# Extraction automatique de motifs syntaxiques

Jean-Gabriel Ganascia

LIP6 - Université Pierre et Marie Curie
8, rue du Capitaine Scott, 75015 Paris France
Jean-Gabriel.Ganascia@lip6.fr

## Résumé – Abstract

Cet article présente un nouvel algorithme de détection de motifs syntaxiques récurrents dans les textes écrits en langage naturel. Il décrit d'abord l'algorithme d'extraction fondé sur un modèle d'édition généralisé à des arbres stratifiés ordonnés (ASO). Il décrit ensuite les expérimentations qui valident l'approche préconisée sur des textes de la littérature française classique des XVIII$^{e}$ et XIX$^{e}$ siècle. Une sous-partie est consacrée à l'évaluation empirique de la complexité algorithmique. La dernière sous-partie donnera quelques exemples de motifs récurrents typiques d'un auteur du XVIII$^{e}$ siècle, Madame de Lafayette.

**Mots clefs** : Extraction de motifs, arbres stratifiés ordonnés, distances d'édition, séquences.

This paper presents a new algorithm designed to detect recurrent syntactical patterns in natural language texts. It first describes the pattern extraction algorithm which is based on an edit model generalized to Stratified Ordered Trees (SOT). Then it focuses on experiments with french classical literature of the 18$^{th}$ and 19$^{th}$ century. One section is dedicated to the efficiency evaluation. The last provides some examples of such recurrent patterns that are typical of an 18$^{th}$ century author, Madame de Lafayette.

**Keywords** : Pattern extraction, stratified ordered trees, edit distance, sequences.

## 1 Extraction automatique de motifs syntaxiques

Nous présentons ici un algorithme d'extraction automatique de motifs textuels répétitifs. Notre but est de détecter des motifs syntaxiques récurrents. Trois motivations guident cette recherche.

La première est de caractériser la personnalité des écrivains, car nous pensons que, ce que l'on appelle couramment, et un peu pompeusement *le style* est partiellement déterminé par les structures syntaxiques employées et partiellement par le lexique. Dans le passé, les approches de la « stylométrie » se sont essentiellement fondées sur une analyse statistique des occurrences de mots. Nous souhaiterions développer une approche syntaxique. Dans le futur proche, nous testerons cette hypothèse de travail sur les imitations : il existe toute une littérature de parodie où des auteurs caricaturent de grands écrivains en répétant les traits saillants de leurs styles personnels. Si notre hypothèse se vérifie, quelques unes des manières des écrivains se retrouveront dans les motifs syntaxiques employés par leurs imitateurs.

La seconde motivation est éducative. Notre but serait d'aider les écoliers, les étudiants ou les jeunes écrivains à évaluer la richesse et la diversité de leur style propre. Il serait ainsi possible d'identifier les fautes classiques et de proposer des corrections. Qui plus est, les personnes

apprenant des langues étrangères pourraient profiter d'une telle analyse pour détecter leurs expressions idiomatiques. Plus précisément, il arrive que certaines constructions soient correctes d'un point de vue grammatical, mais inhabituelles, ce qui rend leur usage fréquent étrange et gênant pour un lecteur dont c'est la langue maternelle.

La troisième motivation est universitaire. La linguistique informatique pourrait utiliser un tel outil pour distinguer différents registres de langue et les caractériser. Les progrès dans l'analyse du langage naturel couplés à ceux des techniques d'intelligence artificielle et de fouille des données, aideront à détecter de tels motifs syntaxiques, fournissant ainsi une nouvelle façon d'étudier les usages du langage. Ainsi, il serait possible d'engendrer des figures d'expression et d'automatiser les procédures de la rhétorique classique.

Dans l'architecture préconisée pour découvrir de tels motifs, un apprentissage non supervisé regroupe ensemble des motifs similaires. La première partie de l'article décrit l'algorithme d'apprentissage. Après une présentation de l'ensemble de la chaîne de traitement, l'article porte sur la distance d'édition, puis sur l'algorithme de génération qui construit le graphe de similarité. Ensuite, l'article introduit l'algorithme « centre étoile » conçu pour extraire des sous-graphe fortement connectés du graphe de similarité, et qui correspondent à des classes de motifs similaires. Enfin, la dernière partie de l'article décrit quelques expérimentations. Nous y évaluerons l'approche proposée à la fois d'un point de vue algorithmique et quantitatif, et d'un point de vue linguistique et qualitatif.

# 2 Chaîne de traitement

Pour résumer, nous avons dessiné (voir figure 1) l'ensemble de la chaîne de traitement qui prend, en entrée, un texte en langage naturel et qui engendre, en sortie, un ensemble de motifs récurrents.

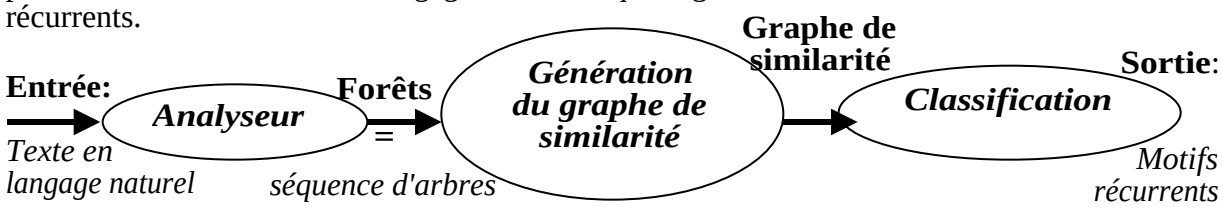


Figure 1 : chaîne de traitement

Dans la première sous-partie, les entrées du processus ainsi que l'étape d'analyse sont présentées, tandis que, dans la seconde, ce sont les ASO (Arbres Structurés Ordonnés) qui sont détaillés, car l'algorithme d'extraction repose en grande partie sur l'utilisation de ces structures de données.

## 2.1 Phase d'analyse du texte en langage naturel

L'entrée principale du système consiste en un texte écrit en langage naturel qui est réductible à une séquence de phrases chacune étant affirmative, interrogative, exclamative ou négative. Un traitement automatique de ce texte réalisé par un analyseur syntaxique ou un catégoriseur, associe des étiquettes à chaque mot (nom, verbe, etc.), ou à chaque groupe de mots (groupe nominal, syntagme verbal, etc.). Cet article ne portant pas, à proprement parler, sur les analyseurs ou les catégoriseurs, nous ne détaillerons pas ces processus. Nous insistons seulement sur la généralité de l'approche qui prend comme entrée un arbre d'analyse engendré par différentes grammaires avec différents ensembles d'étiquettes. La plupart du temps, les arbres sont stratifiés, autrement dit, les étiquettes forment une partition dont les classes dépendent de la profondeur du nœud dans l'arbre d'analyse. Par exemple, un niveau correspond à des syntagmes non-récursifs, un second à des catégories de mots, un troisième à des attributs comme le genre ou le nombre, un dernier à des lemmes, c'est-à-dire à des formes canoniques de noms ou de verbes. Soulignons toutefois que notre approche n'est pas restreinte

à une décomposition syntaxique en groupe non récursifs. La seule limitation porte sur le résultat de la phase initiale d'analyse qui doit s'exprimer sous forme d'un ASO (Arbre Structuré Ordonné) (voir sous-partie suivante). Les expérimentations conduites avec cette architecture ont fait appel à deux traitements différents. Le premier utilise le catégoriseur Winbrill-0.3, dont la base de connaissance a été enrichie par l'INALF-CNRS (Lecomte 1998). Cet outil étiquette simplement les mots, sans extraire de groupes. Le second fait appel à l'analyseur Vergne-98 écrit par Jacques Vergnes (voir http://www.info.unicaen.fr/~jvergnes et (Vergnes 1999)) au GREYC (Groupe de Recherche en Informatique, Image, Instrumentation de Caen). Pour des raisons de place, nous n'évoquerons ici que la seconde expérimentation avec l'analyseur de Jacques Vergnes.

## 2.2 Arbre Stratifiés Ordonnés (ASO)

Selon une définition classique, un arbre ordonné est un arbre dans lequel les relations de gauche à droite, entre fils d'un même nœud, sont porteuses d'information. En d'autres termes, cela signifie qu'un arbre ordonné peut être récursivement défini comme un nœud étiqueté suivi d'une forêt, c'est-à-dire d'une séquence d'arbres.

Toutes les données séquentielles peuvent évidemment se représenter à l'aide d'arbres ordonnés de profondeur 1. En ajoutant des niveaux aux arbres ordonnés, il est possible d'organiser les données de façon à représenter une connaissance implicite. Par exemple, un texte, c'est-à-dire une séquence de caractères, est un liste de phrases, chacune d'entre elles étant composée de mots et de symboles de ponctuation. Ceci peut être représenté à l'aide d'un arbre de profondeur 3 qui explicite cette structure. Considérons maintenant les arbres résultant de l'analyse syntaxique, il faut introduire au moins deux niveaux supplémentaires correspondant aux groupes syntaxiques et aux catégories de mots.

Comme nous l'avons précédemment montré, les arbres ordonnés accroissent la puissance de représentation en sorte qu'il est possible de détecter des sous-arbres semblables, eu égard à cette structuration des données. Il est aussi possible d'extraire des motifs généraux qui ont de multiples occurrences approximatives. Par exemple, n'importe quelle séquence de groupes syntaxiques qui apparaît plusieurs fois dans les données peut être détectée, sans considérer les mots couverts par ces structures syntaxiques, ni leur catégorie.

Toutefois, en pratique, la détection est rendue impossible du fait du très grand nombre de paires potentielles de sous-arbres. Afin de réduire la complexité, les nœuds sont regroupés en classes de sorte que deux nœuds ne s'apparient avec succès que s'ils relèvent de la même classe. En d'autres termes, un appariement entre deux arbres est valide si et seulement si les nœuds correspondants dans l'appariement relèvent tous de la même classe.

En outre, on suppose qu'il existe un ordre total sur l'ensemble des classes et que, en considérant cet ordre, la classe des fils d'un nœud est égale ou immédiatement supérieure à la classe de ce nœud. Cette contrainte définit la notion de stratification et la structure d'ASO (Arbre Stratifié Ordonné). Plus formellement, en définissant un ensemble ordonné **S** (ensemble de "sortes") et une fonction sorte(x) qui associe un élément de **S** à chaque nœud de l'arbre, on peut spécifier un ASO comme étant un arbre ordonné dont la classe de chaque nœud N (sorte(N)) est le successeur immédiat dans **S** de la classe de son père, sauf pour la racine qui n'a pas de père. Dans le cas d'arbres syntaxiques résultant de l'analyse d'un texte en langue naturel, ceci signifie que l'ensemble ordonné **S** peut contenir cinq classes, {*texte, phrase, groupe, catégorie, mot*} avec *texte* < *phrase* < *groupe* < *catégorie* < *mot*.

Etant donné la structure d'ASO, nous pouvons facilement définir une mesure de similarité qui quantifie l'appariement approximatif entre deux ASO (voir partie 3). Grâce à cette mesure, il est possible d'engendrer efficacement un *graphe de similarité* (voir partie 4) qui stocke les sous-arbres les plus proches dans le texte analysé. Et, c'est ce graphe de similarité qui sert d'entré au module de construction des classes de motifs semblables (voir partie 5).

À titre d'illustration, voici l'ASO construit à partir de l'analyse syntaxique d'une proposition tirée de "La comtesse de Tende" de Madame de Lafayette : *Elle exécuta ce qu'elle avait projeté :*[1]

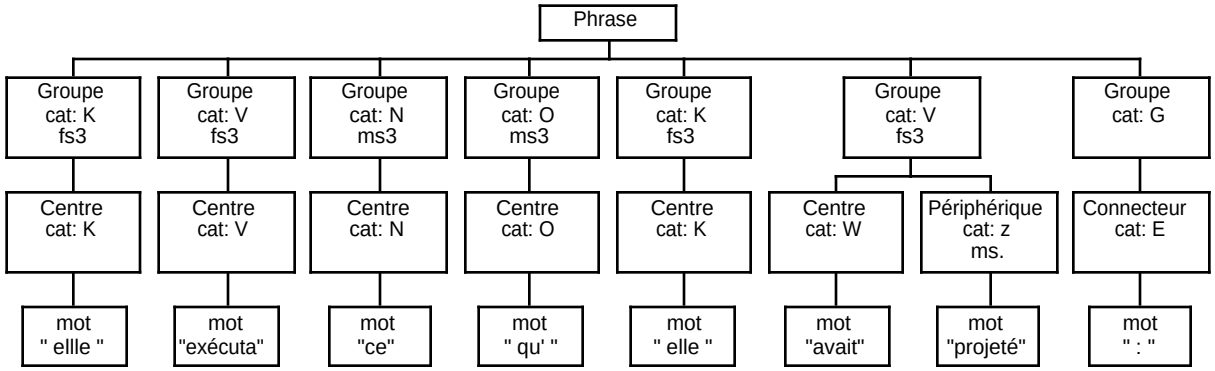

Figure 2 : ASO construit à partir de l'analyse syntaxique d'une proposition simple

# 3 Distance d'édition

Les distances d'édition ont été beaucoup utilisées pour détecter des appariements approximatifs sur des chaînes de caractères (Sankoff & Kruskal 1983), tout particulièrement en biologie moléculaire (Sagot et al. 1996) et en musique (Rolland et Ganascia 1999). Le lecteur intéressé trouvera un panorama complet de ces techniques dans (Crochemore & Rytter 1994). Nous rappellerons simplement quelques principes de base.

**Définition** : une *édition* est une transformation élémentaire qui remplace un caractère dans une chaîne ou plus généralement un nœud dans un arbre, par un autre, éventuellement vide. Par exemple, une *substitution* est une édition qui transforme un caractère d'une chaîne en un autre à la même position. Une *insertion* (respectivement une *destruction*) qui insère (respectivement détruit) un caractère dans une chaîne est aussi une édition.

**Définition** : une *distance d'édition* entre deux chaînes ou deux arbres correspond au nombre minimal d'éditions qui transforme une chaîne dans une autre, ou un arbre dans un autre.

## 3.1 Distance d'édition entre chaînes

Considérons maintenant deux chaînes x et y données sous forme de deux tableaux de taille respective n et m, c'est-à-dire comme x[1..n] et y[1..m]. Il est alors possible de construire une matrice de dimension (n+1)×(m+1) appelée EDIT, où EDIT(i, j) contient la valeur de edit(x[1..i], y[1..j]) pour tout couple (i, j) tel que i∈ [1, n] et j ∈ [1, m], tandis que EDIT(0, j) correspond à l'insertion de y[j] et EDIT(i, 0), à la destruction de x[i]. Sachant que *destruction*(x[i]), *insertion*(y[j]) et *substitution*(x[i], y[j]) correspondent respectivement au coût de la destruction de x[i], de l'insertion de y[j] et de la substitution de x[i] par y[j], une formule simple résume la façon dont les éléments de la matrice sont calculés :

---

[1] Voici le résultat donné par l'analyseur Vergnes-98 : {* 1 1 *}%S K fs3 %C Elle K fs3 {* 2 2 *} %S V fs3 %C exécuta V fs3 {* 3 3 *} %S N ms3 %C ce N ms3 {* 4 4 *} %S O ms3 %C qu' O ms3 {* 5 5 *} %S K fs3 %C elle K fs3 {* 6 6 *} %S V fs3 %C avait W fs3 %P projeté z ms. %G %E : . {* KVNOKV *} {* 123456 6 *} %RD 2 1 K < V %RD 3 2 V < N %RD 4 6 O > V %RA 4 3 N / O %RD 5 3 N < àK %RD 6 5 K < V %Fin. On notera que nous ne retenons ici que l'arbre d'analyse sans les relations entre syntagmes.

$$\text{edit}_{\mathcal{A}}(x[1..i], y[1..j]) = \min\begin{cases} \text{edit}_{\mathcal{A}}(x[1..i-1], y[1..j]) + \textit{destruction}(x[i]) \\ \text{edit}_{\mathcal{A}}(x[1..i-1], y[1..j-1]) + \textit{substitution}(x[i], y[j]) \\ \text{edit}_{\mathcal{A}}(x[1..i], y[1..j-1]) + \textit{insertion}(y[j]) \end{cases}$$

## 3.2 Extension du modèle d'édition aux arbres

Le modèle d'édition peut être aisément étendu aux arbres et aux forêts, mais dans le cas général la complexité inhérente au calcul, rend impossible son utilisation dans des algorithmes d'extraction de motifs. Il a été démontré (Zhang K 1993) que certaines procédures efficaces existent, sous des conditions très strictes, mais les restrictions qu'elles imposent interdisent leur emploi pour des problèmes pratiques. Nous avons montré dans (Ganascia 2001) qu'en limitant les entrées structurées à des ASO, il est possible de construire un nouvel algorithme efficace d'extraction de motifs. Cet algorithme prend pour entrée un immense ASO et engendre des classes de ASO semblables qui ont plusieurs occurrences d'apparitions dans l'ASO d'entrée.

Puisque l'article n'est pas destiné à présenter le détail des algorithmes, nous donnerons juste une idée intuitive de la façon dont la distance d'édition est étendue aux ASO. Toutefois, avant cela, notons que, les arbres étant stratifiés, la classe de chaque nœud se réfère directement au niveau de ce nœud dans l'arbre. De ce fait, la comparaison de deux séquences de nœuds provenant de l'exploration à main gauche de deux arbres est équivalente à la comparaison de ces deux arbres. Tenant compte de cette remarque, la distance d'édition entre deux ASO se réduit à la distance d'édition entre les deux séquences de nœuds produites par l'exploration à main gauche de ces deux ASO.

De façon plus formelle, en dénotant pmg(T) l'exploration à main gauche de l'ASO, c'est-à-dire la séquence des nœuds visités par une exploration en profondeur d'abord, la distance d'édition $\text{edit}_{aso}(T, T')$ entre deux ASO T et T' peut s'exprimer par edit(pmg(T), pmg(T')).

# 4 Construction du graphe de similarité

Utilisant la distance d'édition, un graphe étiqueté appelé le *graphe de similarité* est construit. Ce graphe enregistre la similarité entre les motifs quand ceux-ci ne sont pas trop éloignés, autrement dit quand leur distance est inférieure à une certain seuil. Plus exactement, on dérive une *mesure de similarité* entre motifs à partir de la distance d'édition. Rappelons qu'une mesure de similarité est une fonction binaire positive et symétrique qui atteint son maximum lorsque ses deux arguments sont identiques (voir par exemple Saporta 1990). De façon à ce que les approximations tolérées soient plus ou moins proportionnelles à la longueur du motif, pour ne pas avantager trop fortement les petits motifs, cette longueur a été introduite dans le calcul de la similarité pour pondérer la distance d'édition. Parmi toutes les formules de calcul de similarité envisageables, la suivante s'est avérée tout à fait satisfaisante :

$$s_{\alpha}(i,j) = \frac{1}{1+\alpha \times (\text{edit}(i,j)/\min(\text{length}(i), \text{length}(j)))^4}$$

*Remarque* : $\alpha$ est un nombre positif qui sert de paramètre. La valeur usuelle employée est 0,01.

Le graphe de similarité est d'une importance cruciale ; c'est l'entrée principale du module de classification. De plus, il contient tous les motifs qui généralisent les sous-arbres de l'ASO donné en entrée. Notons que cette généralisation implicite est un point clef dans l'ensemble de l'algorithme, car les motifs engendrés incluent les arbres ordonnés non équilibrés. Dans le cas

des arbres d'analyse syntaxique, ceci signifie que les motifs engendrés peuvent ressembler au suivant (voir figure 3)

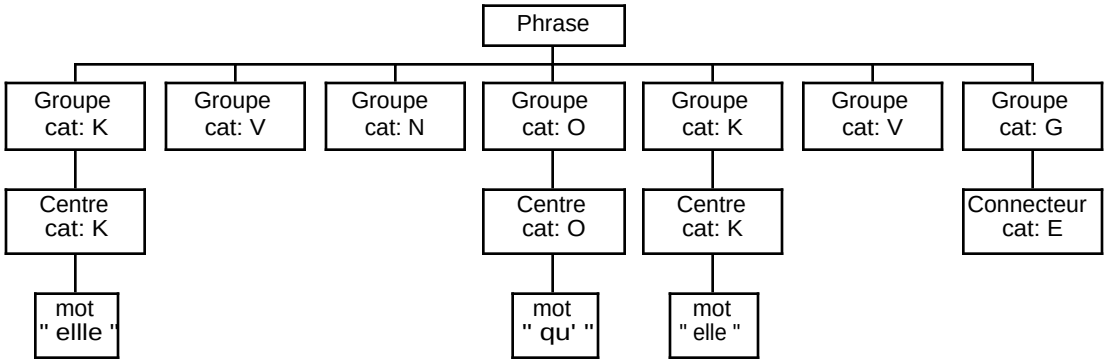


Figure 3 : motif non équilibré

Plus précisément, le graphe de similarité d'un arbre T contient toutes les paires de sous-motifs de T semblables eu égard à la distance d'édition, ou plus exactement à la mesure de similarité, de sorte que cette mesure soit supérieure à un certain seuil. En d'autres termes, ses nœuds correspondent aux sous-arbres de T et ses arcs aux paires de sous-arbres étiquetées par la mesure de similarité. Toutefois, le graphe de similarité ne contient pas toutes les paires de motifs, mais seulement celles de motifs similaires. En prenant en compte les propriétés mathématiques de la mesure de similarité (voir Ganascia 2001), il est possible de réduire considérablement la complexité algorithmique de la procédure de construction du graphe de similarité.

# 5 Algorithme de classification

La dernière étape correspond à l'extraction de motifs, opération qui se subdivise en deux étapes logiques, la *catégorisation* au cours de laquelle des classes de motifs similaires sont construites, et la *description* qui attribut un nom, si possible significatif, à chacune des classes.

## 5.1 Catégorisation

Au cours de la catégorisation, les motifs semblables sont regroupés. Puisque le graphe de similarité enregistre toutes les similarités entre motifs, il est naturel d'extraire les classes de motifs semblables à partir de ce graphe. Toutefois il existe beaucoup de méthodes différentes pour construire de telles classes.

Les approches classiques détectent des sous-graphes fortement connectés (voir Karp et al. 1972). Quelques personnes ont proposé des algorithmes pour détecter des k-cliques appartenant au graphe de similarité, c'est-à-dire des sous-graphe totalement interconnectés comprenant k nœuds. La plupart de ces algorithmes sont employés dans le cas de répétitions exactes entre objets. Il sont généralisables à n'importe quelle relation d'équivalence. Il est aussi possible de les généraliser à des relations non transitives (Soldano et al. 1995), mais c'est au prix d'une très forte complexité.

Nous avons choisi ici une approche beaucoup plus efficace qui fait appel à l'algorithme dit « centre étoile » développé par Gusfield (Gusfield 1993) pour détecter des homologies dans des séquences biologiques et reprise par Rolland et Ganascia (Rolland et Ganascia 1999) pour repérer des motifs musicaux récurrents. Le principe de base est facile à comprendre. Il suffit de définir la notion d'étoile centrée sur un nœud.

**Définition** : une *étoile centrée sur le nœud N* est un graphe dont les arrêtes contiennent le nœud N. Autrement dit, une étoile centrée sur N est composée de tous les nœuds P tels que la paire {P, N} soit une arrête (voir figure 4).


Figure 4 : graphe et étoile centrée sur un nœud N

En faisant appel à une mesure de similarité, l'algorithme « centre étoile » évalue d'abord toutes les étoiles centrées sur tous les différents nœuds N du graphe de similarité en sommant les valeurs des similarités des nœuds de chaque étoile au centre N, puis il prend la meilleure, c'est-à-dire celle dont l'évaluation est la plus élevée. Ensuite les nœuds appartenant à cette étoile sont marqués, et l'algorithme est itéré sur les nœuds non marqués du graphe de similarité, jusqu'à épuisement des nœuds non marqués.

## 5.2 Description

L'étape finale dans l'algorithme de classification porte sur la description de chacune des classes induites. Notre algorithme choisit simplement le motif qui maximise la similarité avec les autres membres de la classe et qui minimise la similarité avec les membres des autres classes. Pour faciliter l'interprétation il est possible d'adjoindre au résultat le fragment original du texte qui est couvert par le motif considéré. Pour illustrer ce point, considérons à nouveau le motif de la figure 3 (dans la partie 4) ; il a été extrait en exécutant notre programme sur un texte français du XVIII$^{e}$ siècle écrit par Madame de Lafayette et intitulé « *La comtesse de Tende* ». Ce motif généralise l'arbre syntaxique donné dans la figure 2 (partie 2.2) qui est issu de l'analyse de la proposition « *Elle exécuta ce qu'elle avait projeté :* ». Il couvre aussi deux autres propositions issues du texte de Madame de Lafayette, « *Il se joignit un nouveau tourment à ceux qu'elle avait déjà :* » et « *Elle vit toute les raisons qu'il avait de l'aimer ;* ». Ceci signifie que la mesure de similarité entre ce motif et l'un des arbres dérivés des arbres syntaxiques de chacune de ces deux propositions est supérieure à un certain seuil. Sous cette condition, ce motif peut être considéré comme une description de la classe regroupant les trois propositions ci-dessus.

# 6 Expérimentations

Le système a été testé sur plus de 100 nouvelles et romans tirés de la littérature classique du XVIII$^{e}$ ou du XIX$^{e}$ siècle, et dont les auteurs sont, entre autres, Madame de Lafayette, Guy de Maupassant, Alphonse Allais, Marcel Schwob, Alphonse Daudet, Eugène Mouton, Hégésippe Moreau, Georges Sand etc. Les textes obtenus sur le site de la bibliothèque électronique de Lisieux (http://www.bmlisieux.com ) furent d'abord soumis à l'analyseur morpho-syntaxique de Jacques Vergnes, puis les séquences résultantes d'arbres syntaxiques furent transformées en un ASO.

## 6.1 Efficacité

D'un point de vue pratique, nous avons étudié la complexité empirique de notre algorithme en reliant le temps d'exécution, exprimé en seconde, à la taille du texte d'entrée, exprimée en milliers de mots. En reportant sur une échelle logarithmique, et en appliquant une régression linéaire, il apparaît clairement (voir figure 5) que le coefficient de régression, c'est-à-dire la pente de la droite de régression, est égal à 2. Ceci prouve, empiriquement, que la complexité temporelle est quadratique.

Ce premier résultat empirique est très satisfaisant car la complexité théorique de notre algorithme est au minimum quadratique avec la taille des entrées. En effet, le calcul du graphe de similarité passe par l'exploration de toutes les paires de motifs. Comme les textes sont représentés par une structure arborescente, le nombre de sous-arbres et donc le nombre de motifs est linéaire avec le nombre de phrases. De la sorte, la complexité globale ne peut pas être inférieure à ce qu'elle est.

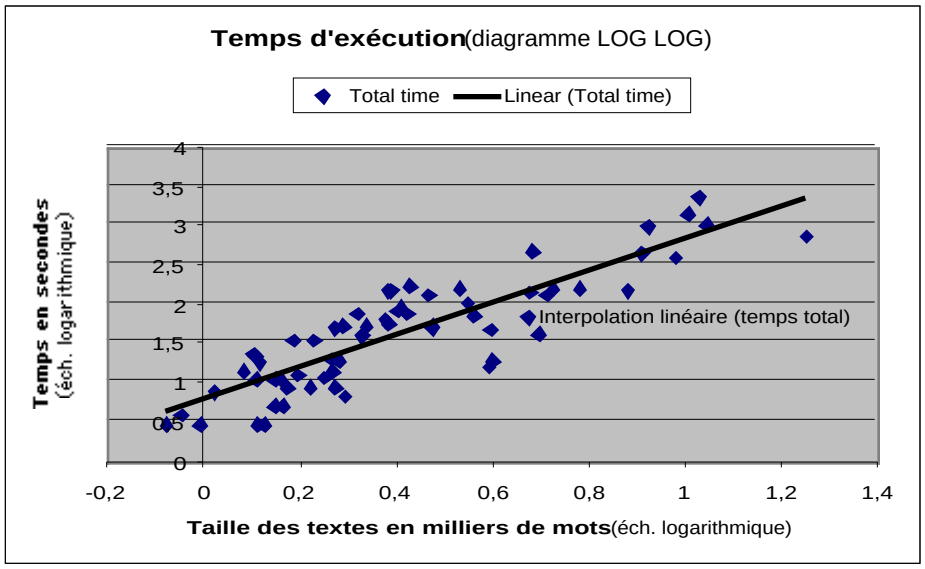


Figure 5 : temps d'exécution (en secondes) en fonction de la taille des textes (en $10^3$ mots).

Pour éviter les malentendus, précisons que dans le cas de répétitions exactes (Karp et al. 1972), la procédure est bien plus efficace, mais que lorsqu'on recherche des répétitions approximatives, il en va tout autrement.

Le système a été implémenté en C++ et testé sur un ordinateur Macintosh G3 avec un processeur à 300 MHz. Le temps d'exécution est compris entre quelques secondes pour extraire des motifs syntaxiques d'une nouvelle tandis que dans le cas d'un roman complet il prend une heure ou plus. Ceci signifie qu'il est possible d'appliquer notre algorithme sur des livres entiers, mais pas sur l'œuvre intégrale d'un auteur. Toutefois, comme nous allons le montrer dans la prochaine sous-partie, les résultats sont déjà suffisamment riches pour être très utiles.

## 6.2 Exemples de motifs extraits

Le programme d'extraction de motifs est complété par une procédure de discrimination. Étant donné deux textes, cette procédure détecte les motifs récurrents couvrant de multiples occurrences d'une structure syntaxique dans le premier texte sans détecter aucune occurrence de cette structure dans le second. Cette procédure a été employée pour détecter des structures syntaxiques caractéristiques d'un auteur, c'est-à-dire qui le distinguent d'autres. L'auteur choisi est Madame de Lafayette avec deux textes, une nouvelle intitulée « *La comtesse de Tende* » et un roman connu, « *La princesse de Clèves* ». Trois auteurs du XIX$^{e}$ furent utilisés dans la procédure de discrimination : Guy de Maupassant, Georges Sand et Marcel Schwob. Plus précisément le corpus employé contient les nouvelles suivantes : de Guy de Maupassant, *La peur (1882), La peur (1884), La veillée, La rempailleuse, Pierrot, En mer, Un normand, Ce cochon de Morin* et *Les sabots,* de Georges Sand, *La fée poussière, Le gnome des huîtres, Le marteau rouge, L'orgue du titan* et *La fée aux gros yeux,* de Marcel Schwob, *Arachné, Béatrice, Sur les dents, Le Dom, L'homme double, Le fort, Gabelous, Parabole, Lilith, Conte des œufs* et *Les portes de l'opium*.

Les trois motifs suivants (voir figure 6) sont présents dans les textes de Madame de Lafayette, sans l'être dans les autres.

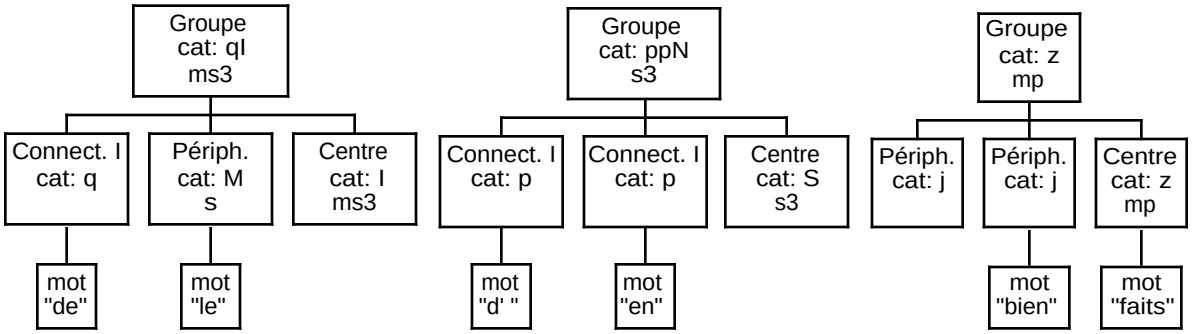


Figure 6 : trois motifs caractéristiques de Madame de Lafayette

Le premier motif couvre, entre autres, les expressions récurrentes suivantes : *"à le servir", "de le supplier", "de l'éviter", "de l'aimer"…* Le second, *"sans en avoir", "d'en avoir", "d'en attendre", "d'en garantir", "d'en faire"...* Quant au troisième, il couvre trois fragments : "*admirablement bien faits*", "*parfaitement bien faits*", "*très bien fait*"

Il y a bien d'autres spécificités syntaxiques caractéristiques de l'œuvre de Madame de Lafayette qui se dégagent de notre analyse. Parmi celles-ci, voici une structure très fréquemment répétée (voir figure 7)

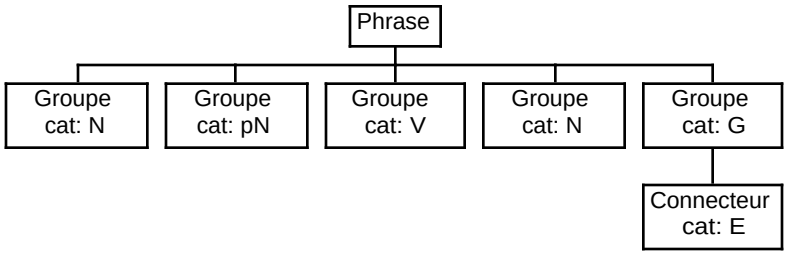


Figure 7 : une figure très fréquente dans l'œuvre de Madame de Lafayette

Cette figure couvre tous les fragments suivants de Madame de Lafayette (et bien d'autres du même auteur) tandis qu'elle est pratiquement absente des textes des trois autres auteurs considérés : "*Le prince de Navarre prit la parole :", "La reine de Navarre avait ses favorites", "Madame de Clèves ne répondit rien", "Monsieur de Nemours prit la reine dauphine",…*

Il existe aussi beaucoup de fragments moins proches qui sont couverts par ce motifs. En voilà quelques uns : "*Madame de Chartres avait une opinion opposée ;", "Le comte de Tende aimait déjà le chevalier de Navarre ;", "Le comte de Tende sentit son procédé dans toute sa dureté ;", "La comtesse reçut ce billet avec joie", …*

À cet égard, notons que les mots « comte », « prince », « madame », « reine », « comtesse », « monsieur » sont bien souvent appariés dans ces motifs. Aucune sémantique n'ayant été donnée à notre algorithme, cet exemple montre comment la syntaxe véhicule parfois la sémantique. Cette étude mérite bien évidemment d'être poursuivie plus longuement.

Et, comme le lecteur peut l'imaginer, il y a bien d'autres hypothèses que l'on souhaiterait explorer à partir des résultats fournis par cet outil.

# 7 Conclusion

Une nouvelle procédure de fouille de texte a été décrite dans cet article. Nous avons prouvé qu'elle était efficiente et féconde. La critique littéraire pourrait bien évidemment bénéficier d'un tel outil. Ainsi, il pourrait être inséré dans un environnement de lecture électronique pour aider les chercheurs ou les étudiants. De même, les didacticiens pourraient le mettre à profit dans l'enseignement des langues. Enfin, les motifs extraits pourraient être réutilisés dans une procédure de synthèse du langage naturel. Ainsi, le traitement du langage naturel pourrait bénéficier des résultats de cette recherche.

# Références

Crochemore M, Rytter W (1994), *Text Algorithms,* "Approximate pattern matching", pp. 237-251.

Ganascia J-G (2001) *Extraction of Recurrent Patterns from Stratified Ordered Trees,* LIP6 internal report.

Gusfield D. (1993) *Efficient methods for multiple sequence Alignment with Guaranteed Error Bounds*, Bull. Math. Biol., 55:141-154.

Karp R M., Miller R E., Rosenberg A L. (1972), *Rapid Identification of Repeated Patterns in Strings, Trees and Arrays,* in Proc. 4th Annu. ACM Symp. Theory of Computing, pp. 125-136.

Landraud A M., Avril J-F, Chrétienne P (1989) *An algorithm for Finding a Common Structure Shared by a Family of Strings,* IEEE transactions on Pattern Analysis and Machine Intelligence, 11 (8), pp. 890-895.

Lecomte J, (1998) BRILL14-JL5 / WINBRILL-0.3, user's manual available at INALF.

Rolland, P-Y, Ganascia J-G, (1999). *Musical Pattern Extraction and Similarity Assessment*. In Miranda, E. (ed.). Readings in Music and Artificial Intelligence. Contemporary Music Studies - Vol 20. Harwood Academic Publishers.

Sagot, Viari A. (1996) A Double Combinatorial Approach to Discovering Patterns in Biological Sequences, *Combinatorial Pattern Matching,* Springer Verlag, LNCS 1075, 168-208

Sankoff D., Kruskal J.B. (1983), *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison,* Addison-Wesley, Reading, Mass..

Saporta G. (1990), *Probabilités, analyse des données et statistique,* TECHNIP Publishing Company, Paris.

Soldano H., Viari A. Champesme M. (1995), Searching for Flexible Repeated Patterns Using a new Transitive Similarity Relation", *Pattern Recognition Letters,* 16:233-246.

Vergne J., (1999) *Analyseur linéaire avec dictionnaire partiel, décembre 1999,* convention d'utilisation de l'analyseur de Jacques Vergne.

Zhang K. (1993) *Fast algorithms for the constrained editing distance between ordered labeled trees and related problems,* report N°361, Department of computer science, University of Western Ontario, London, Ontario, Canada.