EXPLOITATION DES ARBRES FRÉQUENTS DE DÉPENDANCE POUR LA REPRÉSENTATION ET LA CLASSIFICATION AUTOMATIQUE DE TEXTES

par

Ali Meghaoui

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, janvier 2008

Your file Votre référence
ISBN: 978-0-494-42994-5
Our file Notre référence
ISBN: 978-0-494-42994-5

Le 17 janvier 2008

le jury a accepté le mémoire de M. Ali Meghaoui dans sa version finale.

Membres du jury

M. Shengrui Wang
Directeur
Département d'informatique

M. Ernest Monga
Membre
Département de mathématiques

M. André Mayers
Président-rapporteur
Département d'informatique

Sommaire

L'intégration de l'information syntaxique dans la représentation vectorielle des documents s'est avérée une source d'amélioration de la performance des systèmes de classification automatique de documents textuels. Cette information est souvent représentée sous forme d'arbres de dépendance qui peuvent être générés automatiquement par un analyseur syntaxique de la langue naturelle.

Dans ce travail, nous proposons un nouveau modèle de représentation des documents basée sur l'extraction des sous-arbres fréquents d'arbres de dépendance en utilisant l'algorithme de fouille d'arbres FREQT, que nous avons adapté à nos besoins. Dans ce modèle, un document est représenté par l'ensemble de ses phrases, et chaque phrase est représentée à l'aide d'un ensemble de sous-arbres fréquents.

Afin d'appliquer efficacement cette représentation à la classification automatique non supervisée (ou *clustering*) de documents, nous proposons une nouvelle mesure de similarité entre documents basée sur notre méthode de représentation. Ainsi, nous construisons un système de clustering de documents qui englobe notre méthode de représentation, notre mesure de similarité et l'algorithme de clustering hiérarchique par agglomération. Nous évaluons notre système sur des collections de textes bien connues dans la communauté de la classification de textes : la collection Reuters-21578, 20Newsgroups et OHSUMED. Nous montrons sur ces données que notre méthode améliore le clustering de documents. Nous présentons également une évaluation des approches existantes de représentation des documents.

Remerciements

Je voudrais premièrement remercier mon directeur de recherche M. Shengrui Wang pour le suivi permanent de l'évolution de ce travail. Je tiens à lui exprimer mes profondes gratitudes pour ses conseils et ses suggestions judicieuses qu'il m'a prodigués durant toute la période du projet.

Je remercie également toute l'équipe du laboratoire MOIVRE (MOdélisation en Imagerie, Vision et REseaux de neurones) pour toute l'aide qu'ils m'ont apportée sans faillir quand j'en avais besoin.

J'adresse un remerciement particulier pour ma famille pour son soutien moral.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail reçoivent ma profonde gratitude.

Enfin, mes remerciements vont aux professeurs qui m'ont fait l'honneur de participer au jury de ce mémoire.

Table des matières

Liste des tableaux

Liste des figures

Introduction

Le nombre de documents textuels accessibles via les réseaux Intranet et Internet est en croissance exponentielle, et il est difficile d'accéder aux informations importantes et complémentaires sur un sujet de façon efficace. La fouille de textes (*Text Mining*), définie comme le processus d'extraction d'information utile à partir des motifs non manifestés dans de grands corpus de textes, permet de résoudre ce problème de surabondance d'information textuelle. La classification automatique non-supervisée (ou *clustering*) de documents est une des fonctions fondamentales d'un système de fouille de textes. C'est un processus qui permet de diviser une collection de documents textuels de manière automatique en différents groupes où les documents dans le même groupe décrivent le même sujet [17]. Le clustering de documents trouve de nombreuses applications, notamment dans le domaine de la recherche d'information (RI). Dans ces applications, il a été utilisé pour accélérer la recherche et améliorer les taux de rappel[1] et de précision[2] des systèmes de recherche d'information. En effet, au lieu de comparer la requête à chaque document de la collection, les documents sont classifiés à l'avance, et le système ne compare la requête qu'avec un ensemble de clusters retournant à l'utilisateur un ou plusieurs clusters comme résultats. Une autre application du clustering de documents est l'organisation en clusters des résultats de recherche afin de permettre à l'utilisateur d'y accéder facilement. Parmi les systèmes qui appliquent le clustering de documents pour la recherche sur le

[1] Le rappel est la proportion de documents pertinents retrouvés par rapport à l'ensemble de documents pertinents existants dans la collection pour une requête donnée.

[2] La précision est la proportion de documents pertinents sur l'ensemble de documents retrouvés pour une requête donnée.

web, on trouve : les métamoteurs[3] de recherche VIVISIMO[4] et KartOO[5], et les moteurs de recherche WiseNut[6] et Teoma[7].

En général, un système de clustering de documents est constitué de deux composantes principales, une composante de représentation des documents qui vise à capturer l'information utile pour représenter les documents, et une composante de clustering qui se base sur cette représentation pour construire les clusters en utilisant une mesure de similarité et un algorithme de clustering.

La performance d'un système de clustering est affectée directement par la méthode de représentation des documents. L'approche la plus courante est l'approche vectorielle (*Vector Space Model*) où un document est représenté par un vecteur de termes[8] indépendants dans un espace vectoriel [49, 48]. À la base, les termes sont les différents mots apparaissant dans les textes de la collection. Cette approche est appelée aussi la représentation sac de mots (*bag-of-words*). L'approche vectorielle conduit à des vecteurs de termes de très grande dimension, et qui sont pourtant très creux, c'est-à-dire contenant généralement plus de 90% de valeurs nulles. D'autres approches ont été proposées qui utilisent une structure d'arbre de suffixes [63], une structure de graphe [22, 23], ou des ensembles de termes fréquents [6] pour représenter les documents. Ces approches se basent sur l'hypothèse que les mots sont indépendants ignorant les relations syntaxiques que les mots entretiennent entre eux. En fait, une partie de la sémantique des documents s'exprime à travers ces relations syntaxiques. Par exemple, le terme «classification automatique» a une sémantique différente que lorsque les mots «classification» et «automatique» sont pris séparément. On peut donc croire que l'intégration de l'information syntaxique dans la représentation des documents pourrait améliorer le clustering de documents.

[3]Un métamoteur est un outil permettant de faire la recherche en utilisant plusieurs moteurs de recherche simultanément.

[4]VIVISIMO : http ://vivisimo.com/

[5]KartOO : http ://www.kartoo.net/

[6]WiseNut : http ://www.wisenut.com/

[7]Teoma : http ://www.teoma.com/

[8]Un terme est un mot ou un groupe de mots dont les mots sont en général réduits à leur racine.

Dans la littérature, plusieurs travaux se sont intéressés à l'exploitation de l'information syntaxique pour la représentation des documents [2, 5, 39, 51]. Les modèles proposés sont essentiellement basés sur les groupes nominaux ou les entités nominales (noms propres). Malheureusement, les résultats obtenus par ces modèles ne présentent aucune amélioration significative par rapport à l'approche sac de mots. En effet, les groupes nominaux sont sémantiquement significatifs pour représenter le contenu des documents, mais ils ont généralement une faible fréquence par rapport aux mots simples. D'une part, l'utilisation des mots simples peut considérer deux documents non reliés similaires puisqu'ils partagent les mêmes mots. D'autre part, l'utilisation des groupes nominaux peut considérer deux documents reliés dissimilaires puisqu'il est difficile d'avoir des groupes nominaux communs entre eux.

Une autre façon d'exploiter l'information syntaxique est de considérer les relations de dépendance. L'analyse syntaxique de la phrase permet de déterminer les relations de dépendance syntaxique entre les paires de mots qui la composent. Ces relations forment une structure d'arbre, appelée arbre de dépendance.

Dans ce travail, nous nous intéressons à l'amélioration de la représentation des documents en prenant en compte les relations de dépendance entre les mots. Nous proposons un nouveau modèle de représentation des documents basée sur l'extraction des sous-arbres fréquents d'arbres de dépendance en utilisant les techniques de fouille d'arbres (*tree mining*). Dans ce modèle, chaque document est représenté par l'ensemble des phrases qui le constituent et chaque phrase est représentée, à son tour, par un ensemble de sous-arbres fréquents qu'ils sont inclus dans son arbre de dépendance. En prenant en compte des sous-arbres de différentes tailles, cette approche de représentation nous permet de combiner les mots simples et les groupes de mots qui entretiennent des relations de dépendance entre eux, ce qui nous permet d'exploiter l'avantage statistique des mots simples et l'avantage sémantique des groupes de mots. Une fois que les documents sont représentés dans ce modèle, la difficulté principale réside dans la prise en compte effective de cette représentation pour le clustering de documents.

Le clustering de documents se base sur une certaine mesure de similarité entre documents. Plusieurs mesures de similarité entre documents, conçues essentiellement pour le modèle vectoriel, ont été proposées, comme par exemple, la mesure du cosinus et la mesure de Jaccard. La mesure de similarité est fortement liée à la méthode de représentation des textes adoptée. Il serait donc utile de définir une mesure de similarité qui prend en compte les spécificités de notre approche de représentation. Pour cela, nous proposons une nouvelle mesure de similarité basée sur la similarité entre les phrases constituant les documents.

Dans la littérature, plusieurs algorithmes classiques de clustering de données ont été utilisés pour le clustering de documents, à savoir les algorithmes : K-moyennes, hiérarchique par agglomération, *buckshot* et *bisecting k-means*. Ces algorithmes seront présentés en détail dans le chapitre 1. En général, les algorithmes de clustering sont très reliés aux données. Dans ce travail, nous avons choisi d'utiliser l'algorithme hiérarchique par agglomération pour le clustering de documents.

Enfin, nous construisons un système de clustering de documents pour évaluer notre approche de représentation et notre mesure de similarité. Le système englobe les quatre composantes suivantes :

1. L'unité d'analyse de dépendance syntaxique des phrases de la collection ;
2. L'unité de représentation des documents ;
3. L'unité de calcul de similarité entre documents ;
4. L'unité du clustering de documents.

Notre document est organisé comme suit : nous commençons donc naturellement par présenter l'état de l'art en clustering de documents dans le chapitre 1. Dans le chapitre 2, nous présentons notre système de clustering et nous détaillons notre approche de représentation des documents et notre mesure de similarité. Ensuite, nous présentons une étude expérimentale sur les différentes approches de représentation des documents

avec une évaluation de notre système de clustering dans le chapitre 3. Finalement, nous concluons notre travail sur des ouvertures vers d'autres travaux dans le même sens.

Chapitre 1

État de l'art du clustering de documents

1.1 Introduction

Depuis de nombreuses années, plusieurs travaux ont été effectués dans le domaine de la classification de documents textuels. De manière générale, ces travaux s'intéressaient à développer des méthodes permettant d'associer une classe à un document. Dans la littérature, on distingue deux types de classification : la classification supervisée où les classes existent à priori, et la classification non supervisée (*ou clustering*) qui doit découvrir les classes non connues à l'avance.

L'objectif du clustering de documents est de séparer un ensemble de documents en différents groupes (*ou clusters*) tels que les documents les plus similaires appartiennent au même groupe, et que les documents les plus différents soient séparés dans différents groupes. En général, un système de clustering de documents est constitué de deux composantes principales : une composante de représentation des documents et une composante

de clustering. La première composante convertit les documents textes dans une structure appropriée (habituellement un vecteur). Cette structure est utilisée par la deuxième composante pour déterminer le meilleur regroupement des documents en se basant sur une mesure de similarité et un algorithme de clustering.

Dans ce chapitre, nous présentons l'état de l'art en clustering de documents. Dans la section 1.2, nous présentons les travaux reliés à la représentation des documents. Les différentes mesures de similarité sont présentées dans la section 1.3. Ensuite, nous donnons un aperçu sur les différents algorithmes de clustering de documents dans la section 1.4.

1.2 Représentation des documents

Dans le domaine de la recherche d'information, le modèle de représentation des documents le plus connu est le modèle vectoriel où chaque document est représenté par un vecteur dans un espace vectoriel. Les dimensions de l'espace vectoriel correspondent aux termes présents dans la collection de documents. Les termes peuvent être des mots simples ou des groupes de mots décrivant le contenu des documents. Les composantes du vecteur représentant un document sont les poids des termes dans ce document. On distingue trois méthodes pour calculer ce poids :

1. **Vecteur binaire :** La représentation par vecteur binaire est la représentation la plus ancienne des documents. Les composantes du vecteur représentant un document informent sur la présence (valeur égale à 1) ou l'absence (valeur égale à 0) d'un terme dans le document. Cette représentation ne donne aucune information ni sur la fréquence des termes, ni sur la longueur des documents.
2. **Vecteur fréquentiel :** La représentation fréquentielle (*Term Frequency : Tf*) est une extension de la représentation binaire. Les composantes du vecteur représentant un document indiquent le nombre d'occurrences d'un terme dans le document. Cette représentation ne prend pas en considération l'importance du terme par rapport à

toute la collection des documents.

3. **Vecteur TF-IDF :** La représentation TF-IDF (*Term Frequency, Inverse Document Frequency*) est la représentation la plus utilisée dans le domaine de recherche d'information [28, 47]. Le poids d'un terme est calculé en fonction de sa fréquence d'apparition dans le document (Tf) pondérée par la fréquence d'apparition du terme dans toute la collection (Idf). Ainsi les termes qui apparaissent dans la majorité des documents de la collection auront moins de poids que les termes qui apparaissent dans quelques documents.

   La formule de Tf-Idf est la suivante :

$$TfIdf = Tf \times Idf = Tf \times \log(\frac{\mid D \mid}{df})$$

   où $\mid D \mid$ est le nombre total des documents dans la collection, et df est le nombre de documents contenant le terme.

   Habituellement, les vecteurs sont normalisés afin d'éviter les problèmes posés par la différence dans les longueurs des documents.

Dans le cadre du modèle vectoriel, on distingue deux approches de représentation des documents selon le choix des termes : la représentation sac de mots *(Bag-of-words)* où les termes sont des mots simples et la représentation par séquences de mots *(Bag-of-phrases)* où les termes sont des groupes de mots. Les outils de traitement automatique de la langue naturelle (TALN) sont utilisés pour mettre en œuvre ces deux approches de représentation.

1.2.1 Outils de traitement automatique de la langue naturelle

Nous présentons ici les outils de traitement de la langue naturelle utilisés dans le cadre de la représentation des documents textuels. Le traitement automatique de la langue naturelle est connu depuis longtemps dans plusieurs domaines comme le traitement de la

parole, la traduction automatique, la recherche d'information, la génération de résumés, l'indexation automatique, etc. Il s'agit de mettre à profit des connaissances linguistiques afin d'améliorer le repérage des unités significatives du contenu de documents, en prenant compte du fait que ces derniers sont constitués de mots et non pas simplement de chaînes de caractères quelconques. On distingue habituellement six niveaux d'analyse du traitement de la langue naturelle :

- Analyse phonétique : l'étude de l'identification des mots à partir du son ;
- Analyse morphologique : l'étude de la forme des mots ;
- Analyse syntaxique : l'étude de la structure de la phrase et les relations entre les mots ;
- Analyse sémantique : l'étude du sens des mots et des phrases ;
- Analyse pragmatique : l'étude du sens des phrases en fonction du contexte ;
- Analyse du discours : l'étude des interactions linguistiques complexes.

Les niveaux de l'analyse morphologique et de l'analyse syntaxique sont ceux sur les quels nous nous sommes concentrés pour améliorer la représentation des documents.

Les principaux outils de traitement automatique de la langue naturelle utilisés pour le problème de représentation des documents sont : le segmenteur, l'analyseur morphologique, l'étiqueteur des parties du discours, l'identificateur des groupes grammaticaux, l'analyseur de dépendance syntaxique, et l'analyseur sémantique.

Segmenteur

La segmentation (*tokenization*) est une étape préliminaire qui sert à préparer le texte pour que les différents analyseurs puissent opérer. Elle consiste à structurer le texte en passant d'une suite de caractères à des mots et phrases. Le texte est divisé en un ensemble d'unités, appelées «segments». La méthode la plus simple de segmentation est de prendre les caractères blancs et les signes de ponctuation comme caractères séparateurs

des segments. La segmentation nous donne deux types de résultats : le premier correspond à des unités qui ont une structure de caractère bien spécifique, comme les signes de ponctuation, nombres, dates, sigles et acronymes ; le deuxième correspond aux mots qui vont subir une analyse morphologique. Un exemple d'un segmenteur en phrases est MXTERMINATOR[1], implémenté par Adwait Ratnaparkhi [46]. Un autre segmenteur est Penn Treebank[2].

Analyseur morphologique

L'analyse morphologique est la première grande étape dans les traitements automatiques de la langue écrite. À ce niveau, l'objectif est de ramener les variantes morphologiques (flexion, dérivation, conjugaison) d'un mot à sa forme de base. La méthode la plus simple pour effectuer l'analyse morphologique est la racinisation (ou *stemming*). La racinisation vise à réduire les mots à leur racine la plus simple. Par exemple, les mots *calculate*, *calculated*, *calculating*, *calculator* et *calculators* seront réduits en *calcul*. L'algorithme de racinisation le plus utilisé est celui de Porter[3] [43]. Une autre méthode d'analyse morphologique plus précise que la racinisation est la lemmatisation. Elle vise à convertir le mot à sa forme canonique appelée «lemme» en se basant sur des outils linguistiques. Par exemple, les mots *calculate*, *calculated* et *calculating* ont le même lemme *calculate*, et *calculator* et *calculators* ont le même lemme *calculator*. Morph est un exemple de lemmatiseur d'anglais. Il est utilisé dans le système de traitement de la langue naturelle GATE (General Architecture for Text Engineering)[4][19].

[1]MXTERMINATOR est disponible à : ftp ://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz
[2]Penn Treebank est disponible à : http ://www.cis.upenn.edu/~treebank/tokenization.html.
[3]Porter stemmer est disponible à : http ://www.tartarus.org/~martin/PorterStemmer/
[4]La page d'accueil de GATE est : http ://gate.ac.uk/

Étiqueteur des parties du discours

L'étiquetage des parties du discours (*Part-of-Speech Tagging*) consiste à affecter une étiquette à chaque mot indiquant sa catégorie syntaxique (nom, verbe, adjectif, ...). L'étiquetage est utilisé dans l'analyse syntaxique de la phrase. Les étiqueteurs les plus utilisés sont TBL Tagger[5] de Brill, MXPOST[6] (Maximum Entropy POS Tagger) développé par Adwait Ratnaparkhi.

Identificateur des groupes grammaticaux

L'identification des groupes grammaticaux *(chunking)* consiste à reconnaître les groupes grammaticaux de la phrase (groupes nominaux, groupes verbaux, groupes adjectivaux, groupes adverbiaux, groupes pronominaux) en utilisant l'information d'étiquetage et des règles grammaticales de construction de la phrase. Un exemple de *chunker* syntaxique est LT CHUNK.

Analyseur de dépendance syntaxique

Le but de l'analyseur de dépendance syntaxique[7] est de formaliser la structure syntaxique de la phrase en construisant un arbre de dépendance syntaxique. Il se base sur des règles grammaticales de construction de la phrase, en prenant en compte l'agencement des mots dans la phrase. Sleator et Temperley (1991) ont proposé un analyseur syntaxique de la langue anglaise, basé sur une grammaire de dependance appelé Link Grammar Parser[8][53]. Cet analyseur offre de bonnes performances et semble robuste. Il traite avec succès des phrases complexes, en tolérant la présence de mots inconnus.

[5]La page d'accueil de Brill's tagger est : http ://www.cs.jhu.edu/~brill/.

[6]MXPOST est disponible à http ://www.cis.upenn.edu/~dbikel/software.html

[7]La notion de dépendance syntaxique est reliée à la fonction du mot dans la phrase (exemple : sujet du verbe), elle est définie dans la section 2.4

[8]Link Grammar Parser est disponible à http ://www.link.cs.cmu.edu/link/.

Michael Collins (1999) a proposé Collins Parser[9], basé sur des méthodes statistiques. LoPar[10] est un autre analyseur syntaxique proposé par Schmid [50]. Stanford Parser[11] est un analyseur statistique qui fournit une sortie sous forme des relations de dépendance syntaxique.

Analyseur sémantique

L'analyse sémantique est le niveau d'analyse du sens des mots et des phrases, que introduit des relations sémantiques entre eux. À ce niveau, on s'interesse aux problèmes de synonymes, locutions ou mots composés, homographes, etc. Généralement, l'analyseur sémantique est couplé à des dictionnaires et des thesaurus pour résoudre ces problèmes. Le thesaurus est un répertoire de mots ayant entre eux des relations sémantiques dans un domaine de connaissance. Le thésaurus anglais le plus connu est WordNet[12]. WordNet est un projet mené depuis 1985 à Princeton. Il offre un réseau sémantique de la langue anglaise. Les nœuds sont constitués par des ensembles de synonymes (ou *synsets*), correspondant au sens d'un ou plusieurs *lemmes*. Un *synset* est défini d'une façon différentielle par les relations qu'il entretient avec les sens voisins. WordNet nous sert à déterminer les différents sens d'un mot donné.

1.2.2 Représentation sac de mots

La représentation sac de mots utilise le mot comme unité de base pour représenter les documents. Chaque mot dans la collection de documents correspond à une dimension dans l'espace vectoriel. L'utilisation des mots pose un certain nombre de problèmes. Pour une collection de taille raisonnable, le nombre de mots peut être de plusieurs dizaines de milliers, c'est-à-dire un espace vectoriel de grande dimension, ce qui demande beaucoup

[9]Collins parser est disponible à http ://people.csail.mit.edu/mcollins/code.html.
[10]LoPar est disponible à http ://www.ims.uni-stuttgart.de/projekte/gramotron/resources.html.
[11]Stanford Parser est disponible à : http ://www-nlp.stanford.edu/software/lex-parser.shtml
[12]La page d'accueil du thésaurus WordNet est http ://wordnet.princeton.edu/.

de mémoire et de temps de calcul, et nous empêche d'utiliser efficacement les algorithmes de clustering. Pour réduire la dimension de l'espace de représentation des documents, il existe en général deux manières : par selection d'attributs (ou mots) où les attributs considérés discriminants sont conservés et les attributs inutiles pour distinguer les documents différents sont éliminés ; et par extraction d'attributs où à partir des attributs de départ, des nouveaux attributs sont créés, en faisant soit des regroupements ou des transformations.

La méthode la plus simple pour réduire l'espace de représentation des documents est de raffiner la représentation par :

1. Une étape de pré-traitement : Elle consiste à supprimer les signes de ponctuation, les nombres et les segments non alphabétiques (les segments constitués de moins de deux lettres). Cette étape est motivée par le fait que ces segments ne sont pas liés au contenu des documents et peuvent donc être négligés.
2. La suppression des mots vides (*stopwords*) : Il s'agit de supprimer tous les mots qui n'apportent aucune information. Ces mots sont généralement trop fréquents comme les articles, les prépositions, etc. Cette opération réduit en moyenne l'espace de représentation de 50%.
3. Le racinisation : Il s'agit de remplacer tous les mots par leur racine (*stem*) en utilisant un «*stemmer*». La racinisation réduit en moyenne l'espace de représentation de 30%.
4. L'élimination des mots rares : Il s'agit de ne prendre que les mots qui apparaissent au moins dans un nombre α de documents, où α est un seuil pré-défini. L'idée derrière cette technique est que les mots rares n'aident pas à identifier les clusters appropriés, et ils peuvent ajouter du bruit à la mesure de distance.

Un autre inconvénient de la représentation sac de mots est que les mots sont considérés indépendants, et les relations entre eux sont ignorées. Les mots sont des unités trop petites lorsqu'il s'agit de représenter sémantiquement le contenu des documents. Plusieurs

travaux ont été effectués pour contourner cet inconvénient en créant des attributs plus complexes que les mots et qui portent plus de sémantique. L'idée est de prendre des groupes de mots au lieu des mots isolés.

1.2.3 Représentation par séquences de mots

Dans cette représentation, les documents sont représentés par des séquences de mots. La combinaison des mots se fait soit d'une manière statistique (les séquences de mots qui apparaissent fréquemment dans la collection), soit d'une manière syntaxique (les séquences de mots qui sont syntaxiquement correctes) :

Séquences de mots «statistiques»

L'utilisation des séquences de mots «statistiques» dans le domaine de recherche documentaire est connue depuis longtemps. Récemment, quelques travaux ont été effectués dans ce sens pour le clustering. Les séquences de mots à identifier sont soit d'une longueur (nombre de mots dans la séquence) fixe (*N-grammes*) ou d'une longueur arbitraire.

— **Longueur fixe :** Les modèles basés sur les *N-grammes* représentent les documents par des groupes de N mots. Skogramr et Olsson (2002) ont utilisé les *bi-grammes* ($N = 2$) et *tri-grammes*($N = 3$) au lieu des mots simples (*uni-gramme*)[52]. Modha et Spangler (2003) ont combiné les mots simples avec des séquences de deux et de trois mots qui ont une fréquence d'apparition supérieure à un certain seuil minimal [41]. Broder et al. (1997) ont utilisé une approche «*shifting N-grams*» pour identifier les documents identiques dans le Web [10].

Des modèles basés sur les *N-grammes* de lettres, c'est-à-dire des séquences de N caractères consécutifs, ont été aussi proposés pour représenter les documents [29, 31]. L'ensemble des *N-grammes* d'un terme est l'ensemble de séquences que l'on obtient en

déplaçant une fenêtre de N cases sur le label d'un terme, caractère par caractère, par exemple, l'ensemble de tri-grammes ($N = 3$) du terme (*document*)={_do, doc, ocu, cum, ume, men, ent, nt_}, où le caractère "_" représente le caractère blanc. Wang et al. (2002) ont aussi utilisé cette approche pour le traitement de corpus multilingues [56].

— **Longueur arbitraire :** Plusieurs méthodes ont été proposées pour extraire les séquences de mots utiles d'une longueur arbitraire. Zamir et Etzioni (1998) ont proposé d'utiliser un arbre de suffixes (*suffix tree*) pour identifier toutes les séquences de mots communes entre les documents [63]. Deux documents qui partagent plus de séquences de mots sont plus similaires. Leurs travaux sur les arbres de suffixes ont été poursuivis par Hammouda et Kamel (2002) qui ont proposé d'utiliser une structure de graphe «*Document Index Graph (DIG)*»[23]. Les séquences de mots formant un chemin dans ce graphe sont identifiées pour mesurer la distance entre les documents. Bakus et al. (2002) ont utilisé une technique d'extraction des «*phrase grammar*» pour identifier les phrases pertinentes dans la base d'apprentissage et les utiliser comme des caractéristiques pour le clustering de documents [4].

Séquences de mots «syntaxiques»

L'arrangement des mots dans une phrase ne peut pas être arbitraire, elle suit la syntaxe de la langue. Les mots sont combinés dans des groupes, où chaque groupe a une fonction dans la phrase : groupes nominaux, groupes verbaux, groupes adjectivaux, groupes adverbiaux, groupes pronominaux.

Arampatzis et al. (2000) ont trouvé des bons résultats, en utilisant des groupes de mots composés des paires de mots adjacents (noms et adjectifs) ou des groupes nominaux [2]. Wen et al. (2001) ont proposé l'utilisation d'un système de reconnaissance des groupes nominaux basé sur un dictionnaire des groupes nominaux les plus connus [58]. Basili et al. (2000) ont montré que l'enrichissement du modèle vectoriel «*bag-of-words*» avec

des locutions complexes améliore le clustering [5]. Clifton et al. (2004) ont utilisé un module de reconnaissance des entités nominales (noms propres) pour identifier les noms des personnes, des lieux, et des organisations, afin de remplacer les mots simples dans le modèle vectoriel par des entités nominales [14]. Liu et al. (2002) ont suggéré de combiner les mots simples et les entités nominales [36]. Chu et al. (2003) ont proposé d'utiliser un ensemble de groupes syntaxiques du domaine de corpus (le domaine médical dans leur cas) pour représenter les documents [13].

D'autres modèles basés sur toute la phrase (*Sentence-Based Models*) ont été proposés. Pullwit et Der (2001) ont proposé une méthode de double clustering. Ils commencent par le clustering de toutes les phrases dans un ensemble de catégories. Puis, les documents sont représentés par ces catégories [44].

1.3 Mesures de similarité

Le but du clustering de documents est de regrouper les documents similaires dans le même cluster et les documents dissimilaires dans des clusters différents en se basant sur une mesure de similarité. En général, les mesures de similarité sont conçues pour le modèle vectoriel. On peut les diviser en deux catégories : mesures de similarité entre objets, et mesures de similarité entre clusters.

1.3.1 Mesures de similarité entre objets

Le notion de similarité entre objets est souvent ramenée à une fonction de distance entre paires d'objets. Ainsi, deux objets seront considérés similaires s'ils sont proches selon cette distance, et deux objets seront considérés différents s'ils sont séparés par une grande distance. Soit X un ensemble de données, une fonction de distance ($dist$) devrait satisfaire les quatres propriétés suivantes [25] :

1. $dist(x_i, x_j) \geq 0, \forall x_i, x_j \in X$ (positive).
2. $dist(x_i, x_j) = d(x_j, x_i), \forall x_i, x_j \in X$ (symétrique).
3. $\forall x_i, x_j \in X : dist(x_i, x_j) = 0 \Leftrightarrow x_i = x_j$.
4. $d(x_i, x_k) \leq dist(x_i, x_j) + dist(x_j, x_k), \forall x_i, x_j, x_k \in X$ (l'inégalité triangulaire).

Notons que la notion de similarité va dans le sens inverse que celle de la distance, à savoir que la similarité entre deux objets est d'autant plus grande que les objets sont semblables. Elle a une valeur comprise entre 0 et 1, et elle ne possède pas la propriété d'inégalité triangulaire (propriété 4).

Dans le modèle vectoriel, les documents sont représentés par des vecteurs de termes dans un espace vectoriel. Le calcul de distance entre deux documents revient au calcul de la distance entre les deux vecteurs qui les représentent.

Soient deux documents d_i et d_j représentés par deux vecteurs $\overrightarrow{d_i}$ et $\overrightarrow{d_j}$ de longueur n, où $\overrightarrow{d_i} = (d_{i1}, d_{i2}, \ldots, d_{in})$ et $\overrightarrow{d_j} = (d_{j1}, d_{j2}, \ldots, d_{jn})$.

Mesures de distance

Les différentes mesures de distance utilisées pour calculer la similarité entre deux documents sont :

- Distance de Manhattan *(ou city-block)* :

$$dist(\overrightarrow{d_i}, \overrightarrow{d_j}) = \|\overrightarrow{d_i} - \overrightarrow{d_j}\|_1 = \sum_{k=1}^{n} |d_{ik} - d_{jk}|$$

- Distance euclidienne (La norme L_2) :

$$dist(\overrightarrow{d_i}, \overrightarrow{d_j}) = \|\overrightarrow{d_i} - \overrightarrow{d_j}\|_2 = \sqrt{\sum_{k=1}^{n} (d_{ik} - d_{jk})^2}$$

- Distance de Minkowski :

$$dist(\overrightarrow{d_i}, \overrightarrow{d_j}) = \|\overrightarrow{d_i} - \overrightarrow{d_j}\|_p = \sqrt[p]{\sum_{k=1}^{n} |d_{ik} - d_{jk}|^p}$$

Où p est un entier positif.

- Pour $p = 1$: distance de Manhattan ;
- Pour $p = 2$: distance euclidienne ;
- Pour $p \to \infty$, distance de Chebychev :

$$dist(\overrightarrow{d_i}, \overrightarrow{d_j}) = \|\overrightarrow{d_i} - \overrightarrow{d_j}\|_\infty = \max_{1 \leq k \leq m} |d_{ik} - d_{jk}|$$

Coefficients d'association

Autres mesures de similarité utilisées dans le domaine de recherche d'information sont les coefficients d'association, à savoir le coefficient de Dice et le coefficient de Jaccard. Ces deux derniers sont basés sur les caractéristiques communes entre les deux documents. Initialement, ils ont été développés pour des données binaires. Puis, ils ont été généralisés pour des données non binaires par les formules suivantes :

- Coefficient de Dice :

$$Dice(\overrightarrow{d_i}, \overrightarrow{d_j}) = \frac{2 \sum_{k=1}^{n} d_{ik} d_{jk}}{\sum_{k=1}^{n} d_{ik}^2 + \sum_{k=1}^{n} d_{jk}^2}$$

- Coefficient de Jaccard :

$$Jaccard(\overrightarrow{d_i}, \overrightarrow{d_j}) = \frac{\sum_{k=1}^{n} d_{ik} d_{jk}}{\sum_{k=1}^{n} d_{ik}^2 + \sum_{k=1}^{n} d_{jk}^2 - \sum_{k=1}^{n} d_{ik} d_{jk}}$$

Le coefficient de Dice normalise le produit scalaire des deux vecteurs en divisant par la somme des composantes non nulles des deux vecteurs, et qui donne une valeur comprise entre 0 et 1. Le coefficient de Jaccard normalise le produit scalaire des deux vecteurs en divisant par la somme des composantes non nulles de l'union des deux vecteurs, ce qui donne une mesure de similarité plus faible que celle du coefficient de Dice. Différentes variations très similaires existent de ces deux coefficients dans [21, 48, 55].

Un autre coefficient d'association est la distance *MinMax* utilisé dans les expérimentations de Bellot et El-Bèze (2000) [8] :

- MinMax :

$$MinMax(\overrightarrow{d_i}, \overrightarrow{d_j}) = 1 - \frac{\sum_{k=1}^{n} min(d_{ik}, d_{jk})}{max(\sum_{k=1}^{n} d_{ik}, \sum_{k=1}^{n} d_{jk})}$$

Le cosinus

Le cosinus est une mesure de similarité largement utilisée dans le domaine de clustering de documents. La similarité entre deux documents d_i et d_j est le cosinus de l'angle formé par les deux vecteurs $\overrightarrow{d_i}$ et $\overrightarrow{d_j}$ qui les représentent, un petit angle indique qu'ils sont similaires [48]. Mathématiquement, le cosinus de deux vecteurs est le produit scalaire des deux vecteurs normalisé par le produit de leurs normes :

$$Cosine(\overrightarrow{d_i}, \overrightarrow{d_j}) = \frac{\overrightarrow{d_i} \cdot \overrightarrow{d_j}}{\|\overrightarrow{d_i}\| \cdot \|\overrightarrow{d_j}\|} = \frac{\sum_{k=1}^{n} d_{ik} \cdot d_{jk}}{\sqrt{\sum_{k=1}^{n} d_{ik}^2} \cdot \sqrt{\sum_{k=1}^{n} d_{jk}^2}}$$

Où $(\overrightarrow{d_i} \cdot \overrightarrow{d_j})$ est le produit scalaire de $\overrightarrow{d_i}$ et $\overrightarrow{d_j}$, et $\|\overrightarrow{d_i}\|$ et $\|\overrightarrow{d_j}\|$ sont les normes de $\overrightarrow{d_i}$ et $\overrightarrow{d_j}$ respectivement.

Coefficient de corrélation de Pearson

Le coefficient de corrélation de Pearson est une autre mesure de similarité. Il est défini comme le cosinus de l'écart à la moyenne, il est donné par la formule suivante :

$$Pearson(\overrightarrow{d_i}, \overrightarrow{d_j}) = \frac{1}{2}(\frac{\sum_{k=1}^{n}(d_{ik} - \overline{d_i}) \cdot (d_{jk} - \overline{d_j})}{\sqrt{\sum_{k=1}^{n}(d_{ik} - \overline{d_i})^2} \cdot \sqrt{\sum_{k=1}^{n}(d_{jk} - \overline{d_j})^2}} + 1)$$

avec $(\overline{d_i} = \frac{\sum_{k=1}^{n} d_{ik}}{n})$ et $(\overline{d_j} = \frac{\sum_{k=1}^{n} d_{jk}}{n})$.

1.3.2 Mesures de similarité entre clusters

Quelques algorithmes de clustering, notamment les algorithmes hiérarchiques par agglomération, ont besoin de calculer la similarité non seulement entre les objets individuels mais aussi entre les clusters. Souvent, le problème de calcul de similarité entre deux clusters revient au calcul de similarité entre deux objets qui les représentent. Pour déterminer ces représentants, différentes méthodes ont été proposées dont les plus utilisées dans le domaine de clustering de documents sont présentées ci-dessous.

Considérons deux clusters C_p et C_q. Supposons que le nombre de documents dans C_p est n_p, et le nombre de documents dans C_q est n_q. En considérant d_i un élément de C_p, et d_j un élément de C_q, la distance entre d_i et d_j est définie par $dist(d_i, d_j)$. Les méthodes suivantes sont utilisées pour définir une mesure de distance (similarité) $distance(C_p, C_q)$ entre les deux clusters C_p et C_q :

Plus proche voisin (*Nearest-Neighbor*)

La distance entre deux clusters est définie comme la distance minimale entre toutes les paires d'éléments des deux clusters :

$$distance(C_p, C_q) = \min_{\substack{d_i \in C_p \\ d_j \in C_q}} \{dist(d_i, d_j)\} \tag{1.1}$$

Voisin le plus éloigné (*Farthest neighbor*)

La distance entre deux clusters est définie comme la distance maximale entre toutes les paires d'éléments des deux clusters :

$$distance(C_p, C_q) = \max_{\substack{d_i \in C_p \\ d_j \in C_q}} \{dist(d_i, d_j)\} \tag{1.2}$$

Moyen de groupe (*Group average*)

La distance entre deux clusters est définie comme la distance moyenne entre toutes les paires d'éléments des deux clusters :

$$distance(C_p, C_q) = \frac{\sum_{d_i \in C_p} \sum_{d_j \in C_q} dist(d_i, d_j)}{n_p \cdot n_q} \tag{1.3}$$

Centroïde

La distance entre deux clusters est définie comme la distance entre leurs centroïdes. Si g_p est le centroïde du cluster C_p, et g_q est le centroïde du cluster C_q, alors la distance entre le cluster C_p et le cluster C_q est la distance entre g_p et g_q :

$$distance(C_p, C_q) = dist(g_p, g_q) \tag{1.4}$$

Avec : $g_i = \frac{\sum_{d_j \in C_i} d_j}{n_i}$.

Médoïde

La distance entre deux clusters est définie comme la distance entre leurs médoïdes. Le médoïde est l'élément le plus central dans le cluster. Si m_p est le médoïde du cluster C_p, et m_q est le médoïde du cluster C_q, alors la distance entre le cluster C_p et le cluster C_q est la distance entre m_p et m_q :

$$distance(C_p, C_q) = dist(m_p, m_q) \tag{1.5}$$

Avec : $m_i = \arg\min_{d_j \in C_i} \sum_{d_k \in C_i} dist(d_j, d_k)$.

1.4 Algorithmes de clustering de documents

Dans la littérature, des algorithmes généraux de clustering de données ont été proposés pour le clustering de documents. Ces algorithmes peuvent être divisés en méthodes hiérarchiques et méthodes par partition. D'autres algorithmes sont conçus spécialement pour le problème de clustering de documents. Selon la définition d'un cluster, les algorithmes de clustering peuvent être aussi divisés en algorithmes de *hard clustering* où chaque document ne peut appartenir qu'à un seul cluster, et algorithmes de *soft clustering* où chaque document a une probabilité donnée d'appartenir à chacun des clusters créés.

1.4.1 Algorithmes généraux de clustering de données

Algorithmes hiérarchiques

Le clustering hiérarchique construit une hiérarchie de clusters représentée par un arbre de clusters appelé «dendrogramme» [9]. Il existe deux principales méthodes pour construire cet arbre : une méthode agglomérative (ascendante) démarre avec autant de clusters que d'objets initiaux. Puis, les deux clusters les plus similaires sont successivement fusionnés jusqu'à ce qu'un critère d'arrêt soit vérifié. Et une méthode divisive (descendante) démarre avec un seul cluster contenant tous les objets. Puis, un cluster est récursivement sélectionné et divisé en clusters plus petits et plus différents jusqu'à ce qu'un critère d'arrêt soit vérifié. Un pseudo-code de l'algorithme hiérarchique par agglomération est décrit à la page suivante.

Algorithme 1 Algorithme de Clustering Hiérarchique Agglomérative (HAC)

Entrée : un ensemble d'objets $X = \{x_1, x_2, \ldots, x_n\}$, et un critère d'arrêt ε.
Sortie : un ensemble de clusters $\{C_1, C_2, \ldots, C_k\}$.
1. Pour chaque élément x_i dans X, on construit un cluster Y_i ($Y_i = \{x_i\}$) ;
2. Calculer la distance entre chaque paire de clusters Y_a et Y_b ;
3. Sélectionner deux clusters les plus similaires Y_a et Y_b, et regrouper ces deux clusters dans un seul cluster ;
4. Répéter les étapes 2 et 3 jusqu'à ce que le critère d'arrêt ε soit vérifié.

La complexité en temps de calcul de l'algorithme est quadratique ($O(n^2)$) en fonction du nombre d'objets de la base de données car les distances entre toutes paires d'objets doivent être calculées.

Les différentes méthodes de calcul de similarité entre clusters, présentées dans la section 1.3.2, sont utilisées pour choisir les deux clusters à fusionner :

— **Lien simple (*Single Link*) :** Cette méthode, basée sur la mesure du plus proche voisin (Équation 1.1), est simple à implémenter par des algorithmes efficaces (pour plus

de détails voir [60]). Cependant, elle souffre du problème de l'effet de chaîne, c'est-à-dire que des clusters proches mais distincts peuvent être fusionnés s'il existe une chaîne d'objets qui les relient. Pour résoudre ce problème, Guha et al. (1998) ont proposé la méthode CURE [20]. Dans cette méthode, les clusters sont représentés par un ensemble d'objets distants qui les composent, rapprochés de leur centroïde selon un facteur α. Et pour décider quels clusters doivent être fusionnés, elle choisit les clusters dont les objets représentatifs sont les plus proches.

— **Lien complet (*Complete Link*) :** Cette méthode, basée sur la mesure du voisin le plus éloigné (Équation 1.2), est coûteuse en temps de calcul, mais elle fonctionne bien en présence du bruit. Maarek et al. (2000) ont réussi à l'utiliser avec une complexité quadratique [37].

— **Lien moyen de groupe (UPGMA) :** Cette méthode est basée sur la mesure de similarité du moyen de groupe (Équation 1.3).

Une comparaison est faite par Steinbach, Karypis et Kumar (2000) entre trois algorithmes de clustering hiérarchique par agglomération : IST (Intra-Cluster Similarity Technique), CST (Centroid Similarity Technique), et UPGMA (Unweighted Pair Group Method with Arithmatic Mean) montre que l'algorithme UPGMA est la meilleure dans ces trois méthodes [54]. Zhao et Karypis (2002) ont obtenu les mêmes résultats en comparant la méthode UPGMA avec d'autres mesures de similarité entre clusters [65].

Le critère d'arrêt est un concept important pour les algorithmes hiérarchiques. Une fois que la hiérarchie de clusters est formée, pour déterminer le niveau de coupure le plus approprié dans l'arbre, on peut demander à l'utilisateur de spécifier le nombre de clusters attendus et utiliser les différentes mesures de la qualité interne des solutions de clustering possibles pour sélectionner la plus pertinente.

Les méthodes hiérarchiques ne peuvent pas être utilisées de façon incrémentale du

fait que le regroupement est définitif. Ainsi, ces méthodes ne sont pas extensibles pour des ensembles de données de grandes tailles à cause de leur complexité quadratique en temps de calcul.

Algorithmes par partition

Les méthodes de clustering par partition regroupent les données en un nombre fixe de clusters. Tous les clusters sont au même niveau. Les méthodes de clustering les plus connues dans cette catégorie sont :

— **Méthode de passe simple (*Single pass*) :** La méthode de clustering par partition la plus simple est la méthode de *passe simple* [30]. C'est une méthode incrémentale qui consiste à faire un seul passage dans l'ensemble d'objets. Elle se déroule comme suit : le premier objet construit le premier cluster. Puis, les objets sont traités séquentiellement selon l'ordre de leur arrivée. Pour chaque objet, les distances entre cet objet et les centroïdes des clusters existants sont calculées. Si la distance avec le centroïde du cluster le plus proche ne dépasse pas un seuil fixé par l'utilisateur, l'objet est assigné à ce cluster. Sinon, un nouveau cluster est créé à partir de cet objet. L'inconvénient majeur de cette méthode est que le résultat du clustering dépend de l'ordre de traitement des objets. Une solution de ce problème est la méthode K-moyennes et ses variantes.

— **La méthode K-moyennes et ses variantes :** La méthode K-moyennes de base commence par choisir, aléatoirement, un ensemble d'objets formant l'ensemble des centroïdes initiaux des k clusters recherchés. Chaque objet est assigné au cluster dont le centroïde est le plus proche. A chaque itération de l'algorithme, les centroïdes des clusters seront recalculés en fonction des objets qui leur sont associés, et les objets seront réassignés aux clusters selon leur proximité aux nouveaux centroïdes, jusqu'à ce qu'il n'y a plus de changement dans les clusters [38]. Le pseudo code de l'algorithme K-moyennes de base est décrit dans (Algorithme 2).

Algorithme 2 Algorithme de clustering K-moyennes de base

Entrée : un ensemble d'objets $X = \{x_1, x_2, \ldots, x_n\}$, et le nombre de clusters recherchés k.
Sortie : un ensemble de k clusters $\{C_1, C_2, \ldots, C_k\}$, et leurs centroïdes $\{g_1, g_2, \ldots, g_k\}$.
1. Choisir aléatoirement k objets de la base comme des centroïdes initiaux représentant les k clusters recherchés.
2. Assigner chaque objet $x_i (1 \leq i \leq n)$ au cluster C_j dont le centroïde g_j est le plus proche ;
3. Réévaluer les nouveaux centroïdes des clusters en fonction des objets qui leur sont associés ;
4. Répéter les étapes 2 et 3 jusqu'à ce que les centroïdes ne changent plus significativement.

La complexité en temps de calcul de l'algorithme est linéaire de l'ordre $O(n \cdot k \cdot t)$, où n est le nombre d'objets, k le nombre de clusters, et t le nombre d'itérations nécessaires pour la convergence. L'algorithme K-moyennes est donc applicable pour le clustering des grandes bases de données.

L'inconvénient majeur de cette méthode est qu'elle n'est pas capable de découvrir des clusters avec des structures non hypersphériques, et de gérer des clusters proches dont les tailles sont différentes. De plus, le besoin de spécifier le nombre de clusters à priori est aussi un problème. Dans de nombreux cas, l'utilisateur ne possède pas cette information à l'avance. Elle est aussi applicable seulement dans le cas où la moyenne des objets est définie, et par conséquence, elle est sensible aux données bruitées. Une solution de ce dernier problème est l'utilisation des médoïdes au lieu des centroïdes. Cet algorithme est appelé *K-medoids* [30]. Un autre inconvénient de la méthode k-moyennes est qu'elle converge vers une solution optimale locale. Pour trouver une optimalité globale, l'algorithme doit être exécuté plusieurs fois et le meilleur résultat sera retenu.

Nous trouvons l'utilisation de la méthode K-moyennes pour le clustering de documents dans [7, 27]. Quelques variantes de K-moyennes sont aussi utilisées comme *K-medoids* [25, 45] et *global k-means* [33]. Larsen et Aone (1999) ont développé un système pour découvrir la hiérarchie des thèmes dans des grandes collections de documents en utilisant

l'algorithme k-moyennes. Ils ont proposé une méthode appelée (*vector average damping*) qui donne plus de poids aux documents qui viennent de joindre un cluster par rapport à ceux qui sont déjà assignés à ce cluster [32].

Le choix des centroïdes initiaux a une influence sur la partition finale. Dans la méthode k-moyennes de base, ce choix est aléatoire. *Buckshot* et *Fractionation* sont deux méthodes proposées pour déterminer les centroïdes initiaux [15]. La méthode *Buckshot* est une combinaison de la méthode K-moyennes et la méthode HAC (*Hierarchical Agglomerative Clustering*). Dans la méthode *Buckshot*, $\sqrt{n}$ objets sont sélectionnés, aléatoirement, comme un ensemble d'échantillon de toute la collection. Par la suite, la méthode HAC est appliquée sur cet ensemble d'échantillon. Les centroïdes des k clusters dans l'ensemble d'échantillon sont pris comme les centroïdes initiaux pour appliquer la méthode K-moyennes sur toute la collection. La méthode *Fractionation* utilise une sous routine agglomérative pour trouver les centroïdes initiaux. Elle commence par scinder toute la collection en un nombre de groupes (*buckets*) de même taille m ($m > k$). À chacun de ces groupes, une sous routine agglomérative est appliquée pour obtenir ρm clusters, avec $\rho \in (0, 1)$ est le facteur de réduction désiré. Dans l'itération suivante, les documents dans ces clusters sont représentés par leurs centroïdes. Ces centroïdes sont encore divisés en groupes de taille ρm. Ce processus est ensuite appliqué, itérativement, sur la nouvelle partition jusqu'à ce qu'on obtient k clusters. Cette partition finale est utilisée pour générer les centroïdes initiaux pour la méthode K-moyennes.

Les méthodes *Buckshot* et *Fractionation* sont utilisées avec succès dans le système *Scatter/Gather* de clustering de documents [15]. Le système *Scatter/Gather* est un processus interactif et itératif. Il commence par diviser un ensemble de documents retournés en réponse à une requête en un ensemble de clusters (*Scatter*). Ces clusters sont présentés à l'utilisateur. Et à chaque itération, l'utilisateur choisit un ou plusieurs clusters qui lui semblent pertinents (*Gather*). Les clusters choisis sont à nouveau classés pour l'itération suivante.

Steinbach, Karypis et Kumar ont proposé l'algorithme *bisecting K-means* pour générer

une hiérarchie de clusters, en appliquant récursivement la méthode K-moyennes de base [54]. Bisecting K-means est un algorithme de clustering hiérarchique descendant. Initialement, tous les documents de collection forment un seul cluster. Puis l'algorithme sélectionne récursivement un cluster selon des critères donnés (par exemple : le plus grand cluster), et il le divise en deux sous clusters en utilisant l'algorithme K-moyennes de base. L'algorithme s'arrête lorsque le nombre de clusters recherchés est atteint. Le pseudo code de l'algorithme Bisecting K-means est décrit dans (Algorithme 3).

Algorithme 3 Algorithme de clustering Bisecting K-means

Entrée : Un ensemble d'objets $X = \{x_1, x_2, \ldots, x_n\}$, et le nombre de clusters recherchés k.
Sortie : Un ensemble de clusters $\{C_1, C_2, \ldots, C_k\}$.
1. Former un unique cluster regroupant tous les objets ;
2. Choisir un cluster à diviser selon un certain critère ;
3. Trouver deux sous-clusters en utilisant l'algorithme K-moyennes de base (Étape de Bisecting) ;
4. Répéter l'étape 3 ITER fois, et prendre la division qui donne un clustering avec la plus grande valeur de la similarité d'ensemble[13] (*overall similarity*) ;
5. Répéter les étapes 2, 3 et 4 jusqu'à ce que le nombre k de clusters recherchés est atteint.

— **Méthodes basées sur la densité :** Ces méthodes sont basées sur la notion de densité. Un objet est dans une région dense s'il y a un nombre minimum m d'objets dans un voisinage de rayon maximum ε. Les objets connectés par des voisinages denses forment un cluster. La méthode la plus connue dans cette catégorie est la méthode DBSCAN (*Density-based Spatial Clustering of Applications with Noise*)[16]. Sa complexité en temps de calcul est de l'ordre $O(n \log n)$. L'avantage de cette méthode est qu'elle peut découvrir des clusters de formes arbitraires mais sa performance dépend beaucoup du choix des paramètres m et ε. On trouve l'utilisation de la méthode DBSCAN dans le demaine du clustering de documents dans [58, 59].

— **Cartes auto-organisatrices :** La méthode de cartes auto-organisatrices (*Self Organizing Maps SOM*) est une méthode incrémentale qui suit une approche de réseaux de neurones. On trouve l'utilisation de la méthode SOM pour le clustering de documents dans le projet WEBSOM [26]. WEBSOM utilise un réseau de cartes auto-organisatrices pour produire des classes de mots en fonction des co-occurrences sur les voisins immédiats. Ces classes servent à coder les documents, eux mêmes classés par un réseau semblable. Bakus et al. ont aussi utilisé la méthode SOM [4].

Les algorithmes incrémentiels de clustering sont utiles dans le domaine de la recherche sur le Web, où les documents arrivent séquentiellement et ils doivent être traités rapidement. Dans ce contexte, Wong et Fu (2000) ont proposé une structure d'arbre appelée DC-tree (*Document Cluster tree*) pour faciliter le processus de clustering incrémentiel de documents [61]. Cette structure contient des informations condensées sur le cluster. L'idée de DC-tree est similaire à celle de CF-tree (*Clustering Feature*) utilisée dans l'algorithme BIRCH destiné au clustering de grandes bases de données [64].

1.4.2 Algorithmes particuliers de clustering de documents

D'autres algorithmes ont été proposés spécialement pour le clustering de documents.

La méthode d'arbre des suffixes «*Suffix tree method*»

Zamir (1998) a proposé l'utilisation d'un arbre de suffixe (*suffix tree*) pour le clustering de documents [63]. L'arbre de suffixes est une structure de données qui permet de représenter une séquence de caractères s sous forme d'arborescence orienté obtenu par l'insertion de tous les suffixes de s. Cette structure facilite la résolution de beaucoup de problèmes de chaînes de caractères comme la comparaison entre deux chaînes de caractères et l'identification des sous chaînes communes. L'algorithme le plus utilisé pour construire l'arbre des suffixes en temps linéaire est l'algorithme de Ukkonen (1995).

Dans la méthode de clustering d'arbre de suffixes, chaque document est considéré comme une séquence de mots. Tous les documents sont combinés ensemble pour construire l'arbre de suffixes où chaque nœud représente un groupe de documents qui partage un mot commun. Ces groupes de documents sont considérés comme les clusters de base. Tous les clusters de base avec un chevauchement élevé dans leurs ensembles de documents sont combinés ensemble pour générer les partitions finales. L'utilisation de l'arbre de suffixes facilite l'identification des documents qui partagent les mêmes séquences de mots. Cependant, il est difficile d'identifier des clusters de documents qui ont un contenu similaire mais qui ne partagent aucune séquence de mots. Ainsi, deux clusters de base voisins doivent être similaires et deux clusters de base distants ne peuvent pas être similaires.

Algorithmes basés sur les itemsets fréquents

Beil, Ester, et Xu (2002) ont proposé deux méthodes de clustering, FTC (Frequent Term-based Clustering) et HFTC (Hierarchical Frequent Term-based Clustering) basées sur les ensembles de termes fréquents [6]. L'idée de termes fréquents est inspirée du problème d'extraction des itemsets fréquents dans des bases de transactions. Chaque document est considéré comme une transaction, et chaque terme comme un item. La méthode FTC génère une partition unique où les clusters sont au même niveau. La première étape de FTC consiste à extraire tous les ensembles de termes fréquents dans toute la collection de documents en utilisant l'algorithme Apriori [1]. Ensuite, elle utilise un algorithme glouton pour sélectionner les ensembles de termes fréquents qui couvrent tous les documents avec un chevauchement minimal entre leurs couvertures[14]. Pour chaque ensemble de termes fréquents, un cluster est créé, et les documents contenant cet ensemble de termes fréquents sont assignés au cluster créé. L'algorithme HFTC est basé sur l'algorithme FTC afin de créer une hiérarchie de clusters. L'inconvénient majeur de ces deux méthodes est que le résultat de clustering est affecté par l'ordre de sélection

[14]la couverture d'un ensemble de termes fréquents est l'ensemble de documents contenant cet ensemble de termes

de l'ensemble suivant de termes fréquents.

Fung (2003) a proposé une autre méthode hiérarchique de clustering basé sur l'ensemble de termes fréquents, HIFC (Frequent Itemset-based Hierarchical Clustering) afin d'améliorer la méthode HFTC [18]. L'algorithme HIFC construit un arbre de clusters pour tous les ensembles de termes fréquents. Puis, un algorithme d'élagage (*pruning*) est utilisé pour fusionner les clusters les plus similaires.

Autres algorithmes

Weiss et al. (2000) ont décrit une méthode de clustering de documents «*lightweight*», en utilisant la technique des plus proches voisins [57]. Lin et Ravikumar (2001) ont décrit un soft système de clustering de documents, nommé WBSC (*Word-based Soft Clustering*)[35]. Hammouda (2002) a proposé un algorithme de clustering incrémentiel, en représentant chaque cluster par un histogramme de similarité [22].

1.5 Synthèse

Dans ce chapitre, nous avons présenté l'état de l'art du clustering de documents, à savoir la représentation des documents, les mesures de similarité et les algorithmes de clustering.

Les approches actuelles de représentation des documents présentent quelques limites. La représentation sac de mots met à l'écart une quantité considérable d'informations sur le contenu de document. Les phrases, l'ordre des mots et les relations syntaxiques et sémantique entre les mots sont ignorés. La représentation basée sur les séquences de mots statistiques permet de déterminer toutes les combinaisons possibles de mots en se basant sur leur fréquence d'apparition dans la collection. Cependant, certaines combinaisons extraites n'ont pas une signification syntaxique ou sémantique et par conséquence

elles ne sont pas pertinentes pour représenter le contenu des documents. Pour pallier à ce problème, des séquences de mots liés syntaxiquement ont été utilisées. Cette représentation se base sur des outils linguistiques complexes comme l'analyseur syntaxique. La performance de ces méthodes dépend de la taille de la collection. En effet, malgré le fait que ces termes sont sémantiquement riches pour représenter le contenu des documents, ils ont une faible fréquence d'apparition dans la collection. Une exploitation efficace de ces termes nécessite une fonction de pondération permettant de les accorder plus d'importance.

Dans ce travail, nous nous intéressons à un autre type d'information syntaxique : les relations de dépendance. L'analyseur syntaxique permet de formuler la structure de la phrase sous forme d'arbre où les mots sont liés par des relations de dépendance. Notre motivation principale pour exploiter cette structure d'arbre vient du fait que plusieurs variantes syntaxiques peuvent être ramenées aux mêmes relations de dépendance. Par exemple, les groupes syntaxiques : «*document clustering*», «*clustering of documents*» et «*clustering of textuel documents*» partagent la même structure de dépendance «*clustering-document*». On peut donc considérer que les groupes de mots avec des relations de dépendance offrent une forme généralisée des groupes syntaxiques.

En général, les groupes de mots refèrent à un domaine de connaissance plus spécialisé. Dans certains cas, il est utile de combiner les mots simples avec les groupes de mots dans la représentation des documents.

Un autre point important pour le clustering de documents est le choix de la mesure de similarité. Strehl et al. (2000) [55] ont effectué une comparaison entre les mesures de similarité : la distance euclidienne, le cosinus, coefficient de Jaccard et la corrélation de Pearson. Ils ont montré que le cosinus et le coefficient de Jaccard sont les plus appropriés pour le clustering de documents. Les mesures de similarité cosinus et Jaccard ont été utilisées seulement pour les mots simples. Elles ne prennent pas en compte la proximité des mots (relations sémantiques) et les groupes de mots.

En effet, les mesures de similarité dépendent du modèle de représentation adopté. Par exemple, Chim et Deng ont proposé une mesure de similarité entre documents en passant du modèle d'arbre de suffixes au modèle vectoriel où chaque nœud dans l'arbre de suffixes est devenu une dimension dans l'espace vectoriel. Le poids des termes est calculé selon une formule *Tf-Idf* et la mesure de similarité utilisée est le cosinus [12]. Hammouda et Kamel ont proposé une mesure de similarité entre documents hybride qui combine la mesure de similarité cosinus pour les mots simples et une autre mesure de similarité proposée pour les groupes de mots communs entre les documents dans le modèle de graphe d'index de document (DIG) [24]. Dans ce travail, nous proposons une mesure de similarité pour notre approche de représentation.

Nous avons présenté aussi les algorithmes de clustering de documents existants. Chaque algorithme possède des avantages et des inconvénients. Le choix d'un algorithme est intimement lié à la collection de documents sur laquelle il est appliqué. Dans ce travail, nous avons choisi d'utiliser l'algorithme hiérarchique par agglomération.

Chapitre 2

Présentation de notre système de clustering de documents

2.1 Introduction

Dans notre système de clustering de documents, nous combinons les techniques de traitement de la langue naturelle et les techniques de fouille d'arbres pour représenter les documents. La fouille d'arbres est le problème de la recherche de sous-arbres fréquents dans une base de données d'arbres. Elle trouve de nombreuses applications, notamment dans les domaines de bio-informatique (structures RNA), la classification de documents semi-structurés (documents XML) et le Web (analyse du comportement des internautes).

Pour une collection de textes, nous construisons une base de données d'arbres de dépendance, où chaque arbre représente une phrase dans la collection. Une fois cette base construite, nous appliquons un algorithme de fouille d'arbres pour découvrir tous les sous-arbres fréquents qui permettront par la suite de représenter les documents. Ainsi, nous introduisons une nouvelle mesure de similarité entre documents basée sur cette représentation. L'algorithme hiérarchique par agglomération est utilisé comme algorithme

de clustering.

Dans ce chapitre, nous décrivons notre système de clustering de documents. Nous commençons par la définition de quelques notions de base de fouille d'arbres dans la section 2.2. Ensuite, une vue générale du système est présentée dans la section 2.3. Dans les sections 2.4, 2.5 et 2.6, nous donnons des détails sur chaque unité de notre système.

2.2 Notions de base : rappel et définitions

Un arbre libre est un graphe non orienté, connexe et sans cycle. Un arbre enraciné est un arbre libre dans lequel un nœud n'a aucune arête entrante, appelé la racine, les autres nœuds ont exactement une seule arête entrante, et il n'existe qu'un seul chemin de la racine vers n'importe quel nœud. Il est dit étiqueté si on associe à chaque nœud une étiquette. Il s'agit d'un arbre ordonné s'il existe une relation d'ordre entre les frères et d'un arbre non ordonné sinon.

Définition 2.1 (Arbre enraciné, étiqueté et ordonné) *Soit Σ un ensemble d'étiquettes pour les nœuds. Un arbre enraciné, étiqueté et ordonné est la structure $T = (V, E, \Sigma, L, r, \prec)$ où V est un ensemble fini de nœuds, E est un ensemble fini d'arêtes, $L : V \rightarrow \Sigma$ est une fonction d'assignation d'étiquettes pour les nœuds (pour chaque nœud $v \in V$), le nœud r ($r \in V$) est la racine de T, et $\prec$ est une relation binaire ($\prec \subseteq V^2$) qui représente la relation d'ordre entre les paires de fils de chaque nœud.*

Si $(u, v) \in E$ alors on dit que u est le père de v et v est le fils de u. On dit que le nœud s est un grand frère du nœud t si et seulement si s et t sont les fils du même père et $s \prec t$. Chaque nœud sur le chemin unique allant de la racine r à v est appelé un ancêtre de v. Si u est un ancêtre de v, alors v est un descendant de u. Une feuille est un nœud qui ne possède pas de fils. La taille de T, notée $\mid T \mid$, est le nombre de nœuds de T.

Exemple 2.1 *Pour l'arbre $T = (V, E, \Sigma, L, r, \prec)$ dans l'exemple de la figure 2.1, on a :*

$V = \{n0, n1, n2, n3, n4, n5\}$;

$E = \{(n0, n1), (n0, n2), (n0, n5), (n2, n3), (n2, n4)\}$;

$\Sigma = \{A, B, C, D, E, F\}$;

$L(n0) = A, L(n1) = B, L(n2) = C, L(n3) = E, L(n4) = F, L(n5) = D$;

$r = n0$;

$\prec$: *Ordre alphabétique des étiquettes.*

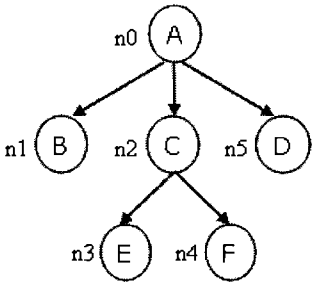


Figure 2.1: Exemple d'arbre enraciné, étiqueté et ordonné.

Dans le but de comparer deux arbres, différents types de sous-arbres peuvent être définis. Le cas le plus spécifique est un sous-arbre de racine r où r et tous ses descendants doivent apparaître dans les deux arbres (sous-arbre *bottom-up*). Cette restriction peut être allégée en permettant de supprimer quelque fils (sous-arbre induit), ou en conservant la relation ancêtre-descendant même si quelques autres descendants sont manquants (sous-arbre incrusté «*embedded*»). La figure 2.2 montre des exemples de ces trois types de sous-arbres. Pour l'arbre T, l'arbre S_1 est un sous-arbre *bottom-up*, l'arbre S_2 est un sous-arbre induit mais il n'est pas un sous-arbre *bottom-up*, et l'arbre S_3 est un sous-arbre incrusté mais il n'est ni un sous-arbre *bottom-up* ni un sous-arbre induit.

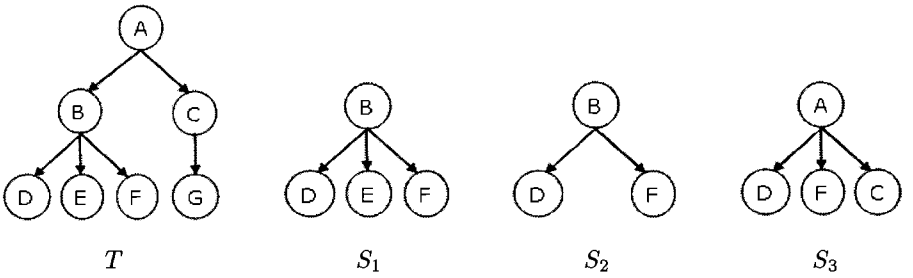


Figure 2.2: Exemples de types de sous-arbres.

Définition 2.2 (Sous-arbre induit et ordonné) *Soient deux arbres* $S = (V_S, E_S, L_S, r_S, \prec_S)$ *et* $T = (V_T, E_T, L_T, r_T, \prec_T)$. *On dit que* S *est un sous-arbre induit de* T, *noté par* $S \sqsubseteq T$, *si et seulement s'il existe une fonction injective* $g : V_S \rightarrow V_T$ *des nœuds de* S *aux nœuds de* T, *qui vérifie les conditions suivantes :*

1. g *préserve les étiquettes :* $\forall x \in V_S, L_S(x) = L_T(g(x))$
2. g *préserve la relation d'ordre entre frères :* $\forall x, y \in V_S$, *si* $x \prec_S y$ *alors* $g(x) \prec_T g(y)$
3. g *préserve la relation de parenté,* $\forall x, y \in V_S$, *si* $(x, y) \in E_S$ *alors* $(g(x), g(y)) \in E_T$

Définition 2.3 (Support) *Soit* $\mathcal{D}$ *une base de données d'arbres où chaque arbre* $T \in \mathcal{D}$ *est un arbre enraciné, étiqueté et ordonné. Pour un arbre motif* P, *qui est aussi un arbre enraciné, étiqueté et ordonné, on dit que* P *est inclus dans* T *si* P *est un sous-arbre de* T.

Soit $\phi_T(P)$ *le nombre d'occurrences de* P *dans* T. *On définit* $\psi_T(P)$ *comme suit :*

$$\psi_T(P) = \begin{cases} 1 & si\ \phi_T(P) > 0 \\ 0 & sinon \end{cases}$$

Le support du sous-arbre P *dans la base de données* $\mathcal{D}$ *est définit comme la proportion*

des arbres de D qui contiennent au moins un exemplaire de P.

$$support(P, \mathcal{D}) = \frac{\sum_{T \in D} \psi_T(P)}{\mid \mathcal{D} \mid}$$

Où $\mid \mathcal{D} \mid$ est le nombre d'arbres dans $\mathcal{D}$.

Un arbre motif P est dit fréquent si le support de P est supérieur ou égal à un support minimum défini par l'utilisateur (*min_sup*).

Le but de la fouille d'arbres fréquents est de trouver tous les arbres fréquents dans la base de données d'arbres. Les algorithmes de fouille d'arbres utilisent la propriété *a priori* suivante : chaque sous-arbre d'un arbre fréquent est aussi fréquent, et chaque super-arbre d'un arbre non fréquent est aussi non fréquent.

2.3 Vue générale du système

Notre système de clustering de documents est constitué de deux composantes principales : représentation de documents et clustering. Chacune de ces composantes est divisée en unités, et chacune des unités a une fonction définies dans les sections qui suivent. L'architecture globale du système est montré dans la figure 2.3.

Pour la représentation des documents, le système commence par une étape de prétraitement des textes qui consiste à découper chaque document dans la collection à un ensemble de phrases qui le constituent.

Étant donnée une collection D de n documents textes : $D = \{d_1, d_2, \cdots, d_i, \cdots, d_n\}$, où d_i est le i-ème document.

En utilisant le segmenteur MXTERMINATOR, chaque document est découpé en phrases :

$$d_i = \{s_1^i, s_2^i, \cdots, s_{m(i)}^i\}$$

Où $m(i)$ est le nombre de phrases dans le i-ème document.

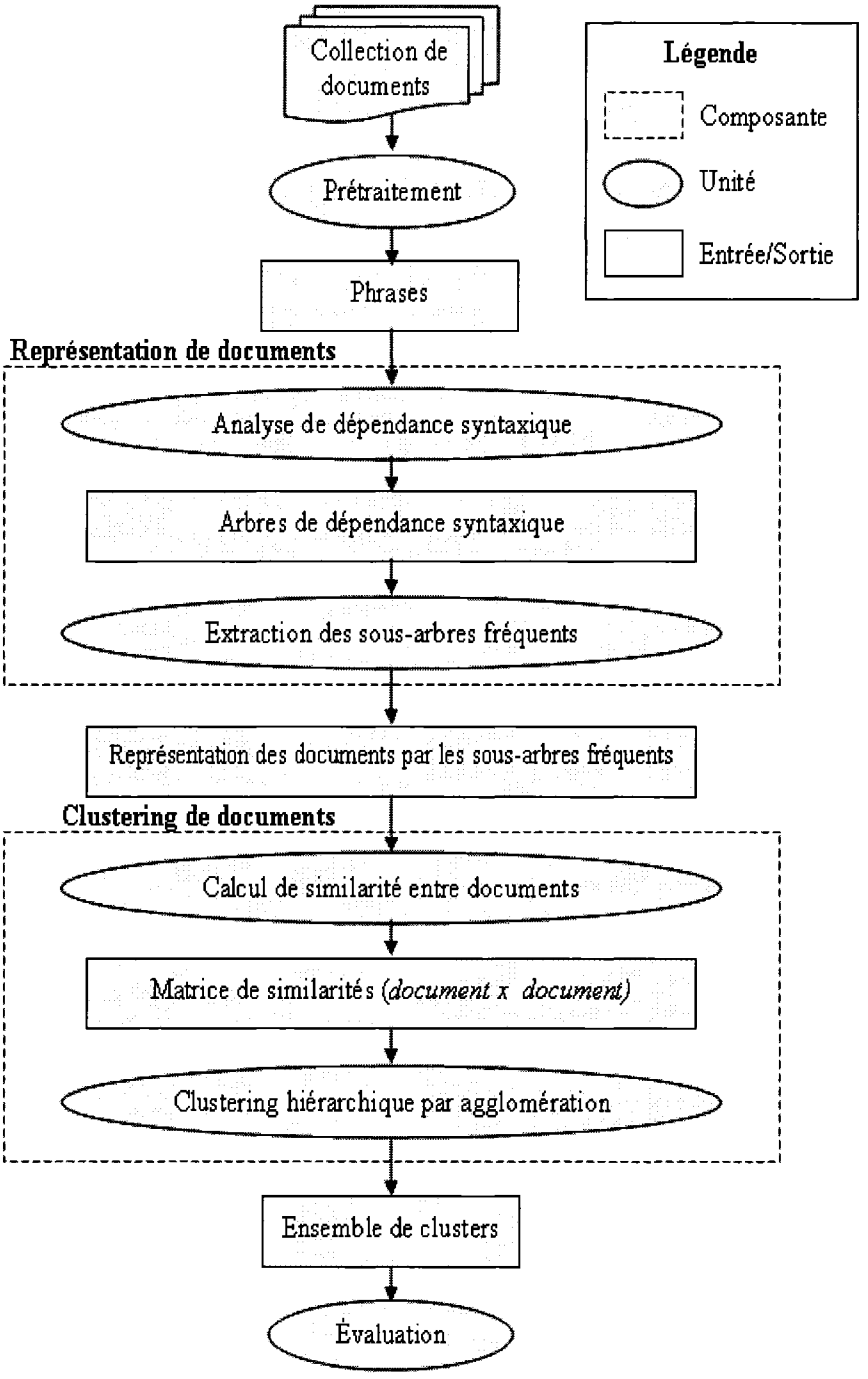


Figure 2.3: Architecture globale du système.

Ensuite, l'unité de dépendance syntaxique construit l'arbre de dépendance syntaxique pour chaque phrase. Nous obtenons en sortie de cette unité un ensemble d'arbres de dépendance syntaxique. Les sous-arbres fréquents sont extraits en appliquant un algorithme de fouille d'arbres et un filtrage pour enlever les sous-arbres redondants. Ces sous-arbres fréquents vont être utilisés pour représenter les documents.

Pour le clustering, nous construisons une matrice de similarités *document* × *document*, en nous basant sur une nouvelle mesure de similarité qui sera présentée en détail dans la section 2.6. Le système utilise l'algorithme hiérarchique par agglomération comme algorithme de clustering. La dernière étape consiste à l'évaluation de notre système.

2.4 Analyse de dépendance syntaxique

Le but de cette unité est de construire les arbres de dépendance syntaxique. Pour chaque phrase, une analyse morphologique et une analyse syntaxique sont effectuées. L'analyse syntaxique achève la formulation de la structure de la phrase en construisant un arbre de dépendance.

L'arbre de dépendance représente les liens existants entre les mots au niveau de la phrase :

1. Chaque nœud d'un arbre correspond à un et un seul mot de la phrase ;
2. La relation d'ordre entre les nœuds frères correspond à l'ordre d'apparition des mots dans la phrase ;
3. La direction des arêtes correspond à la direction de dépendance syntaxique ;
4. L'étiquette de chaque nœud est la forme de base (*lemme*) de chaque mot.

Pour réaliser cette tâche, nous avons utilisé «*Stanford Parser*», un analyseur syntaxique de la langue anglaise qui permet de générer l'arbre de dépendance syntaxique d'une phrase. La sortie de cet analyseur présente les relations de dépendance entre les

mots de la phrase. Le principe est que chaque mot de la phrase est en relation de dépendance avec d'autres mots. Par exemple, considérons la phrase «The documents are correctly classified.». Le format de relations de dépendance en sortie du *Stanford parser* pour cette phrase est :

det(document-2, the-1)

nsubjpass(classify-5, document-2)

auxpass(classify-5, be-3)

advmod(classify-5, correctly-4)

Ces relations de dépendance ont la signification suivante :

- det (*determiner*) : relation de dépendance déterminant-nom ;
- nsubjpass (*passive nominal subject*) : relation de dépendance sujet-verbe ;
- auxpass (*passive auxiliary*) : relation de dépendance auxiliaire-verbe ;
- advmod (*adverbial modifier*) : relation de dépendance adverbe-verbe.

L'arbre de dépendance syntaxique de cette phrase est montré dans la figure 2.4.

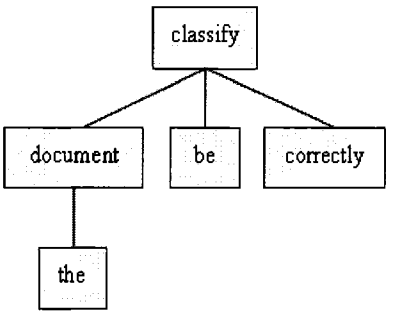


Figure 2.4: L'arbre de dépendance syntaxique de la phrase : *The documents are correctly classified.*

Soit $AD(s)$ l'arbre de dépendance généré par cette unité à partir de la phrase s, et soit T_j^i l'arbre de dépendance qui correspond à la j-ème phrase du i-ème document s_j^i, tel que $T_j^i = AD(s_j^i)$. Soit l'ensemble $\hat{d}^i = \{T_1^i, \ldots, T_j^i, \ldots, T_{m(i)}^i\}$ le résultat de l'analyse

syntaxique du i-ème document, où $m(i)$ est le nombre de phrases dans le i-ème document. Les arbres de dépendance sont des arbres enracinés, étiquetés et ordonnés. En sortie de cette unité, nous obtenons une base de données d'arbres enracinés, étiquetés et ordonnés $\mathcal{D} = \{\hat{d}^1, \ldots, \hat{d}^i, \ldots, \hat{d}^n\}$. Cette base de données d'arbres sera l'entrée de l'unité suivante.

2.5 Extraction des sous-arbres fréquents

Le but de cette unité est de découvrir les sous-arbres caractéristiques dans la base de données d'arbres $\mathcal{D}$. Afin de conserver les relations de dépendance entre les nœuds, les sous-arbres caractéristiques doivent être des sous-arbres induits et ordonnés. Pour réaliser cette tâche, nous avons utilisé l'algorithme FREQT (*FREQuent Tree miner*), mais nous aurions pu utiliser n'importe quel algorithme de recherche de sous-arbres induits dans une base de données d'arbres ordonnés.

2.5.1 L'algorithme FREQT :

L'algorithme FREQT est un algorithme développé par Asai et al. en 2002 [3] pour trouver tous les sous-arbres induits fréquents dans une base de données d'arbres enracinés, étiquetés et ordonnés. Il se déroule comme suit :

1. Recherche des étiquettes fréquentes dans la base de données.
2. Construction de l'arbre d'énumération de tous les sous-arbres fréquents en utilisant la méthode *rightmost expansion* [3, 62]. Dans cette méthode, l'ensemble des sous-arbres candidats de taille (k) (où k est le nombre de nœuds de l'arbre) sont générés par l'extension des sous-arbres fréquents de taille $(k-1)$ en attachant un nouveau nœud avec une étiquette fréquente à la branche la plus à droite de chaque arbre, où la branche la plus à droite d'un arbre est le chemin unique de la racine à la feuille la plus à droite. Cette méthode assure que chaque sous-arbre candidat soit généré

exactement une seule fois.

3. Calcul du support basé sur une approche de liste d'occurrence. Pour chaque sous-arbre, une liste d'occurrence est associée. Les éléments de cette liste sont tous les nœuds dans la base de données d'arbres qui ont un *mapping* avec le nœud le plus à droite de ce sous-arbre. L'algorithme 4 est un pseudo-code de l'algorithme FREQT.

Algorithme 4 Algorithme FREQT.

Entrées: $\mathcal{D}$: une base de données d'arbres,
min_sup : un support minimum ($0 < min\_sup \leq 1$).
Sorties: F : l'ensemble de tous les sous-arbres fréquents dans $\mathcal{D}$
$k \leftarrow 1$;
$C_1 \leftarrow$ l'ensemble d'arbres candidats de taille 1 ;
$F_1 \leftarrow$ l'ensemble d'arbres fréquents de taille 1 ;
$C_1 \leftarrow F_1$;
$RMO_1 \leftarrow$ les occurrences les plus à droite des arbres dans F_1 ;
tantque $F_k \neq \emptyset$ **faire**
  $k \leftarrow k + 1$;
  $C_k, RMO_k \leftarrow$ génération de l'ensemble d'arbres candidats de taille k (C_k) et leurs occurrences (RMO_k) à partir de (F_{k-1}, RMO_{k-1}) ;
  $F_k \leftarrow \emptyset$
  pour chaque arbre candidat $C \in C_k$ **faire**
    Calcul du support de C ($support(C, \mathcal{D})$) à partir de $RMO_k(C)$
    si $support(C, \mathcal{D}) \geq min\_sup$ **alors**
      $F_k \leftarrow F_k \cup \{C\}$
    finsi
  fin pour
fin tantque
Retourner $F = F_1 \cup F_2 \cup \ldots \cup F_{k-1}$.

Exemple : La figure 2.5 montre un exemple d'une base de données de trois arbres enracinés, étiquetés et ordonnés. On fixe le support minimum à $\frac{2}{3}$ ($minsup = \frac{2}{3}$), les sous-arbres p_1, p_2, p_3 et p_4 de la figure 2.6 sont fréquents. Par exemple, le sous-arbre p_3 apparaît deux fois dans t_1 et t_2, et une seule fois dans t_3, donc son support est égal à $\frac{3}{3}$ qui est supérieur au support minimum. La figure 2.7 montre l'arbre d'énumération pour tous les sous-arbres candidats se composant des étiquettes fréquentes A et B.

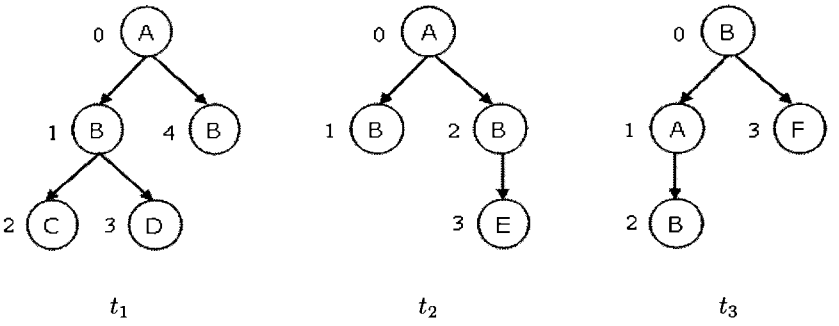


Figure 2.5: Exemple d'une base de données d'arbres.

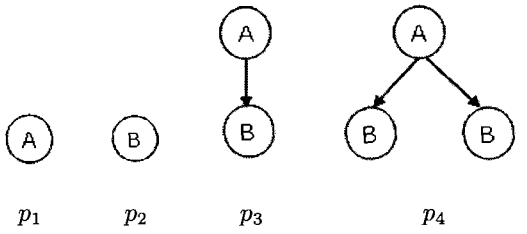


Figure 2.6: Exemples de sous-arbres fréquents.

Dans l'exemple de la figure 2.8, la liste d'occurrences pour l'arbre p_1 est $\{(t_1, 0), (t_2, 0)$-$, (t_3, 1)\}$, qui signifie que p_1 se produit trois fois : une fois dans l'arbre t_1 comme le nœud numéro 0, une fois dans t_2 comme le nœud numéro 0, et une fois dans t_3 comme le nœud numéro 1. Une extension qui est fréquente dans cette base de données est p_3. La liste d'occurrences de p_3 est $\{(t_1, 1), (t_1, 4), (t_2, 1), (t_2, 2), (t_3, 2)\}$; ici $(t_1, 1)$ dénote que le nœud le plus à droite de p_3 peut être tracé au nœud numéro 1 de l'arbre t_1. Seulement les occurrences des nœuds sur la branche la plus à droite, c'est-à-dire le chemin unique allant de la racine vers la feuille la plus à droite de l'arbre, sont nécessaires pour une nouvelle extension, ces occurrences peuvent être dérivées de l'occurrence du nœud le plus à droite. Par exemple pour l'arbre p_4, on doit seulement enregistrer les occurrences du nœud le plus à droite B, donc la liste d'occurrences de p_4 est $\{(t_1, 4), (t_2, 2)\}$. Pour avoir

la liste d'occurrences du nœud A, il suffit de trouver les parents des nœuds dans la liste d'occurrences du nœud le plus à droite B, qui donne $\{(t_1, 0), (t_2, 0)\}$ pour notre exemple.

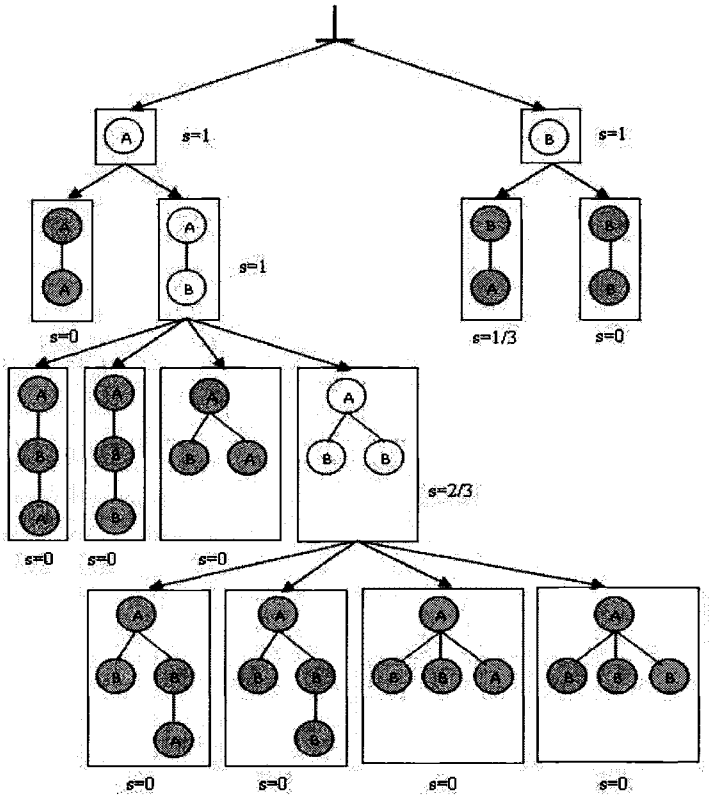


Figure 2.7: L'arbre d'énumération.

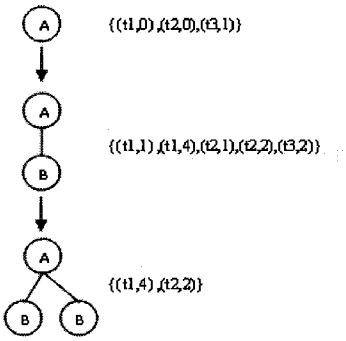


Figure 2.8: La liste d'occurrences.

La complexité de l'algorithme FREQT est de l'ordre $O(fm|V_D|)$, Où $|V_D|$ est le nombre de nœuds dans la base de données, f est le nombre d'arbres fréquents dans la base de données, et m est le nombre des nœuds du plus grand arbre dans la base de données.

2.5.2 Adaptation de l'algorithme FREQT pour les arbres de dépendance

Afin de pouvoir utiliser l'algorithme FREQT sur notre base de données d'arbres de dépendance, nous avons besoin d'adapter l'algorithme pour notre problème de représentation des documents. Les changements suivants sont nécessaires :

Suppression des mots vides

Les mots vides sont des mots non significatifs, comme les articles, les prépositions, les pronoms, ...etc. Il est généralement admis que ces mots très fréquents sont à supprimer car ils ne sont pas informatifs, et ils augmentent énormément la taille de l'espace de représentation des documents. Les mots vides sont alors souvent regroupés dans une liste

de *stopwords*.

Dans l'arbre de dépendance syntaxique, les mots vides sont, en général, des feuilles. Leur suppression n'influe pas sur la structure de l'arbre. Nous avons éliminé ces mots dans la première étape de l'algorithme lors de la sélection des étiquettes fréquentes en utilisant une liste de 571 *stopwords* de la langue anglaise développée par *SMART project*[1].)

Taille de sous-arbres fréquents

Dans la littérature, beaucoup de travaux ont suggéré d'utiliser des structures de taille entre un et trois mots pour représenter les documents [11, 40, 41, 52]. Pour avoir plus de flexibilité dans notre représentation, nous avons défini deux paramètres : taille minimale min_motif et taille maximale max_motif de sous-arbres fréquents. Ces deux paramètres nous permettent de limiter la recherche de sous-arbres fréquents à des structures de taille comprise entre min_motif et max_motif.

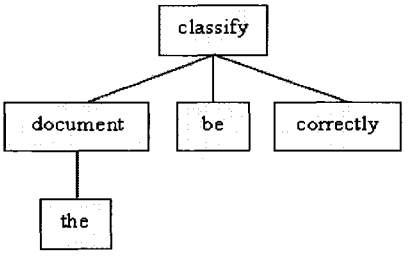
Regroupement des sous-arbres équivalents

L'utilisation des arbres ordonnés pose le problème suivant : les arbres qui doivent être comptés ensemble peuvent être des différents arbres ordonnés. En fait, dans l'arbre de dépendance syntaxique, la différence dans l'ordre de dépendance est représentée par la différence dans l'ordre des nœuds frères, alors que la différence dans la direction de dépendance est représentée par la différence de la direction des arêtes.

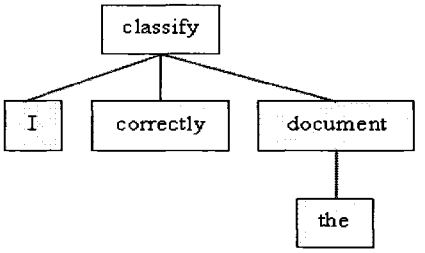
La figure 2.9 illustre le problème que nous visons à résoudre. Dans l'exemple, les trois arbres (a), (b) et (c) représentent les trois phrases «*The documents are correctly classified*», «*I classified correctly the documents*» et «*I have the documents correctly classified*», respectivement. Ces phrases ont le même sens mais l'ordre des nœuds dans chaque arbre est différent. L'arbre (d) qui représente «*classify documents correctly*» est

[1]SMART Project : une liste de stopwords de la langue anglaise pour la recherche d'information, http ://www.unine.ch/info/clef/englishST.txt

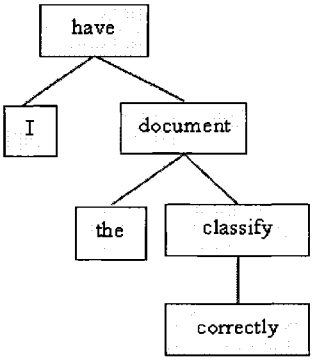
un sous arbre ordonné de (a), mais il n'est pas un sous arbre ordonné de (b) car l'ordre des frères est différent, ni de (c) car la direction des arêtes est différente. La sous-structure «*classify document correctly*» n'est pas considérée comme une structure commune entre les trois phrases.



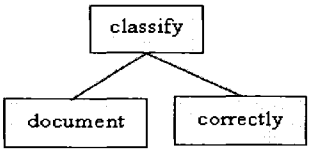
(a) The documents are correctly classified.



(b) I classified correctly the documents.



(c) I have the documents correctly classified.



(d)...classify document correctly ...

Figure 2.9: Exemples des (sous) arbres de dépendance syntaxique.

Dans la littérature, la méthode utilisée pour résoudre ce problème est basée sur une transformation structurelle des arbres de dépendance [42]. Deux transformations structurelles sont effectuées successivement :

1. La première transformation consiste à réordonner les nœuds frères par ordre alphabétique de leurs étiquettes. Dans ce sens, les arbres ordonnés peuvent être normalisés, en prenant en compte l'ordre alphabétique des nœuds frères au lieu de l'ordre de dépendance. Dans le cas où l'arbre de dépendance contient des nœuds frères qui ont la même étiquette, on construit un nouvel arbre pour chaque permutation de ces nœuds frères. Dans la figure 2.10, les arbres (a) et (b) sont devenus le même arbre en utilisant cette transformation.

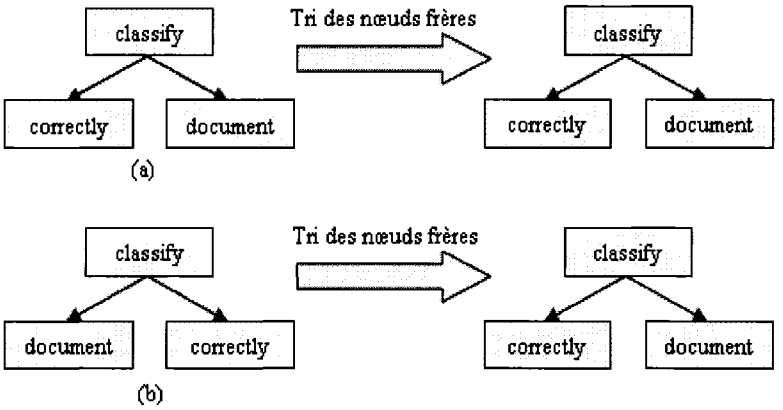


Figure 2.10: Exemple du tri des nœuds frères.

2. La deuxième transformation consiste à faire une expansion de la racine. Pour chaque nœud de l'arbre de dépendance original, un nouvel arbre est construit dont ce nœud est la racine. Cette méthode permet de résoudre le problème causé par la différence dans les directions des arêtes. La figure 2.11 illustre cette transformation.

Les deux transformations permettent d'énumérer tous les sous-arbres quels que soient l'ordre des nœuds frères et la direction de dépendance. Mais, cette solution est très coûteuse en temps de calcul car elle augmente considérablement le nombre d'arbres dans la base de données. En plus des arbres de dépendance avec différents sens peuvent devenir le même arbre en utilisant ces deux transformations.

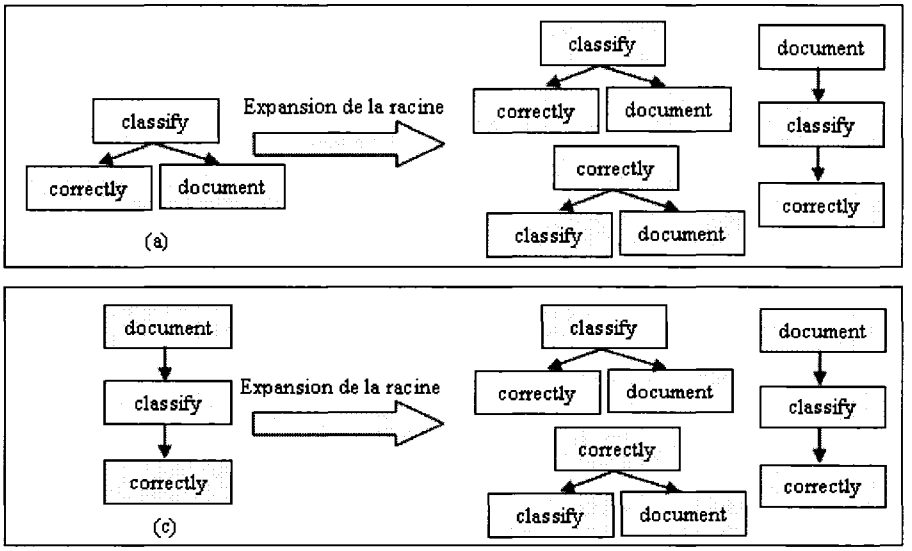


Figure 2.11: Exemples de l'expansion de la racine.

Nous proposons une autre solution, moins coûteuse en temps de calcul par rapport à celle présentée ci-dessus, qui consiste en deux étapes :

1. La normalisation des arbres de dépendance, en prenant en compte l'ordre alphabétique des nœuds frères au lieu de l'ordre de dépendance.
2. Le calcul du support des sous-arbres candidats : le support de chaque sous-arbre candidat est la somme des supports de tous les sous-arbres candidats qui lui sont équivalents comme des arbres non ordonnés et libres. L'appariement entre ces sous-arbres candidats est fait de manière à ignorer l'ordre entre frères en les considérant comme des arbres non ordonnés, et à ignorer aussi la direction des arêtes en les considérant comme des arbres libres.

En sortie de l'algorithme, nous obtenons l'ensemble de sous-arbres fréquents dans $\mathcal{D}$ avec les informations suivantes pour chaque sous-arbre ST_i :

- Son support *sup* dans $\mathcal{D}$;

- La liste d'arbres où il apparaît avec sa fréquence d'apparition dans ces arbres $L(ST_i) = \{(T_1, f_{1i}), \ldots, (T_n, f_{ni})\}$.

Le pseudo-code de l'algorithme FREQT adapté, où les changements ci-dessus sont apportés, est décrit dans (Algorithme 5).

2.5.3 Filtrage

Beaucoup de sous-arbres fréquents extraits sont redondants. Pour enlever cette redondance, nous avons procédé par deux types de filtrages :

1. **Regroupement des sous-arbres équivalents :** Nous avons regroupé les sous-arbres équivalents à d'autres sous-arbres comme arbres non ordonnés ou libres.
2. **Suppression des sous-arbres inclus dans d'autres sous-arbres :** Les sous-arbres inclus dans d'autres sous-arbres comme arbres non ordonnés ou libres doivent être éliminés, en se basant sur leur fréquence d'apparition dans leur arbres d'origine. Par exemple, considérons l'arbre de dépendance T de la phrase «*Document clustering is the act of collecting similar documents into groups*», et supposons que les trois sous-arbres ST_1, ST_2 et ST_3 sont fréquents, tels que : ST_1 est constitué d'un seul nœud «*document*», ST_2 est constitué d'un seul nœud «*clustering*» et ST_3 est constitué de deux nœuds «*document*» et «*clustering*». Les fréquences d'apparition de ST_1, ST_2 et ST_3 dans T sont 2, 1 et 1, respectivement. Mais ST_1 et ST_2 sont inclus dans ST_3, nous soustrayons donc de leur fréquence d'apparition la fréquence de ST_3. Par conséquent, nous éliminons ST_2 pour T et nous gardons ST_1 et ST_3 avec une fréquence de 1.

Algorithme 5 Algorithme FREQT Adatpé.

Entrées: $\mathcal{D}$: base de données d'arbres de dépendance,
min_sup : un support minimum ($0 < min\_sup \leq 1$),
min_motif : taille minimale de sous-arbres,
max_motif : taille maximale de sous arbre.
Sorties: F : l'ensemble de tous les sous-arbres fréquents dans $\mathcal{D}$.

// Ordonner chaque arbre dans la base de données par ordre alphabétique.
pour chaque arbre $T \in \mathcal{D}$ **faire**
ordonner les nœuds frères de l'arbre par ordre alphabétique.
fin pour
$k \leftarrow 1$;
$C_1 \leftarrow$ l'ensemble d'arbres candidats de taille 1 ;
$F_1 \leftarrow$ l'ensemble d'arbres fréquents de taille 1 ;
$C_1 \leftarrow F_1$;
$RMO_1 \leftarrow$ les occurrences les plus à droite des arbres dans F_1 ;
// Suppression des mots vides parmi les étiquettes fréquentes.
pour chaque sous-arbre $ST \in F_1$ **faire**
si $stopword(etiquette(ST))$ **alors**
$F_1 = F_1 - \{ST\}$
finsi
fin pour
tantque $F_k \neq \emptyset$ **faire**
$k \leftarrow k + 1$;
$C_k, RMO_k \leftarrow$ génération de l'ensemble d'arbres candidats de taille k (C_k) et leurs occurrences (RMO_k) à partir de (F_{k-1}, RMO_{k-1}) ;
$F_k \leftarrow \emptyset$
pour chaque candidat $T \in C_k$ **faire**
$support(T) = support(T, \mathcal{D})$ *// Calcul du support de T à partir de $RMO_k(T)$*
pour chaque candidat $U \in C_k$ et $U \neq T$ **faire**
si U est un arbre equivalent à T **alors**
$support(T) = support(T) + support(U, \mathcal{D})$
finsi
fin pour
si $support(T) \geq min\_sup$ **alors**
$F_k \leftarrow F_k \cup \{T\}$
finsi
fin pour
fin tantque
Retourner $F = F_1 \cup F_2 \cup \ldots \cup F_{k-1}$.

2.5.4 Modèle de représentation

Nous décrivons ici notre modèle de représentation. Dans ce modèle, un document est représenté par l'ensemble des arbres de dépendance des phrases qui le constituent dont chaque arbre est représenté, à son tour, par l'ensemble des sous-arbres fréquents extraits :

$$\hat{d^i} = \{T^i_1, \ldots, T^i_j, \ldots, T^i_{m(i)}\} \tag{2.1}$$

$$T^i_j = \{(ST^i_{j1}, f^i_{j1}), \ldots, (ST^i_{jk}, f^i_{jk}), \ldots, (ST^i_{m(i)n(j)}, f^i_{m(i)n(j)})\} \tag{2.2}$$

Où :

$\hat{d^i}$: est le document i,

T^i_j : est l'arbre de dépendance de la phrase j du document i,

$m(i)$: est le nombre de phrases dans le document i,

ST^i_{jk} : est le sous-arbre fréquent k de l'arbre T^i_j, et

f^i_{jk} : est la fréquence d'apparition du sous-arbre ST^i_{jk} dans l'arbre T^i_j.

Cette représentation des documents permet de maintenir la structure de la phrase dans le document original qui va être utilisée dans le calcul de similarité entre documents. Une représentation semblable à été proposé dans [24] où un document est représenté par un vecteur de phrases qui le constituent, et chaque phrase dans le vecteur est représenté par l'ensemble des termes qui la composent. Le modèle «graphe d'index de document» (Document Graph Index) est utilisé pour maintenir la structure des phrases et identifier les séquences de mots communes entre eux.

2.6 Nouvelle mesure de similarité

Le but de cette section est de construire une mesure de similarité entre documents, en se basant sur notre modèle de représentation. La similarité entre deux documents est calculée à l'aide des similarités entre les phrases qui les constituent. La motivation derrière l'utilisation des phrases dans le calcul de similarité vient du fait que notre modèle de représentation considère la phrase comme l'unité sémantique la plus petite pour représenter le contenu du document. Nous commençons alors par définir la similarité entre deux phrases.

2.6.1 Similarité entre phrases

Soient les deux phrases S_1 et S_2, et soient T_1 et T_2 leurs arbres de dépendance respectifs. Soient les deux ensembles E_1 et E_2, les ensembles de sous-arbres fréquents inclus dans T_1 et T_2, respectivement, tels que :

$$E_1 = \{(ST_{11}, f_{11}), (ST_{12}, f_{12}), \ldots, (ST_{1l}, f_{1l})\}$$

$$E_2 = \{(ST_{21}, f_{21}), (ST_{22}, f_{22}), \ldots, (ST_{1m}, f_{1m})\}$$

Où l et m sont les nombres de sous arbres-fréquents inclus dans T_1 et T_2, respectivement. ST_{ij} est le sous-arbre fréquent j de l'arbre T_i.

La similarité entre les deux phrases S_1 et S_2 est réduite à trouver la similarité entre les deux ensembles E_1 et E_2. Une mesure de similarité entre ensembles devrait posséder les propriétés suivantes [34] :

- Plus les ensembles ont d'éléments en commun plus la mesure sera élevée.
- Plus les ensembles ont des éléments différents plus la mesure sera faible.
- La mesure est maximale si et seulement si les deux ensembles sont identiques.

Une mesure de similarité qui respecte les propriétés ci-dessous est définie de la manière suivante (selon Lin [34]) :

$$Sim(E_1, E_2) = \frac{\mid E_1 \cap E_2 \mid}{\mid E_1 \cup E_2 \mid}$$

Nous définissons l'ensemble de sous-arbres fréquents communs entre T_1 et T_2 ($E_1 \cap E_2$) par l'ensemble des sous-arbres inclus à la fois dans T_1 et T_2.

Le cœfficient de Jaccard est l'une des mesures de similarité qui respectent la définition de Lin. Nous avons utilisé le cœfficient de Jaccard pour calculer la similarité entre deux phrases. Cette mesure de similarité est une fonction des facteurs suivants :

- n : le nombre de sous-arbres communs entre T_1 et T_2 ;
- les fréquences d'apparition des sous-arbres ST_i dans T_1 et T_2 ($f_{1i} : i = 1, 2, \ldots, l$ et $f_{2i} : i = 1, 2, \ldots, m$)
- le poids de chaque sous-arbre ST_i dans les deux arbres T_1 et T_2 ($p_{1i} : i = 1, 2, \ldots, l$ et $p_{2i} : i = 1, 2, \ldots, m$)

Le poids du sous-arbre est un facteur important dans le calcul de la similarité. Il indique l'importance du sous-arbre selon sa taille en donnant plus de poids aux sous-arbres de grandes tailles. Il est donné ici par la formule suivante :

$$p_{1k} = \frac{\mid ST_{1k} \mid}{\sum_{i=1}^{l} f_{1i} \mid ST_{1i} \mid}, \; p_{2k} = \frac{\mid ST_{2k} \mid}{\sum_{j=1}^{m} f_{2j} \mid ST_{1j} \mid}$$

La similarité entre les deux phrases S_1 et S_2 est égale à la similarité entre T_1 et T_2. Elle est donnée par :

$$Sim(T_1, T_2) = \frac{\sum_{k=1}^{n} (f_{1k} p_{1k})(f_{2k} p_{2k})}{\sum_{i=1}^{l} (f_{1i} p_{1i})^2 + \sum_{j=1}^{m} (f_{2j} p_{2j})^2 - \sum_{k=1}^{n} (f_{1k} p_{1k})(f_{2k} p_{2k})}$$

2.6.2 Similarité entre documents

Après avoir calculer la similarité entre chaque paire d'arbres (phrases) des deux documents, la similarité entre deux documents d_1 et d_2 est donnée par :

$$Sim(d_1, d_2) = \frac{1}{2}[\frac{1}{m(1)} \sum_{i=1}^{m(1)} \max_{T_j \in d_2}(Sim(T_i, T_j)) + \frac{1}{m(2)} \sum_{j=1}^{m(2)} \max_{T_i \in d_1}(Sim(T_i, T_j))]$$

Où $m(1)$ est le nombre de phrases dans le document d_1, et $m(2)$ est le nombre de phrases dans le document d_2.

Cette mesure de similarité respecte bien les propriétés mathématiques d'une fonction de similarité. En effet,

- elle est positive : $\forall d_i, d_j \in D,\ 0 \leq Sim(d_i, d_j) \leq 1$;
- elle est égale à 1 lorsque les deux documents sont identiques :
  $\forall d_i \in D,\ Sim(d_i, d_i) = 1$;
- elle est symétrique : $\forall d_i, d_j \in D,\ Sim(d_i, d_j) = Sim(d_j, d_i)$.

Enfin, nous construisons une matrice $document \times document$, où les valeurs des éléments de cette matrice sont les similarités entre les paires de documents. Cette matrice sera l'entrée pour l'algorithme de clustering hiérarchique par agglomération.

2.7 Conclusion

Dans ce chapitre, nous avons présenté notre système de clustering de documents. Le but de ce système est d'améliorer la performance du clustering de documents textuels. Le système est constitué de deux composantes principales : une composante de représentation et une composante de clustering. Ces deux composantes peuvent être utilisées indépendamment. Ainsi, nous avons choisi l'algorithme hiérarchique par agglomération comme algorithme de clustering pour notre système.

Les avantages de notre approche de représentation par rapport aux approches existantes sont les suivants :

- Sémantique : notre approche prend en considération les relations de dépendance entre les mots et elle préserve la structure de la phrase, ce qui permet de représenter le contenu du document en tenant compte de la sémantique.
- Flexible : notre approche offre à l'utilisateur la possibilité de définir la taille minimale et la taille maximale des sous-arbres fréquents à extraire, ce qui permet de représenter les documents avec des mots simples et/ou des groupes de mots.
- Compacte : le nombre de sous-arbres fréquents est contrôlé par la valeur du support minimum fixée à l'avance. Une valeur raisonnable du support minimum permet d'avoir une représentation compacte des documents.

Notre approche de représentation des documents est coûteuse en terme de temps de calcul par rapport aux approches qui n'intègrent pas les techniques de TALN. Cependant, il peut y avoir un gain de temps pour l'utilisateur si le clustering est efficacement effectué.

Chapitre 3

Étude expérimentale

3.1 Introduction

L'objectif de ce chapitre est d'étudier expérimentalement les différentes approches existantes pour représenter les documents, afin de pouvoir les comparer à notre approche. Les sections 3.2 et 3.3 présentent les collections de documents utilisées pour les expériences et les mesures de qualité de clustering utilisées pour évaluer le clustering. Dans le but de choisir l'algorithme de clustering et la méthode de pondération les plus appropriés pour les collections de textes utilisées dans les expériences, nous effectuons une comparaison entre les algorithmes K-moyennes, Buckshot, HAC et bisecting K-moyennes dans la section 3.4, et nous comparons les trois techniques de pondération de termes (binaire, fréquentielle et TF-IDF) dans la section 3.5. Une fois l'algorithme de clustering et la méthode de pondération sélectionnés, l'étude expérimentale commence par une évaluation de la représentation «sac de mots» et les techniques de réduction de l'espace de représentation utilisées pour l'améliorer. Ensuite, nous évaluons la représentation par séquences de mots, à savoir l'utilisation des bi-grammes de mots et les groupes nominaux comme une manière d'exploiter l'information syntaxique contenue dans les documents. Enfin, nous évaluons notre système de clustering de documents dans la section 3.7.

3.2 Collection de textes

Pour nos expériences, nous avons utilisé trois collections de textes largement utilisées dans le domaine du clustering de documents, qui sont : Reuters-21578 [54, 18, 6], Newsgroups [24] et OHUSMED [13].

Reuters-21578 : Dans les dernières années, la collection de test la plus utilisée dans le domaine de clustering de documents est la collection Reuters-21578. Elle comporte 21578 articles qui sont apparus dans Reuters *newswire* de 1987. Chaque article est indexé manuellement par un nombre variable de thèmes (135), personnes (267), places (175), organisations (56) et bourses (39).

20 Newsgroups : Une autre collection de textes très populaire est 20 Newsgroups. C'est une collection d'environ 20 000 *Usenet messages* distribués sur 20 groupes de journaux. Quelques groupes sont très similaires.

OHSUMED : La collection OHSUMED est composée de 348 566 extraits de 270 journaux médicaux publiés dans la période 1987-1991. En général, ces textes comportent un titre et un résumé, mais certains d'entre eux peuvent ne comporter qu'un titre. Les documents sont classifiés selon les catégories du MeSH[1]. Cette collection de textes est régulièrement utilisée dans les conférences annuelles de TREC (*Text REtrieval Conferences*).

À partir de ces trois collections, nous avons extrait sept sous-collections, qui sont :

- ***Re0***, ***Re1*** et ***Re2*** : Elles sont extraites de la collection *Reuters-21578* où nous n'avons pris que les documents qui sont assignés exactement à une seule classe (thème).

[1]MeSH : Medical Subject Heading est un thesaurus utilisé pour indexer de manière standard des articles scientifiques et médicaux.

- ***20NG*** : Elle est extraite de la collection *20 Newsgroups*. Pour chaque groupe dans la collection *20 Newsgroups*, nous avons pris 100 documents qui peuvent appartenir à une ou plusieurs classes (groupes).
- ***OHS0***, ***OHS1*** et ***OHS2*** : Elles sont extraites de la collection *OHSUMED*. La particularité de cette collection est que les documents contiennent beaucoup de longues structures comme, par exemple, *calcium channel blockers* qui est différent au mot *calcium* seul.

Le tableau 3.1 résume quelques informations statistiques sur chaque collection :

Source	20 Newsgroups	Reuters-21578			OHSUMED		
Collection	**20NG**	**Re0**	**Re1**	**Re2**	**OHS0**	**OHS1**	**OHS2**
Nbr. de documents	2000	1311	2499	3089	1591	2742	2633
Nbr. de classes	20	13	25	54	10	13	23
Taille des classes	100	12-369	16-433	3-177	65-283	95-299	58-249
Nbr. de mots	17766	25858	44229	51783	26654	33991	34699
Nbr. de phrases	5874	11195	19732	23253	12648	22894	21412

Tableau 3.1: Informations statistiques sur les collections de test.

3.3 Méthodes d'évaluation

Généralement, on distingue deux types de méthodes pour évaluer la qualité d'un clustering : mesures de la qualité interne et mesures de la qualité externe.

3.3.1 Mesures de la qualité interne d'un clustering

Quand les classes des objets ne sont pas connues, des mesures de la qualité interne sont utilisées pour évaluer la qualité d'un clustering. Le résultat du clustering d'un ensemble D de N documents est un ensemble de k clusters $C_1, C_2, \ldots, C_k$ de tailles (nombre de

documents dans le cluster) $n_1, n_2, \ldots, n_k$ respectivement. Les principales mesures de la qualité interne d'un clustering sont les suivantes :

La variance

La variance est la somme des carrés des écarts de chaque document par rapport au centroïde du cluster auquel il appartient. La plus petite valeur de la variance correspond à la meilleure solution de clustering.

$$E^2 = \sum_{i=1}^{k} \sum_{d \in C_i} (dist(d, g_i))^2$$

Où d est un élément de l'ensemble de documents D, et g_i est le centroïde du cluster C_i.

La similarité d'ensemble (*Overall similarity*)

Steinbach et al. (2000) ont proposé d'utiliser la similarité d'ensemble comme mesure de la qualité interne d'un clustering [54]. La cohésion interne d'un cluster est mesurée par la similarité moyenne entre toutes paires de documents dans ce cluster. Elle est donnée par la formule suivante :

$$CI_i = \frac{\sum_{d_i \in C_i} \sum_{d_j \in C_i} sim(d_i, d_j)}{n_i^2}$$

La similarité d'ensemble d'une solution de clustering est la moyenne pondérée des similarités moyennes des clusters. Elle est donnée par la formule suivante :

$$Overall\_Similarity_{CS} = \frac{\sum_{i=1}^{k} n_i \times CI_i}{N}$$

La plus grande valeur de la similarité d'ensemble correspond à la meilleure solution de clustering.

3.3.2 Mesures de la qualité externe d'un clustering

Pour évaluer la qualité d'un clustering en fonction des classes connues de chaque document, deux mesures de la qualité externe d'un clustering sont classiquement utilisées : la *F-mesure* et l'*Entropie*. Ces mesures sont basées sur deux notions : la précision et le rappel. Elles sont données par les deux formules suivantes :

$$Precision(i,j) = \frac{N_{ij}}{N_j}$$

$$Rappel(i,j) = \frac{N_{ij}}{N_i}$$

Où N_{ij} est le nombre de documents de la classe i et qui sont affectés au cluster j, N_i est le nombre de documents de la classe i et n_j est le nombre de documents du cluster j.

Entropie

L'entropie mesure la façon dont les classes sont distribuées ou réparties dans chaque cluster. Une valeur faible de l'entropie correspond à un meilleur clustering.

L'entropie E_j du cluster j est calculée comme suit :

$$E_j = -\sum_{i=1}^{q} Precision(i,j) \log_2 (Precision(i,j))$$

Où q est le nombre de classes de la collection.

L'entropie pour une solution de clustering (CS) est la somme des entropies de chaque cluster, pondérée par leur taille :

$$E_{CS} = \frac{\sum_{j=1}^{k} n_j \times E_j}{N}$$

Où N est le nombre total de documents, k est le nombre de clusters et n_j est le nombre de documents dans le cluster j.

F-mesure

La F-mesure combine les mesures de précision et de rappel pour essayer d'assigner chaque cluster à une classe de tel sorte que deux clusters ne peuvent pas être assignés à la même classe.

La F-mesure de cluster j et classe i est donnée par :

$$F(i,j) = \frac{2 \times Precision(i,j) \times Rappel(i,j)}{Precision(i,j) + Rappel(i,j)} \tag{3.1}$$

Pour une solution de clustering (CS), la formule générale de F-mesure est donnée par :

$$F_{CS} = \frac{\sum_{j=1}^{k} n_j \max_i F(i,j)}{N} \tag{3.2}$$

Où N est le nombre total de documents et k le nombre de clusters.

La solution de clustering qui maximise la F-mesure associée, ou minimise l'entropie associée, est alors considérée comme la plus pertinente car elle correspond le mieux à la solution externe attendue.

Les deux mesures de qualité externe : la F-mesure et l'entropie sont largement utilisées dans le domaine du clustering de documents [54]. Nous allons les utiliser dans nos expériences pour évaluer le clustering.

3.4 Choix de l'algorithme de clustering

Dans le chapitre 1, nous avons présenté les différents algorithmes de clustering de documents, parmi lesquels nous avons sélectionné quatres algorithmes, largement utilisés, comme candidats pour nos expériences dont le but est de choisir l'algorithme le plus approprié à nos collections de textes. Ces algorithmes sont : K-moyennes, buckshot, hiérarchique par agglomération (HAC) et bisecting K-moyennes.

Pour les expériences de cette section, nous utilisons la représentation sac de mots, la méthode de pondération TF-IDF et le cosinus comme mesure de similarité entre documents.

3.4.1 Comparaison des différentes mesures de similarité entre clusters pour HAC

Pour l'algorithme HAC, nous avons évalué trois mesures de similarité entre clusters présentées dans la section 1.4.1 : lien simple, lien complet et lien moyen. Les résultats expérimentaux présentés dans le tableau 3.2 montrent que la mesure lien moyen est la meilleure en terme de F-mesure pour toutes les collections, et pour six collections en termes d'entropie. En fait, la méthode lien moyen fait intervenir tous les documents présents dans les deux clusters alors que les méthodes lien simple et lien complet utilisent la distance minimale et la distance maximale entre deux documents particuliers des deux clusters qui ne donnent pas une information complète sur la totalité des documents dans les cluster. Ainsi, la méthode lien moyen est moins sensible aux documents bruits.

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
Lien simple	0,0952	0,2870	0,1518	0,0736	0,2050	0,1504	0,0956
Lien complet	0,2102	0,3821	0,3088	0,3633	0,2229	0,1737	0,1446
Lien moyen	**0,2566**	**0,3972**	**0,4290**	**0,3817**	**0,4108**	**0,3040**	**0,2505**
	Entropie						
Lien simple	0,9884	0,7874	0,8587	0,8822	0,9587	0,9819	0,9659
Lien complet	0,7981	0,5399	0,5598	**0,4709**	0,9066	0,9238	0,8725
Lien moyen	**0,7260**	**0,5228**	**0,4438**	0,4832	**0,6009**	**0,6827**	**0,7329**

Tableau 3.2: Comparaison des différentes mesures de distance entre clusters pour HAC.

3.4.2 Comparaison des algorithmes de clustering

Dans cette expérience, nous avons évalué les algorithmes : K-moyennes, buckshot, HAC et bisecting K-moyennes pour le clustering de nos collections. La mesure de similarité lien moyen est utilisée pour l'algorithme HAC. Afin de contourner le problème des centroïdes initiaux, les algorithmes K-moyennes, buckshot et bisecting K-moyennes sont exécutés 20 fois. Nous avons fait varier le nombre de clusters k dans $\{5, 15, 30, 45, 60\}$, et nous avons pris le nombre de clusters qui maximise la performance. Les valeurs moyennes de F-mesure et d'entropie sont données dans le tableau 3.3.

Les résultats de la comparaison montrent que l'algorithme HAC est le meilleur pour la plupart de nos collections par rapport aux autres algorithmes. Les résultats des algorithmes K-moyennes et Buckshot sont presque similaires.

3.5 Comparaison des méthodes de pondération des termes

La fonction de pondération des mots représente numériquement l'importance d'un mot dans une collection de documents en donnant plus de poids aux mots qui permettent

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
K-moyennes	0.2160	**0.4162**	0.3583	0.3895	0.3243	0.2729	0.2378
Buckshot	0.1864	0.4023	0.3579	0.3884	0.3373	0.2684	0.2168
Bi. K-moyennes	0.1927	0.4035	0.3489	0.2754	0.3045	0.2547	0.1838
HAC	**0.2566**	0.3972	**0.4290**	**0.3817**	**0.4108**	**0.3040**	**0.2505**
	Entropie						
K-moyennes	0.8004	0.5381	0.5049	0.4729	0.7101	0.7073	0.7866
Buckshot	0.8904	**0.4742**	0.5679	**0.4597**	0.7703	0.8361	0.7945
Bi. K-moyennes	0.8582	0.4945	0.5292	0.6029	0.8248	0.8502	0.8389
HAC	**0.7260**	0.5228	**0.4438**	0.4832	**0.6009**	**0.6827**	**0.7329**

Tableau 3.3: Comparaison des algorithmes de clustering.

de distinguer les différents documents.

Dans cette expérience, nous avons évalué les différentes méthodes de pondération de termes introduites dans la section 1.2 : représentation binaire, fréquentielle (TF) et TF-IDF. L'algorithme HAC avec la mesure de similarité entre clusters lien moyen est utilisé comme algorithme de clustering. Les résultats obtenus (tableau 3.4) montrent que la méthode TF-IDF est la meilleure aussi bien en terme de F-mesure qu'en terme d'entropie pour toutes les collections. L'inconvénient majeur de la méthode binaire et la méthode fréquentielle est qu'elles sont basées sur une information statistique locale qui renseigne uniquement sur le nombre d'occurrences des mots au niveau du document. Cependant, la méthode TF-IDF prend en compte l'importance des mots dans toute la collection de documents. Ainsi, les mots qui se trouvent dans la majorité des documents comme ceux qui sont propres à quelques documents n'ont pas un rôle très important dans le clustering, et leur poids sera faible. Par exemple, les documents dans les collections OHS0, OHS1 et OHS2 partagent beaucoup de mots largement utilisé dans le domaine de la médecine en général, comme ils contiennent aussi des mots spécifiques à leurs branches de la médecine.

Dans la suite de ce document, l'algorithme utilisé dans les différentes expériences est

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
Binaire	0,1007	0,3875	0,2567	0,3207	0,2072	0,1500	0,0988
TF	0,1244	0,3995	0,3918	0,3055	0,2043	0,1501	0,1245
TF-IDF	**0,2566**	**0,3972**	**0,4290**	**0,3817**	**0,4108**	**0,3040**	**0,2505**
	Entropie						
Binaire	0,9791	0,5712	0,6886	0,5574	0,9508	0,9718	0,9477
TF	0,9597	0,6889	0,5782	0,5688	0,9536	0,9746	0,9278
TF-IDF	**0,7260**	**0,5228**	**0,4438**	**0,4832**	**0,6009**	**0,6827**	**0,7329**

Tableau 3.4: Comparaison des méthodes de pondération de termes.

HAC, la mesure de similarité entre documents est le cosinus, la mesure de similarité entre clusters est lien moyen et la méthode de pondération des termes est la fonction TF-IDF.

3.6 Comparaison des différentes approches de représentation des documents

Dans la section 1.2, nous avons présenté les différentes approches existantes de représentation des documents. Dans cette section, nous évaluons ces approches sur nos collections de textes. L'étalon de départ (*baseline*) de cette comparaison est la représentation sac de mots ($BOW_{stop,stem}$). Nous incorporons progressivement des connaissances linguistiques (lemmatisation, étiquetages des parties du discours) pour créer d'autres représentations sémantiquement plus riches, et nous étudions leur apport pour le clustering de documents par rapport à la représentation standard $BOW_{stop,stem}$. Nous examinons aussi l'utilisation des bi-grammes comme un exemple de la représentation séquences de mots «statistiques», et les groupes nominaux comme un exemple de la représentation séquences de mots «syntaxiques».

3.6.1 Représentation sac de mots

Suppression des mots vides et racinisation

Dans cette expérience, nous évaluons la représentation sac de mots. Tout d'abord, nous effectuons une comparaison entre la technique de suppression des mots vides, la technique de racinisation et la combinaison des deux techniques.

Pour cela, nous avons utilisé une liste de 571 *stopwords* de la langue anglaise pour la suppression des mots vides et le «stemmer» de Porter pour la racinisation des mots. Nous avons construit les trois représentations sac de mots suivantes :

- BOW : La représentation sac de mots avant la suppression des mots vides et la racinisation ;
- BOW_{stop} : La représentation sac de mots après la suppression des mots vides ;
- $BOW_{stop,stem}$: La représentation sac de mots après la suppression des mots vides et la racinisation.

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
BOW	0,1570	0,3945	0,3933	0,3258	0,3550	0,2591	0,2122
BOW_{stop}	0,2389	0,3955	0,4078	0,3806	0,4059	0,3034	0,2402
$BOW_{stop,stem}$	**0,2566**	**0,3972**	**0,4290**	**0,3817**	**0,4108**	**0,3040**	**0,2505**
	Entropie						
BOW	0,8285	0,5537	0,4840	0,5666	0,6753	0,7550	0,7913
BOW_{stop}	0,7580	0,5417	0,4527	0,4869	0,6118	0,6992	0,7482
$BOW_{stop,stem}$	**0,7260**	**0,5228**	**0,4438**	**0,4832**	**0,6009**	**0,6827**	**0,7329**

Tableau 3.5: Comparaison des techniques de reduction de l'espace de représentation.

Les résultats obtenus montrent que la $BOW_{stop,stem}$ donne des résultats légèrement meilleurs par rapport aux autres représentations.

Lemmatisation

La racinisation n'est pas basée sur des contraintes linguistiques fortes. Elle peut créer des mots qui n'existent pas (par exemple, la racine de *queries* est *querie*, un mot qui n'existe pas), comme des mots différents peuvent être réduit à la même racine (par exemple, la racine des mots *police* et *policy* est *polic*). De manière plus exacte, la lemmatisation permet d'associer à chaque mot son lemme grâce à un étiquetage morpho-syntaxique (nom, verbe, adjectif, etc.).

Dans cette expérience, nous avons examiné l'utilisation des lemmes au lieu des racines. Nous avons utilisé l'analyseur syntaxique de Stanford pour accomplir la tâche de lemmatisation.

Les résultats obtenus (tableau 3.6) montrent que la lemmatisation et le stemming donnent presque des résultats similaires pour la plupart des collections. La lemmatisation nécessite beaucoup de ressources linguistiques alors que l'utilisation de la racinisation comme alternative a le même effet pour le clustering avec un coût minimal. En terme de dimension, la lemmatisation réduit l'espace de représentation pour nos collections de 19-24%, alors que la racinisation le réduit de 27-30%.

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
$BOW_{stop,stem}$	**0,2566**	0,3972	0,4290	**0,3817**	0,4108	0,3040	**0,2505**
BOW_{lemme}	0,2260	**0,4203**	**0,4447**	0,3425	**0,4116**	**0,3041**	0,2433
	Entropie						
$BOW_{stop,stem}$	**0,7260**	0,5228	0,4438	**0,4832**	0,6009	**0,6827**	**0,7329**
BOW_{lemme}	0,8158	**0,5038**	**0,4419**	0,5086	**0,5984**	0,6938	0,7398

Tableau 3.6: Comparaison entre l'utilisation de la racinisation et la lemmatisation pour la représentation des documents.

Étiquetage des parties du discours

L'étiquetage des parties du discours est utilisé pour la sélection des attributs qui portent plus d'information sur le contenu du document. Les noms sont susceptibles d'être les parties du discours les plus discriminants par rapport aux adjectifs, verbes et adverbes. Dans cette expérience, nous avons examiné l'impact de l'utilisation des noms, adjectifs et verbes comme attributs pour la représentation des documents. Pour l'étiquetage des parties du discours, nous avons utilisé l'analyseur syntaxique de Stanford. Cet outil attribue une catégorie grammaticale à tous les mots contenus dans un texte. Par exemple, pour le texte *«The documents are correctly classified.»*, l'étiquetage des parties du discours donne : *The/DT documents/NNS are/VBP correctly/RB classified/VBN ./.*.

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
$BOW_{stop,stem}$	**0,2566**	**0,3972**	0,4290	**0,3817**	**0,4108**	0,3040	**0,2505**
Noms	0,2050	0,3499	0,4407	0,3745	0,3640	0,2616	0,2274
Noms et adjectifs	0,2359	0,3959	**0,4481**	0,3470	0,3914	0,3037	0,2363
Noms, adj. et verbes	0,2330	0,3998	0,4173	0,3475	0,3839	**0,3064**	0,2492
	Entropie						
$BOW_{stop,stem}$	**0,7260**	0,5228	0,4438	0,4832	0,6009	0,6827	**0,7329**
Noms	0,8538	0,5066	**0,4258**	**0,4774**	0,6596	0,7134	0,7668
Noms et adjectifs	0,7983	**0,4942**	0,4355	0,5013	**0,5997**	0,6868	0,7438
Noms, adj. et verbes	0,7671	0,5160	0,4386	0,5076	0,6073	**0,6726**	0,7406

Tableau 3.7: Utilisation de l'étiquetage des parties du discours dans la représentation des documents.

Les résultats obtenus (tableau 3.7) montrent que l'utilisation de l'étiquetage des parties du discours n'améliore pas la qualité du clustering. En terme de F-mesure, la représentation $BOW_{stop,stem}$ est la meilleure pour 5 collections. En général, les verbes ne présentent pas une grande importance pour le clustering. La représentation basée sur les noms et les adjectifs donne presque les mêmes résultats que la représentation standard $BOW_{stop,stem}$.

3.6.2 Représentation par séquences de mots

Dans cette section, nous avons étudié l'effet de la représentation par séquences de mots sur la qualité du clustering de documents.

Bi-grammes de mots

La représentation par séquences de mots la plus simple est la représentation bi-grammes. Pour créer une représentation bi-grammes pour chaque document, nous avons défini les étapes suivantes :

- Suppression des mots vides ;
- Racinisation de tous les mots ;
- Construction des paires successives de deux mots (bi-grammes) en prenant en compte les frontières des phrases ;
- Normalisation des bi-grammes : les deux mots de chaque bi-gramme sont convertis en minuscule et ils sont ordonnés par ordre alphabétique[2].

Exemple : Pour le texte suivant : «*Replacement of an aortic valve cusp after neonatal endocarditis. Septic arthritis developed in a neonate after an infection of her hand.*»

La liste des bi-grammes est :(aortic, replac)(aortic, valv)(cusp, valv)(cusp, neonat)(endocard, neonat)(arthriti, septic)(arthriti, develop)(develop, neonat)(infect, neonat)(hand, infect)

Les résultats obtenus sont présentés dans le (tableau 3.8). L'utilisation des bi-grammes n'améliore pas significativement la qualité du clustering pour nos collections de textes sauf pour la collection 20NG. Comme les classes originales de la collection 20NG sont très proches les unes des autres, les bi-grammes de mots deviennent des attributs plus

[2]D'après nos expériences, l'utilisation de bi-grammes non ordonnés dégrade la qualité du clustering de documents par rapport aux bi-grammes ordonnés

discriminants que les mots individuels pour séparer ces classes.

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
$BOW_{stop,stem}$	0,2566	0,3972	**0,4290**	**0,3817**	**0,4108**	**0,3040**	**0,2505**
Bigrammes	**0,4725**	0,4347	0,3723	0,3544	0,3125	0,2476	0,2079
Combinaison	0,2536	**0,4513**	0,4139	0,3788	0,3703	0,2918	0,2473
	Entropie						
$BOW_{stop,stem}$	0,7260	0,5228	0,4438	0,4832	0,6009	**0,6827**	**0,7329**
Bigrammes	**0,4477**	0,4945	0,4605	0,4950	0,7118	0,7486	0,7909
Combinaison	0,7316	**0,4814**	**0,4311**	**0,4548**	**0,5949**	0,6968	0,7440

Tableau 3.8: Comparaison entre la représentation sac de mots et la représentation bigrammes.

Groupes nominaux

Le format en sortie de l'analyse syntaxique d'une phrase en utilisant *Stanford parser* peut être une structure d'arbre des groupes syntaxiques indiquant le découpage récursif de la phrase en constituants grammaticaux (par exemple : des groupes nominaux et des groupes verbaux). La figure 3.1 montre la structure d'arbre des groupes syntaxiques de la phrase «*The documents are correctly classified.*».

Dans le chapitre 1, nous avons présenté quelques travaux qui ont essayé d'exploiter la structure de l'arbre des groupes syntaxiques pour générer des nouveaux attributs pour la représentation des documents afin d'améliorer le clustering. Une manière d'utiliser cet arbre est de prendre que les groupes nominaux comme attributs pour le modèle vectoriel.

Dans l'expérience suivante, nous avons comparé une représentation basée sur les groupes nominaux avec la représentation sac de mots standard. Nous avons défini les étapes suivantes pour créer une représentation avec groupes nominaux :

- Extraire tous les groupes nominaux ;

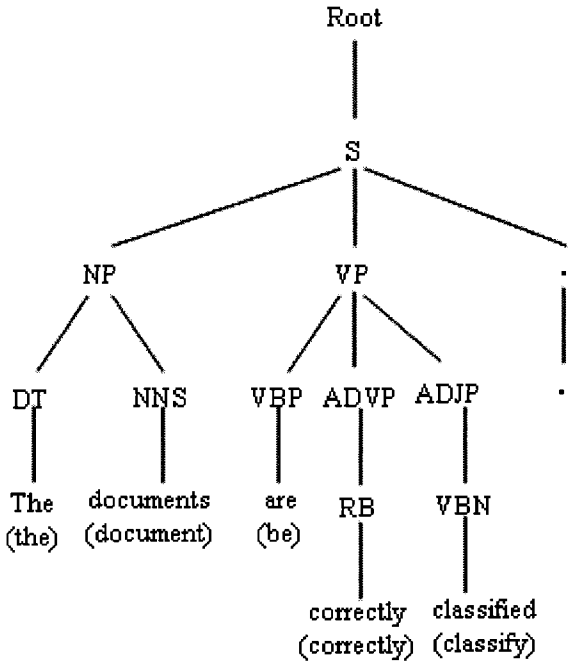


Figure 3.1: La structure d'arbre des groupes grammaticaux de la phrase : *The documents are correctly classified.*

- Pour chaque groupe nominal, garder que les noms, les adjectifs et les verbes, supprimer les mots vides et ordonner les mots par ordre alphabétique ;
- Supprimer des groupes constitués d'un seul mots et les groupes redondants.

Collections	20NG	Re0	Rel	Re2	OHS0	OHS1	OHS2
	F-mesure						
$BOW_{stop,stem}$	**0.2566**	**0.3972**	**0.4290**	**0.3817**	**0.4108**	**0.3040**	**0.2505**
Groupes nominaux	0.1504	0.3866	0.3591	0.2927	0.2647	0.2076	0.1711
	Entropie						
$BOW_{stop,stem}$	**0.7260**	**0.5228**	**0.4438**	**0.4832**	**0.6009**	**0.6827**	**0.7329**
Groupes nominaux	0.9046	0.5733	0.5252	0.5700	0.7496	0.8146	0.8263

Tableau 3.9: Utilisation des groupes nominaux pour la représentation des documents.

Les résultats obtenus (tableau 3.9) montrent que l'utilisation des groupes nominaux n'améliore pas la qualité du clustering. En effet, les groupes nominaux portent plus de sémantique par rapport aux mots individuels, mais ils possèdent une qualité statistique inférieure.

La présente section a montré l'impact de l'utilisation des connaissances linguistiques (lemmatisation, étiquetage des parties du discours, groupes nominaux) dans la représentation des documents sur les performances du clustering de documents. Les résultats de nos expériences ont montré que ces approches améliorent légèrement le clustering dans certains cas, et elles le dégradent dans d'autres cas. Toutefois, l'utilisation des techniques TALN permet d'extraire des structures qui portent plus de sémantique sur le contenu des documents que les mots simples, mais leur exploitation efficace dans les systèmes de clustering de documents dépend de la nature de l'intégration de ces structures dans la représentation des documents et le calcul de similarité entre documents, en particulier le modèle de représentation adopté, la méthode de pondération appliquée et la mesure de similarité utilisée.

3.7 Évaluation de notre système de clustering

3.7.1 Évaluation de notre approche de représentation des documents

La première expérience décrite dans cette section évalue notre approche de représentation des documents en utilisant les sous-arbres fréquents extraits. Dans cette expérience, nous avons utilisé le cosinus pour calculer la similarité entre documents et la méthode TF-IDF pour la pondération des sous-arbres.

La mesure de similarité du cosinus s'applique sur le modèle vectoriel. Nous devons donc représenter les documents dans un espace vectoriel de n dimensions, où n est le nombre de sous-arbres fréquents. Chaque dimension dans cet espace vectoriel correspond à un sous-arbre fréquent, et chaque document est représenté par un vecteur TF-IDF (voir section 1.2) :

$$\overrightarrow{d} = (TfIdf(ST_1, d), \ldots, TfIdf(ST_n, d))$$

Le poids TF-IDF d'un sous arbre ST dans un document d est définie comme suit :

$$TfIdf(ST, d) = Tf(ST, d) \times \log(\frac{\mid D \mid}{df(ST, D)})$$

Où $\mid D \mid$ est le nombre de documents dans la collection, $Tf(ST, d)$ est le nombre d'occurrences de ST dans l'ensemble $\hat{d}$ d'arbres de dépendance du document d ($\hat{d} = \{T_1, T_2, \ldots, T_m\}$), tel que :

$$Tf(ST, d) = \sum_{T_i \in \hat{d}} freq(ST, T_i)$$

Et $df(ST, D)$ est le nombre de documents contenant le sous-arbre ST, tel que :

$$df(ST, D) = \sum_{\hat{d}_i \in \mathcal{D}} \phi(ST, \hat{d}_i)$$

Où $\phi(ST, \hat{d})$ est une fonction indicatrice définie comme suit :

$$\phi(ST, \hat{d}) = \begin{cases} 1 & \text{si } ST\text{est un sous-arbre au moins d'un arbre } T_i \in \hat{d} \\ 0 & \text{sinon} \end{cases}$$

Nous devons aussi spécifier les paramètres : min_sup, min_motif et max_motif.

La taille minimale des sous-arbres

Afin d'étudier l'effet de la taille minimale des sous-arbres fréquents, nous avons effectué deux expériences. Dans les deux expériences, nous avons fixé le support minimum à 1% ($min\_sup = 0,01$) et la taille maximale des sous-arbres à 4 ($max\_motif = 4$). En effet, les sous-arbres fréquents de taille plus de 4 sont rares et n'améliorent pas le clustering.

- **Expérience 1 :** Dans cette expérience, nous avons utilisé les sous-arbres fréquents de taille minimale égale à 1 ($min\_motif = 1$). Les résultats obtenus pour cette expérience sont présentés dans le tableau 3.10. En terme de F-mesure, nous remarquons que cette représentation améliore la qualité du clustering pour les collections Re0, Rel et Re2, mais elle est moins bonne par rapport à la représentation standard $BOW_{stop,stem}$ pour les collections 20NG, OHS0, OHS1 et OHS2. En terme d'entropie, elle donne les meilleurs résultats pour 6 collections, Re0, Rel, Re2, OHS0, OHS1 et OHS2.
- **Expérience 2 :** Dans cette expérience, nous avons utilisé les sous-arbres fréquents de taille minimale égale à 2 $min\_motif = 2$). Les résultats obtenus montrent que l'utilisation de cette représentation n'améliore que la qualité du clustering de la collection 20NG en terme de F-mesure et entropie. Aucune amélioration n'est apportée

pour les autres collections par rapport à la représentation standard $BOW_{stop,stem}$.

En analysant ces résultats, nous constatons que la valeur du paramètre min_motif est liée à la nature de la collection. Dans la collection 20NG, la plupart des classes originales sont très similaires dont les documents partagent beaucoup de mots simples, par exemple, les classes : *talk.politics.guns*, *talk.politics.mideast* et *talk.politics.misc*. L'enrichissement de la représentation avec des structures plus longues (sous-arbres de taille 2, 3 et 4) permettent d'avoir plus de similarité entre les documents de la même classe et plus de dissimilarité entre les documents des classes différentes. Dans les collections Re0, Re1 et Re2, les classes originales sont disjointes donc l'enrichissement de la représentation avec des structures de taille plus de 1 n'a aucun effet sur le calcul de similarité entre les documents. Les documents dans OHS0, OHS1 et OHS2 ont des longueurs différentes. Il existe des documents qui ne contiennent qu'une seule phrase. Dans ce cas, l'utilisation des structures plus longues que les mots simples n'améliore pas la qualité du clustering. Le problème vient du calcul de similarité entre deux documents de longueurs différentes. En utilisant le cosinus, deux documents de longueurs différentes peuvent avoir une valeur de similarité faible, même s'ils partagent beaucoup de sous-arbres fréquents en commun à cause de la grande norme du document long. La mesure de similarité que nous avons proposée permet de régler ce problème puisque elle est basée sur les similarités entre les phrases des documents. Les documents ne sont pas considérés comme une seule unité mais comme un ensemble de phrases donc la longueur des documents n'influence pas beaucoup le calcul de similarité.

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
BOW standard	0.2566	0.3972	0.4290	0.3817	**0.4108**	**0.3040**	**0.2505**
Expérience 1	0.2220	**0.5257**	**O.4526**	**0.5033**	0.3926	0.2942	0.2435
Expérience 2	**0.6188**	0.4476	O.2898	0.3566	0.3319	0.2413	0.2253
	Entropie						
BOW standard	0.7260	0.5228	0.4438	0.4832	0.6009	0.6827	0.7329
Expérience 1	0.7410	**0.3934**	**0.4044**	**0.3157**	**0.5629**	**0.6307**	**0.5158**
Expérience 2	**0.3108**	0.5565	O.7143	0.5142	0.7110	0.7759	0.7903

Tableau 3.10: Évaluation de notre approche de représentation des documents.

Sensibilité au choix du support

Dans cette expérience, nous avons étudié l'effet du choix du support sur la représentation. La figure 3.2 montre les résultats obtenus. Nous observons que les meilleurs résultats du clustering sont obtenus pour une petite valeur (1%) du support minimum.

Après plusieurs expériences, nous suggérons l'utilisation d'une valeur de support minimum qui dépend au nombre moyen de phrases par document. En fait, nous recherchons des sous-arbres présents dans plus d'un document. La formule suivante peut être utilisée pour fixer la valeur du support :

$$min\_sup = \alpha \times \frac{\text{Nombre moyen de phrases par document}}{\text{Nombre total de phrases dans la collection}}$$

où α est une constante qui indique approximativement le nombre de documents devront contenir un sous-arbre quelconque.

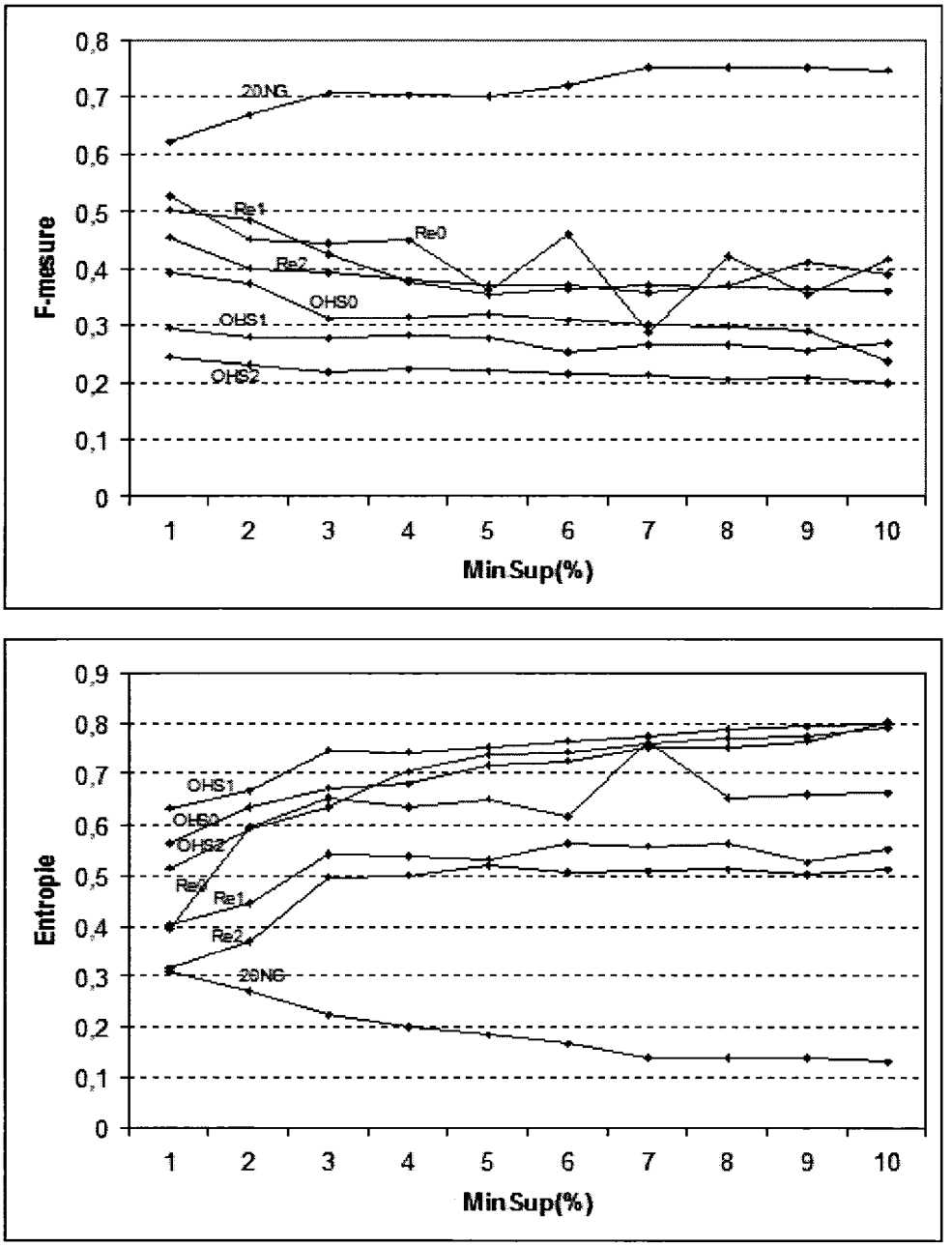


Figure 3.2: Sensibilité au choix du support.

3.7.2 Évaluation de notre mesure de similarité

Dans cette expérience, nous avons évalué la nouvelle mesure de similarité entre documents proposée dans le chapitre 2. Nous avons fixé le support minimum à 1% ($min\_sup = 0,01$), la taille minimale à 1 ($min\_motif = 1$) et la taille maximale à 4 ($max\_motif = 4$).

Collections	20NG	Re0	Re1	Re2	OHS0	OHS1	OHS2
	F-mesure						
$BOW_{stop,stem}$	0.2566	0.3972	0.4290	0.3817	0.4108	0.3040	0.2505
Notre mesure	**0.7572**	**0.5812**	**0.6126**	**0.6385**	**0.7231**	**0.7039**	**0.6716**
	Entropie						
$BOW_{stop,stem}$	0.7260	0.5228	0.4438	0.4832	0.6009	0.6827	0.7329
Notre mesure	**0.1402**	**0.1254**	**0.1368**	**0.1247**	**0.0913**	**0.0927**	**0.0854**

Tableau 3.11: Évaluation de notre mesure de similarité.

Les résultats obtenus (tableau 3.11) montrent l'efficacité de notre mesure de similarité par rapport à la mesure cosinus. Nous pouvons expliquer ces résultats par les points suivants :

- La similarité entre documents de longueurs différentes : Le calcul de similarité entre documents n'est pas disproportionnellement influencé par la longueur des documents, puisque ce calcul est basé sur les similarités entre les phrases.
- Le poids de sous-arbres : Dans le calcul de similarité entre documents, nous avons donné plus de poids aux sous-arbres de grande taille. En fait, un sous-arbre de taille supérieure à 1 porte plus de sémantique mais sa fréquence d'apparition est faible par rapport à un sous-arbre de taille 1. Pour avoir un équilibre, nous avons associé un poids plus grand aux sous-arbres de taille supérieure à 1.
- La dépendance de la mesure de similarité au choix des attributs représentants les documents : Notre représentation est basée sur les phrases, une mesure de similarité basée aussi sur les phrases est plus appropriée à notre approche de représentation.

Conclusion

La représentation des documents joue un role très important pour le clustering de documents. En général, la plupart des représentations proposées dans la littérature se base sur le modèle vectoriel dont la représentation la plus populaire est «sac de mots». Cette dernière est souvent raffinée par la suppression des mots vides et la racinisation de tous les mots. La méthode TF-IDF est généralement utilisée pour la pondération des mots. La représentation «sac de mots» génère un espace de représentation de grande dimension qui nous empêche d'utiliser efficacement les algorithmes de clustering. En plus, elle ignore les relations syntaxiques et sémantiques entre les mots. Pour réduire l'espace de représentation, d'autres modèles basés sur des techniques de traitement de la langue naturelle comme la lemmatisation et l'étiquetage des parties du discours ont été proposés. Nous avons testé ces modèles sur plusieurs collections de documents textuels. Les résultats obtenus ont montré qu'ils permettent essentiellement de réduire l'espace de représentation sans amélioration de la qualité du clustering par rapport à la représentation «sac de mots». Dans le but d'avoir plus de sémantique par la combinaison des mots, d'autres représentations alternatives définissent les groupes de mots à partir des données statistiques (bigrammes) ou linguistiques (groupes nominaux) au lieu des mots individuels. Les résultats obtenus lors de notre étude expérimentale ont montré que ces représentations ne sont pas notablement meilleures que la représentation «sac de mots».

L'objectif de ce travail est de trouver une représentation compacte qui représente

mieux le contenu du document. L'information syntaxique nous paraît intéressante malgré les résultats négatifs observés lors de l'utilisation des groupes nominaux. L'arbre de dépendance syntaxique obtenu par l'analyse syntaxique d'une phrase est une forme de l'information syntaxique. L'idée de notre travail est de trouver une manière efficace d'exploiter les arbres de dépendance pour la représentation et le clustering de documents textuels.

Dans ce mémoire, nous avons proposé une nouvelle approche de représentation des documents basée sur les sous-arbres fréquents d'arbres de dépendance. Ces sous-arbres sont extraits en utilisant les techniques de la fouille d'arbres. Les avantages de cette représentation sont :

- Elle prend en considération les relations de dépendances syntaxiques entre les mots formant des groupes de mots qui portent plus de sémantique que les mots individuels et les séquences de mots regroupés de façon arbitraire. Ainsi, elle considère la phrase comme unité minimale pour représenter le contenu du texte.
- Elle permet aussi de combiner les mots simples avec les groupes de mots. Les paramètres taille minimale et taille maximale des sous-arbres nous donnent la possibilité de prendre les sous-arbres de taille 1 (mots simples), et de taille plus de 1 (groupes de mots).
- Elle est compacte du fait que les mots vides sont supprimés, les mots sont réduits à leur racine (lemme), et le nombre de sous-arbres à extraire est contrôlé par la valeur du support minimal fixé à l'avance.

La deuxième contribution de ce travail est la proposition d'une nouvelle mesure de similarité entre documents basée sur notre approche de représentation. L'avantage de cette mesure est l'utilisation de la phrase comme unité de base dans le calcul de similarité.

Enfin, un système de clustering de documents est construit afin de tester notre approche. Pour ce système, nous avons choisi d'utiliser l'algorithme hiérarchique par agglomération (HAC) avec la mesure de similarité entre clusters lien moyen. Les résultats

obtenus par l'application de notre méthode prouvent son efficacité.

Une extension de ce travail consiste à enrichir notre approche de représentation par l'intégration de l'information sémantique (synonymie, polysémie) en utilisant un thésaurus comme WordNet. Nous envisageons aussi la sélection d'un ensemble de phrases pertinentes qui reflètent le contenu du document au lieu de prendre toutes les phrases. Ces phrases peuvent être extraites à partir des parties du document riches d'information sur son contenu comme le titre et le résumé.

Une autre direction de recherche consiste à donner des descripteurs à chaque cluster généré. La description du cluster présente le contenu du cluster d'une manière compréhensible aux utilisateurs. Nous envisageons la possibilité de sélectionner les descripteurs d'un cluster parmi les sous-arbres caractéristiques dans ce cluster.

Notre mesure de similarité peut aussi être appliquée pour la recherche d'information afin identifier les documents les plus proches de la requête formulée par l'utilisateur.

Bibliographie

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the* 20^{th} *International Conference on Very Large Data Bases (VLDB 1994)*, pages 487–499, Santiago de Chile, Chile, 1994.

[2] A. Arampatzis, T. P. van der Weide, C. H. A. Koster, and P. van Bommel. An Evaluation of Linguistically-motivated Indexing Schemes. In *Proceedings of the* 22^{nd} *BCS-IRSG Annual Colloquium on Information Retrieval Research*, pages 34–45, Cambridge, United Kingdom, 2000.

[3] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient Substructure Discovery from Large Semi-structured Data. In *Proceedings of the* 2^{nd} *SIAM International Conference on Data Mining*, pages 158–174, Arlington, Virginia, USA, April 11-13 2002.

[4] J. Bakus, M. Hussin, and M. S. Kamel. SOM-Based Document Clustering Using Phrases. In *Proceedings of the* 9^{th} *International Conference on Neural Information Processing (ICONIP 2002)*, volume 5, pages 2212–2216, Singapore, 2002.

[5] R. Basili, A. Moschitti, and M. T. Pazienza. Language-Sensitive Text Classification. In *Proceedings of* 6^{th} *International Conference "Recherche d'Information Assistée par Ordinateur" (RIAO 2000)*, pages 331–343, Paris, France, 2000.

[6] F. Beil, M. Ester, and X. Xu. Frequent Term-Based Text Clustering. In *Proceedings of the* 8^{th} *International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, pages 436–442, Edmonton, Alberta, Canada, 2002.

[7] P. Bellot and M. El-Bèze. A Clustering Method for Information Retrieval. Technical Report IR-0199, Laboratoire d'Informatique d'Avignon, France, 1999.

[8] P. Bellot and M. El-Bèze. Clustering by Means of Unsupervised Decision Trees or Hierarchical and K-means-like Algorithm. In *Proceedings of the* 6^{th} *International Conference "Recherche d'Information Assistée par Ordinateur" (RIAO 2000)*, pages 344–363, Paris, France, 2000.

[9] P. Berkhin. Survey Of Clustering Data Mining Techniques. Technical report, Accrue Software, San Jose, California, USA, 2002.

[10] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G.Zweig. Syntactic Clustering of the Web. In *Proceedings of the* 6^{th} *International World Wide Web Conference (WWW-6)*, pages 1157–1166, Santa Clara, California, USA, April 7-11 1997.

[11] M. F. Caropreso, S. Matwin, and F. Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In A. G. Chin, editor, *Text Databases and Document Management : Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, USA, 2001.

[12] H. Chim and X. Deng. A new suffix tree similarity measure for document clustering. In *Proceedings of the 16th international conference on World Wide Web (WWW 2007)*, pages 121–130, Banff, Alberta, Canada, 2007.

[13] W. W. Chu, Z. Liu, and W. Mao. Techniques for Textual Document Indexing and Retrieval Knowledge Sources and Data Mining. In W. Wu, H. Xiong, and S. Shekhar, editors, *Clustering and Information Retrieval*, pages 135–160. Kluwer, 2003.

[14] C. Clifton, R. Cooley, and J. Rennie. TopCat : Data Mining for Topic Identification in a Text Corpus. *IEEE Transactions on Knowledge and Data Engineering*, 16(8) :949–964, 2004.

[15] D. R. Cutting, J. O. Pedersen, D. Karger, and J. W. Tukey. Scatter/Gather : A Cluster-based Approach to Browsing Large Document Collections. In *Proceedings of the* 15^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, Copenhagen, Denmark, 1992.

[16] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the* 2_{nd} *International Conference on Knowledge Discovery and Data Mining(KDD 1996)*, pages 226–231, Portland, OR, 1996.

[17] W. B. Frakes and R. Baeza-Yates. *Information Retrieval : Data Structures and Algorithms.* Englewood Cliffs : Prentice-Hall, 1992.

[18] B. C. M. Fung, K. Wang, and M. Ester. Hierarchical Document Clustering Using Frequent Itemsets. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 59–70, San Francisco, California, USA, May 1-3 2003.

[19] R.G. Gaizauskas, H. Cunningham, Y. Wilks, P. Rodgers, and K. Humphreys. GATE – an Environment to Support Research and Development in Natural Language Engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-96)*, Toulouse, France, October 1996.

[20] S. Guha, R. Rastogi, and K. Shim. CURE : An Efficient Clustering Algorithm for Large Databases. In *Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD 1998)*, pages 73–84, Seattle, Washington, USA, June 2-4 1998.

[21] K. M. Hammouda. Web Mining : Clustering Web Documents, A Preliminary Review. Technical report, University of Waterloo, Ontario, Canada, 2001.

[22] K. M. Hammouda. Web Mining : Identifying Document Structure for Web Document Clustering. Master's thesis, Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, 2002.

[23] K. M. Hammouda and M. S. Kamel. Phrase-based Document Similarity Based on an Index Graph Model. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pages 203–210, Maebashi City, Japan, December 9-12 2002.

[24] K. M. Hammouda and M. S. Kamel. Efficient Phrase-Based Document Indexing for Web Document Clustering. *IEEE Transactions on Knowledge ans Data Engineering*, 16(10) :1279–1296, October 2004.

[25] J. Han and M. Kamber. *Data Mining : Concepts and Techniques.* Morgan Kaufmann Publishers, 2001.

[26] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. WEBSOM—Self-Organizing Maps of Document Collections. In *Workshop on Self-Organizing Maps (WSOM 1997)*, pages 310–315. Espoo, Finland, 1997.

[27] I. Iliopoulos, A. J. Enright, and C. A. Ouzounis. TEXTQUEST : Document Clustering of MEDLINE Abstracts For Concept Discovery In Molecular Biology. In *Proceedings of the* 6^{th} *Annual Pacific Symposium on Biocomputing (PSB 001)*, pages 384–395, 2001.

[28] T. Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proceedings of* 16^{th} *International Conference on Machine Learning (ICML 1999)*, pages 200–209, San Francisco, California, USA, 1999.

[29] A. Joshi, Z. Jiang, R. Krishnapuram, and L. Yi. Retriever : Improving web search engine results using clustering. In A. Gangopadhyay, editor, *Managing Business With Electronic Commerce : Issues and Trends*, pages 59–81. Idea Group Publishing, Hershey, Pennsylvania, USA, 2001.

[30] L. Kaufman and P. Rousseeuw. *Finding Groups in Data : An Introduction to Cluster Analysis.* John Wiley and Sons, New York, NY, USA, 1990.

[31] J. LaroccaNeto, A. D. Santos, C. A. A. Kaestner, and A. A. Freitas. Document Clustering and Text Summarization. In *Proceedings of the* 4^{th} *International Conference on Practical Applications of Knowledge Discovery and Data Mining (PADD-2000)*, pages 41–55, London, United Kingdom, 2000.

[32] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the* 5^{th} *ACM SIGKDD international conference on*

Knowledge discovery and data mining (KDD 1999), pages 16–22, San Diego, California, USA, 1999.

[33] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2) :451–461, February 2003.

[34] D. Lin. An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, Madison, Wisconson, USA.

[35] K. I. Lin and R. Kondadadi. A word-based soft clustering algorithm for documents. In *Proceedings of the* 16^{th} *International Conference on Computers and Their Applications*, pages 391–394, March 2001.

[36] X. Liu, Y. Gong, W. Xu, and S. Zhu. Document clustering with cluster refinement and model selection capabilities. In *Proceedings of the* 25^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 191–198, Tampere, Finland, August 11-15 2002.

[37] Y.S. Maarek, R. Fagin, I.Z. Ben-Shaul, and D. Pelleg. Ephemeral Document Clustering for Web Applications. Technical Report RJ 10186, IBM Research, 2000.

[38] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the* 5^{th} *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, California, USA, 1967.

[39] M. Mitra, C. Buckley, A. Singhal, and C. Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO-97,* 5^{th} *International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 200–214, Montreal, Canada, June 1997.

[40] D. Mladenic and M. Grobelnik. Word Sequence as Features in Text-learning. In *Proceedings of the* 17^{th} *Electrotechnical and Computer Science Conference (ERK-1998)*, pages 200–214, Ljubljana, Slovenia, June 1998.

[41] D. S. Modha and W. S. Spangler. Feature Weighting in k-Means Clustering. *Machine Learning*, 52(3) :217–237, 2003.

[42] S. Morinaga, H. Arimura, T. Ikeda, Y. Sakao, and S. Akamine. Key semantics extraction by dependency tree mining. In *Proceeding of the* 11^{th} *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD 2005)*, pages 666–671, Chicago, Illinois, USA, 2005.

[43] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3) :130–137, July 1980.

[44] D. Pullwitt and R. Der. Integrating Contextual Information into Text Document Clustering with Self-Organizing Maps. In *Advances in Self-Organizing Maps : Proceedings of the Workshop on Self-Organizing Maps (WSOM 2001)*, pages 54–60, Lincoln, United Kingdom, June 2001.

[45] L. Yi R. Krishnapuram, A. Joshi. A Fuzzy Relative of the k-Medoids Algorithm with Application to Web Document and Snippet Clustering. In *Proceedings of IEEE International Conference Fuzzy Systems*, Korea, August 1999.

[46] J. C. Reynar and A. Ratnaparkhi. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the* 5^{th} *Conference on Applied Natural Language Processing*, pages 16–19, Washington, D.C., USA, March 31-April 3 1997.

[47] G. Salton and C. Buckley. Term-Weighting Approaches In Automatic Text Retrieval. *Information Processing and Management*, 24(5) :513–523, 1988.

[48] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, NY ,USA, 1983.

[49] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM*, 18(11) :613–620, November 1975.

[50] H. Schmid. LoPar : Design and Implementation. *Arbeitspapiere des Sonderforschungsbereiches*, 340(149), July 2000.

[51] S. Scott and S. Matwin. Feature Engineering for Text Classification. In *Proceedings of the* 16^{th} *International Conference on Machine Learning (ICML 1999).*

[52] K. Skogmar and J. Olsson. Clustering Documents with Vector Space Model using N-Grams. Course work, Lund Institute of Technology, Sweden, 2002.

[53] D. Sleator and D. Temperley. Parsing English with a Link Grammar. Technical Report CMU-CS-9 1-196, Department of Computer Science, Carnegie Mellon University, October 1991.

[54] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[55] A. Strehl, J. Ghosh, and R. Mooney. Impact of Similarity Measures on Web-page Clustering. In *Proceedings of the* 17^{th} *National Conference on Artificial Intelligence : Workshop of Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, Austin, Texas, USA, July 30-31 2000.

[56] Z. Wang, U. Topkara, T. Schultz, and A. Waibel. Towards Universal Speech Recognition. In *Proceedings of the 2002 International Conference on Multimodal Interfaces (ICMI 2002)*, October 2002.

[57] S. Weiss, H. White, and C. Apté. Lightweight document clustering. In *Proceedings of PKDD-2000, Springer*, pages 665–672, 2000.

[58] J. R. Wen, J. Y. Nie, and H. J. Zhang. Clustering user queries of a search engine. In *Proceedings of the* 10^{th} *International Conference on World Wide Web (WWW 2001)*, pages 162–168, Hong Kong, China, 2001.

[59] J. R. Wen and H. J. Zhang. Query Clustering in the Web Context. In W. Wu, H. Xiong, and S. Shekhar, editors, *Clustering and Information Retrieval*, pages 195–226. Kluwer, 2003.

[60] P. Willett. Recent Trends in Hierarchical Document Clustering : a Critical Review. *Information Processing and Management*, 24(5) :577–597, 1988.

[61] W. Wong and A. Fu. Incremental document clustering for web page classification. In *IEEE 2000 International Conference on Information Society in the* 21^{st} *Century : Emerging Technologies and New Challenges (IS2000)*, Japan, November 5-8 2000.

[62] M. J. Zaki. Efficiently Mining Frequent Trees in a Forest : Algorithms and Applications. *IEEE Transaction on Knowledge and Data Engineering, special issue on Mining Biological Data*, 17(8) :1021–1035, 2005.

[63] O. Zamir and O. Etzioni. Web Document Clustering : A Feasibility Demonstration. In *Proceedings of the* 21^{st} *annual international ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 46–54, Melbourne, Australia, 1998.

[64] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH : An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Quebec, Canada, June 4-6 1996.

[65] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the* 11^{th} *international conference on Information and knowledge management (CIKM 2002)*, pages 515–524, McLean, Virginia, USA, 2002.