Cette communication présente les résultats statistiques obtenus après cette recherche sur des automates représentant des dictionnaires.

Mots-clés : Automates, Sous automates, Dictionnaires, Traitement automatique du langage naturel.

1. Introduction

L'utilisation de la théorie des automates à nombre fini d'états dans le cadre du traitement automatique des langues, déjà préconisée par Maurice Gross à la fin des années quatre-vingts (Gross et Perrin, 1989), est aujourd'hui un lieu commun (Roche et Schabes, 1997).

La taille d'un automate peut être réduite à celle de l'automate déterministe minimal, et ce, même lors de sa construction (Revuz, 1992 ; Daciuk et al., 2000 ; Watson, 2003), mais celle-ci reste importante. Prenons un exemple simple. En français, un dictionnaire de mots monolexicaux contient dans les huit cent mille mots et on dépasse le million avec les mots polylexicaux.

Cet article se focalise sur les automates représentant des dictionnaires pour lesquels une réduction de la taille apporterait un gain logiciel. Diverses méthodes de représentation ont été proposées (Daciuk, 2000 ; Graña et al., 2001). Une méthode consiste à détecter des structures similaires, en particulier des sous automates, à l'intérieur de l'automate initial. Une première approche a été proposée dans (Tounsi et al., 2005), et approfondie pour mener à une étude statistique des structures présentes dans un automate.

Nous présentons ici les résultats obtenus sur deux automates, celui représentant le dictionnaire des Noms des villes et localités françaises et le dictionnaire DELAF (Courtois et Silberztein, 1990) qui contient presque tous les mots monolexicaux de la langue française.

Il ressort de cette analyse qu'il existe de nombreuses séries similaires à l'intérieur d'un automate. La construction d'un DAWG (Direct Acyclic Word Graph), en utilisant l'algorithme présenté par Mohri (Mohri, 1996), a permis une indexation de ces séries et la mise en évidence de celles qui peuvent être factorisées dans le but de réduire la taille de l'automate initial pour le compresser (Ristov et Laporte, 2000).

Cet article s'articule autour des points suivants :

Tout d'abord, nous allons introduire quelques définitions mathématiques pour appliquer une recherche de sous automates séries-parallèles (Valdez et al., 1982 ; Bachelet, 2003 : 97-130) et non séries-parallèles aux automates des dictionnaires, ensuite un traitement particulier sera consacré aux séries en générant le DAWG associé. Ce traitement permettra de mettre en évidence une redondance dans la structure interne des automates de dictionnaire. Des perspectives sont présentées après la conclusion.

2. Automate et sous automates

Un **automate** déterministe à nombre fini d'états A est un quintuplet.

$A = <\Sigma, Q, \delta, q_i, q_f >$ où Σ est l'alphabet, Q est l'ensemble des états, δ est une fonction de transition : $\delta : Q \times \Sigma \rightarrow Q$, q_i est l'état initial ($q_i \in Q$) et q_f est l'état final ($q_f \in Q$).

Les automates sur lesquels nous travaillons sont de plus minimaux et acycliques.

Soit $p \in Q$, on note Succ(p) l'ensemble des successeurs immédiats de p et Pred(p) l'ensemble des prédécesseurs immédiats de p :

$$Succ(p) = \{ q \in Q : \exists \alpha \in \Sigma : \delta(p, \alpha) = q \}$$
$$Pred(p) = \{ q \in Q : \exists \alpha \in \Sigma : \delta(q, \alpha) = p \}$$

Soit $S \subseteq Q$ et s_i, s_f deux états distincts de S :

Un automate $A = < \Sigma, S, \delta_{/S}, s_i, s_f >$ est un **sous automate** de A si et seulement si :

$$\forall q \in S \setminus \{s_i, s_f\} : Succ(q) \subset S, \quad Pred(q) \subset S$$

3. Recherche de sous automates Séries-Parallèles

Il est connu que les automates représentant des dictionnaires contiennent de nombreuses transitions en série ou en parallèle. Une recherche et une étude statistique de ces transitions sont proposées dans ce qui suit.

3.1. Détection de sous automates séries-parallèles

La méthode s'exécute en plusieurs passes. L'idée conductrice est de remplacer itérativement les transitions parallèles et les transitions séries par une unique transition qui référence un sous automate. La recherche considère d'abord les transitions parallèles, puis les transitions séries, etc. Dans la figure 1, l'automate initial contient plusieurs transitions en parallèle, par exemple les états 2 et 7 sont reliés par trois transitions. La première passe nous permet d'aplatir directement toutes ces transitions pour obtenir un automate dénué de parallèle. La seconde passe concatène toutes les séries en une unique transition et donne l'automate A3, en aplatissant de nouveau les parallèles on obtient l'automate A4 et en aplatissant ensuite les séries on obtient l'automate A5. L'algorithme se termine donc sur l'automate A5 qui ne présente plus aucune série ni aucun parallèle. Cet automate reconnaît le même dictionnaire que l'automate A1, mais ses transitions se réfèrent soit à des lettres, soit à des sous-automates de A1.

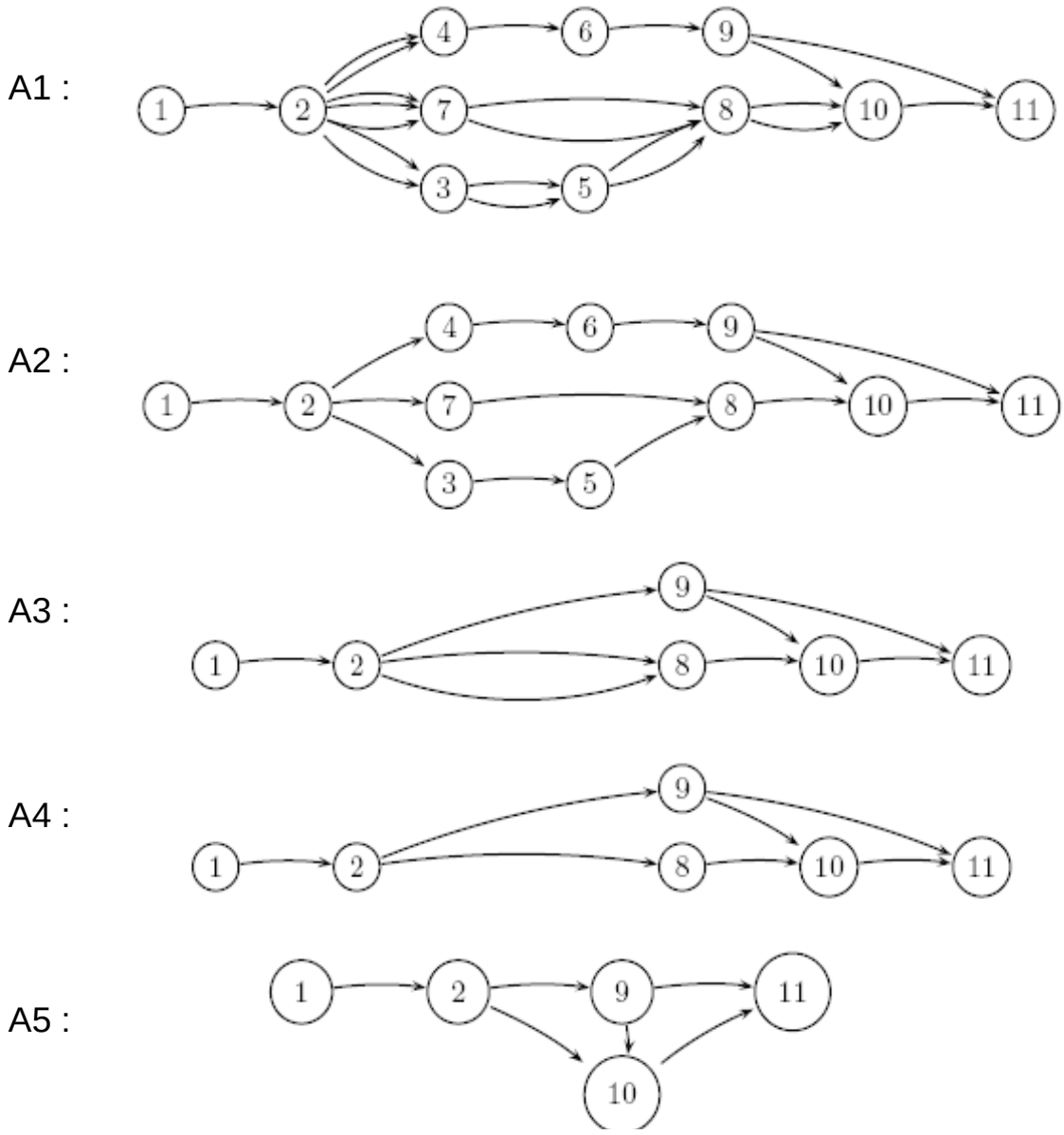

Figure 1 : Recherche de sous automates séries-parallèles

3.2. Application aux dictionnaires

La méthode décrite ci-dessus a été appliquée et expérimentée pour rechercher des sous automates séries-parallèles au sein de deux dictionnaires de langue française. Le tableau 1 résume les grandes caractéristiques de ces dictionnaires. Il ressort que la plupart des répétitions apparaissent lors de la première passe.

Dictionnaires	Nombre de mots	Nombre d'états	Nombre de transitions
DELAF	800 000	67 995	177 465
Noms des villes	38 472	61 240	95 589

Tableau 1 : Dictionnaires

3.2.1. Les parallèles de niveau 1

Le tableau 2 indique le nombre de sous automates parallèles reconnus ; en examinant les résultats, on constate que les parallèles sont de largeur réduite, 4 au maximum pour le DELAF et 5 pour le dictionnaire des Noms de villes, il en ressort aussi que le nombre de parallèles composées de deux transitions est important, il représente en moyenne 97% du nombre total des parallèles pour les deux dictionnaires. Seulement 9% des parallèles du DELAF et 35% des parallèles des Noms de villes sont totalement distincts, ce qui représente un nombre de répétitions considérable.

	Nombre de transitions	Nombre de parallèles distincts	Nombre total de parallèles
DELAF	2	158	1 678
	3	18	24
	4	4	4
Noms des villes	2	129	416
	3	24	29
	4	1	1
	5	2	2

Tableau 2 : Largeur des sous automates parallèles

Le tableau 3 représente la répartition des transitions en parallèles au niveau 1 (des plus fréquentes au moins fréquentes). Ainsi, le dictionnaire du DELAF possède 44 parallèles dont le nombre de répétitions varie de 5 à 760, ce qui représente réellement 1 314 répétitions. Pour le dictionnaire des Noms de Villes on obtient 19 parallèles avec des répétitions variant de 5 à 33, c'est-à-dire au total 141 répétitions. Une factorisation de toutes ces structures redondantes apporterait certainement un gain. Ce sera l'objet de nos travaux futurs.

DELAF	Nombre de répétitions	760	138	50-100	20-49	10-19	5-9	2-4	1
	Nombre de parallèles	1	1	2	6	10	24	54	60
		44 (1 314 répétitions)							
Noms des villes	Nombre de répétitions	33	16	11-13	9	7	5-6	2-4	1
	Nombre de parallèles	1	3	3	3	2	7	25	45
		19 (141 répétitions)							

Tableau 3 : Répartition des sous automates parallèles

3.2.2. Les séries de niveau 1

Le tableau 4 indique le nombre de sous automates séries reconnus au niveau 1. Cette recherche s'effectue à la suite du calcul et de la mise à plat des parallèles, ce qui implique que parmi les séries détectées certaines contiennent un ou plusieurs parallèles.

Dictionnaire	Nombre de séries	Nombre de séries sans parallèle
DELAF	18 093	17 965
Noms des villes	13 981	13 883

Tableau 4 : Sous automates séries

Le tableau 5 représente le nombre de séries correspondant à chaque longueur trouvée dans l'automate représentant le DELAF et le tableau 6 donne les valeurs correspondantes pour le dictionnaire des noms des villes. Les différences entre ces deux tableaux ne sont pas surprenantes car un grand nombre de noms de ville sont construits sur des patrons réguliers (Saint, sur-Loire, etc).

Longueur des séries	2	3	4	5	6	7	8	9	10-18
Nombre de séries	10 502	4 223	1991	835	319	133	55	17	18
			3 350						

Tableau 5 : Nombre de séries du DELAF en fonction de leurs longueurs

Longueur des séries	2	3	4	5	6	7	8	9	10-14	15-18	19-33
Nombre de séries	5 151	2 666	1 882	1 240	922	601	428	300	592	78	67
			5 373								

Tableau 6 : Nombre de séries du dictionnaire des Noms de villes en fonction de leurs longueurs

Le tableau 7 représente les répétitions de sous séries distinctes. Ainsi, 42% des séries du DELAF et 17% des séries du dictionnaire des Noms de Villes se répètent plus d'une fois. Le nombre de transitions impliquées dans ces séries est conséquent, leur factorisation est donc envisageable.

DELAF	Nombre de répétitions	4397-1 035	936-500	498-200	198-100	99-50	49-10	9-5	4-2	1
	Nombre de sous séries distinctes	19	7	45	68	104	389	542	1 170	3 087
Noms des villes	Nombre de répétitions	5 588-1018	999-506	482-200	199-100	99-50	49-10	9-5	4-2	1
	Nombre de sous séries distinctes	14	30	66	107	148	1 033	1 064	3 453	29 490

Tableau 7 : Répétitions de sous-séries distinctes

Ces séries ont été indexées afin de poursuivre la recherche de redondance au sein des automates étudiés.

4. Traitement des séries

L'étude des textes et des répétitions que l'on peut y trouver commence par la création d'un index de tous les mots, ou facteurs, apparaissant dans ces textes. Pour cela, les structures de données les mieux adaptées et les plus simples, tant au point de vue de espace mémoire qu'elles occupent, qu'au point de vue de rapidité de traitement qu'elles procurent, sont les arbres ou les automates de suffixes. En effet, elles ont l'avantage primordial d'avoir une taille linéaire par rapport au texte de base. De plus, le temps d'accès à un facteur du texte se fait en temps linéaire par rapport à la taille de ce facteur.

À côté du développement des arbres de suffixes, qui sont une compression de l'arbre de base, les automates de suffixes ont fait leur apparition en tant que minimisation de cet arbre de base. Appelés DAWG d'après leur nom anglais (Directed Acyclic Word Graphs), ils ont été décrits pour la première fois par Blumer et al. (1983), Blumer et al. (1985). Nous avons

adapté le système d'indexation utilisé par Mohri sur un DAWG et nous l'avons utilisé pour comparer les séries et sous séries trouvées plus haut.

Originellement, cet algorithme permet de construire un automate représentant l'index général d'un texte ; pour l'utiliser, les séries présentes dans un automate ont été concaténées pour former un texte en insérant un séparateur (#) entre chaque série.

L'automate obtenu reconnaît l'ensemble des sous séries créées à partir de ce nouveau texte qui modélise toutes les séries récupérées de l'automate initial.

Afin d'obtenir l'ensemble des sous séries réellement présentes, ainsi que leurs fréquences d'apparition, il est nécessaire d'émonder ce nouvel automate en supprimant l'ensemble des transitions portant l'étiquette # et les états non accessibles après ces suppressions.

L'automate représentant les deux mots *elle* et *les* possède 19 transitions et 13 états, avant émondage (figure 2) et 10 transitions et 8 états, après émondage (figure 3). Les informations portées par chaque état sont présentées dans le tableau 8.

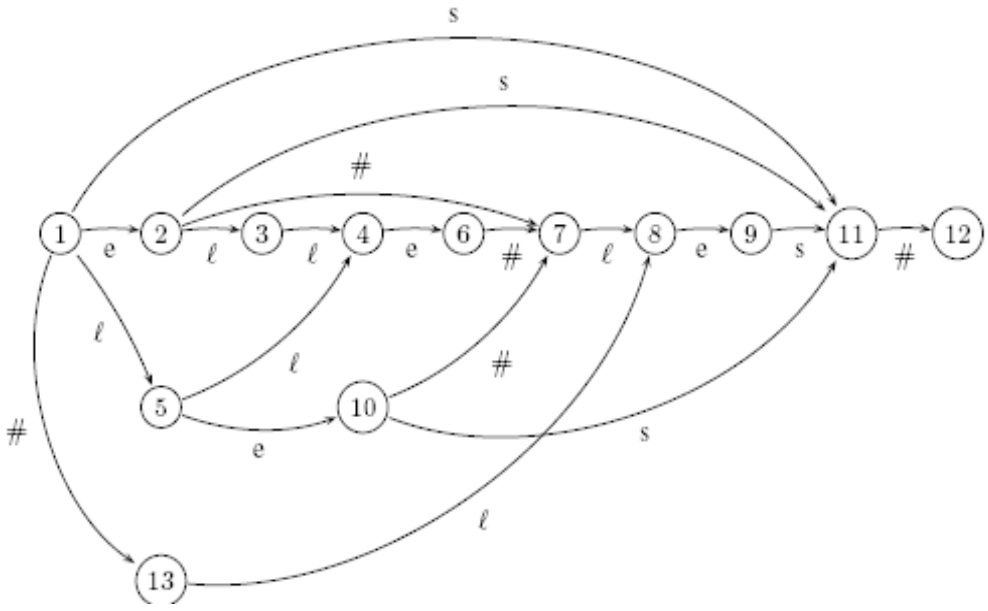

Figure 2 : Représentation des deux mots ″elle″ et ″les″ avant émondage

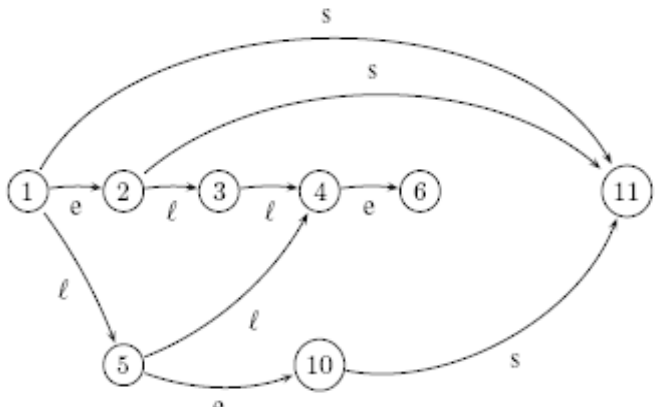

Figure 3 : Représentation des deux mots ″elle″ et ″les″ après émondage

Etat	1	2	3	4	5	6	10	11
Nombre de séries distinctes	0	1	1	2	1	2	1	3
Longueur	0	1	2	3	1	4	2	3
Nombre de répétitions	0	3	1	1	3	1	2	1

Tableau 8 : Informations portées par les états

La cinquième colonne du tableau nous apprend qu'on arrive à l'état 4 par deux sous séries (*ell* et ll) dont la longueur maximale est 3, chacune de ces sous séries est unique. L'avant-dernière colonne du tableau nous apprend qu'on arrive à l'état 10 par une unique sous série de longueur 2 ″le″, cette sous série apparaît 2 fois, une fois dans ″elle″ et une fois dans ″les″.

Après avoir décelé cet ensemble de sous séries, l'idée est de repérer celles dont la factorisation apportera le plus gros gain : la solution optimale proviendra un compromis entre la longueur des séries et le nombre de répétitions.

5. Conclusion et perspectives

L'analyse des deux dictionnaires DELAF et Noms des villes a révélé la présence de nombreuses structures séries et parallèles au sein des automates représentant ces dictionnaires.

On a pu constater que le DELAF contient 28% de transitions en série et 2% en parallèle tandis que le dictionnaire des villes contient 58% de transitions en série et 1% en parallèle. De surcroît, en moyenne, il y a 42% de sous séries qui se répètent plus d'une fois pour le DELAF et 16% pour le dictionnaire des noms de villes.

Une factorisation de ces séries permettrait de réduire la taille de ces automates. Pour cela, le DAWG offre des perspectives intéressantes, car chacun de ses états indexe un ensemble de sous séries de l'automate initial. L'étape suivante consistera à sélectionner un ensemble d'états du DAWG dont la factorisation des séries associées réduira au maximum la taille de l'automate de dictionnaire. Le choix de l'ensemble optimal devra s'effectuer en fonction de la taille des séries et de leur fréquence d'apparition.

Remerciements : Ce travail de recherche est parrainé par la Région Centre. Les auteurs remercient Béatrice Bouchou pour ses conseils sur le sujet ainsi que le Professeur Eric Laporte pour l'utilisation du dictionnaire DELAF.

Références

Bachelet B. (2003). *Modélisation et optimisation de problèmes de synchronisation dans les documents hypermédia,* thèse Université Blaise Pascal, Clermont-Ferrand.

Blumer A., Blumer J., Ehrenfeucht A., Haussler D., McConnel R. (1983). Linear size finite automata for the set of all subwords of a word : an outline of results. Bull. European Assoc. *Theoret.Comput. Sci*., 21 : 12-20.

Blumer A., Blumer J., Haussler D., Ehrenfeucht A., Chen M.T., Seiferas J. (1985). The smallest automaton recognizing the subwords of a text. *Theoret.Comput. Sci*., 40 : 31-55.

Courtois B., Silberztein M. (1990). Dictionnaires électroniques du français, *Langue française*, 87 : 11-22.

Daciuk J. (2000). Experiments with Automata Compression, CIAA 2000, in *LNCS* 2088 : 105-112.

Daciuk J., Mihov S., Watson B. W., Watson R. E. (2000). Incremental construction of Minimal Acyclic Finite State Automata. *Computational Linguistics*, 26-1 : 3-16.

Graña J., Barcala F-M., Alonso M. A. (2001). Compilation Methods of Minimal Acyclic Finite-State Automata for Large Dictionaries. CIAA 2001, in *LNCS* 2494 : 135-148.

Gross M., Perrin D. (ed.) (1989). Electronic Dictionaries and Automata in Computational Linguistics, *LNCS*, 377.

Mohri M. (1996). On some applications of finite-state automata theory to natural language processing, *Natural Language Engineering*, 2 : 1-20.

Revuz D. (1992). Minimization of acyclic deterministic automata in linear time, *Theoretical Computer Science*, 92 : 181-189.

Ristov S., Laporte E. (2000). Ziv Lempel compression of huge natural language data tries using suffix arrays, *Journal of Discrete Algorithms*, 1-1 : 241-256.

Roche E., Schabes Y. ed. (1997). *Finite state language Processing*. Cambridge, Mass./London, England, MIT Press.

Tounsi L., Maurel D., Bouchou B. (2005). Basic Search of Sub Automata. *Application to Electronic Dictionaries*, IEEE NLP-KE 2005 : 543-548.

Valdez J., Tarjan R. E., Lawler E. L. (1982). The recognition of series-parallel digraphs. *SIAM J. Comput*. 11-2 : 298-313.

Watson B.W. (2003). A new algorithm for the construction of minimal acyclic DFAs, *Science of Computer Programming*, 48-2-3 : 81-97.