UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté des Sciences
Département d'Informatique

MEMOIRE DE MAGISTER

Spécialité : Système d'Information et de la Connaissance

SUMMARIZATION DES DOCUMENTS

DANS LES CUBES DE TEXTES

Par

Aicha LABABOU

Devant le jury composé de

S. OUKID	Maître de conférences-A, Université de Blida	Présidente
Z. ALIMAZIGHI	Professeur, U.S.T.H.B, Alger	Examinatrice
D. ZEGOUR	Professeur, E.S.I, Alger	Examinateur
N. BENBLIDIA	Maître de conférences-A, Université de Blida	Promotrice
O. BOUSSAID	Professeur, Université Lumière Lyon 2, France	Co-Promoteur
W. HIDOUCI	Maître de conférences-A, E.S.I, Alger	Invité

Blida, Février 2013

RESUME

Les systèmes décisionnels ont émergé dans la dernière décennie autour de l'utilisation des entrepôts de données (*Data warehouses*) et l'analyse en ligne (OLAP, *On-Line Analytical Processing*). Bien que ces technologies gagnent de plus en plus en popularité au sein des organisations, seuls 20% des informations extraites des données, dites structurées, peuvent être traitées par un système OLAP. En effet, les 80% restants de l'information sont contenus dans des documents non structurés ou semi-structurés. Il est admis que les textes constituent l'essentiel de ces données, car il est le moyen le plus répandu pour exprimer les informations et les connaissances. Avec la croissance explosive des données textuelles, aussi bien dans les organisations que sur le web, il devient nécessaire d'aller au-delà de l'analyse en ligne des données structurées, pour prendre en charge également les données textuelles, non structurées, et couvrir ainsi les 100% des données d'un système d'information. Or, la prise en charge des données textuelles par les systèmes décisionnels constitue un défi pour deux principaux problèmes. Le premier est relatif à l'intégration et au stockage des informations issues de documents hétérogènes. Le second problème consiste à déterminer les informations à extraire des documents textuels pour servir aux différents processus de restitution, notamment l'analyse OLAP. L'agrégation des données textuelles constitue l'un des challenges que doit relever les processus OLAP. En effet, avec les outils OLAP classiques, il est impossible d'agréger des données textuelles selon des fonctions arithmétiques. L'environnement OLAP de données textuelles, a besoin de nouvelles techniques d'agrégation pour ce type de données.

Nos propositions se résument en quatre fonctions de *summarization*, utilisant des techniques du domaine de la fouille de texte. L'objectif de la *summarization* est d'exploiter la mesure textuelle du cube de textes pour résumer le contenu des documents textuels sous une nouvelle forme appréhendable par le décideur. Ces fonctions sont : *Classes*, *Clusters*, *Top_Keyphrases* et *Summary*. La fonction *Classes* résume les documents textuels en offrant au décideur un ensemble d'agrégats sous forme de *n* classes. Pour obtenir ces classes, nous utilisons la technique de catégorisation de texte par le contenu. La fonction *Clusters* résume les documents textuels en *k* groupes, pour cela la classification non supervisée (*clustering*) est utilisée. La fonction *Top_Keyphrases* permet l'agrégation d'un ensemble de documents en les thèmes les plus représentatifs. Enfin, la fonction *Summary* fournit un résumé sous forme des phrases les plus pertinentes extraites des documents à agréger. Nos propositions ont pour objectif, d'aller au-delà des analyses quantitatives sur les documents textuels où un comptage des instances des documents est effectué, et permettre des analyses qualitatives et sémantiques sur le contenu textuel des documents.

Mots-clés : cube de textes, OLAP, données textuelles, fouille de texte, *summarization* de documents.

ملخص

نظم صنع القرار ظهرت في العقد الماضي حول استخدام مستودعات البيانات (*Data warehouses*) والمعالجة التحليلية الفورية (*OnLine Analytical Processing,* OLAP). وعلى الرغم من ان هذه التكنولوجيات يكسبون اكثر شعبية في داخل المنظمات سوى 20 في المائة من المعلومات المستمدة من البيانات، المسماة مهيكلة، يمكن معالجتها عن طريق نظام OLAP. والواقع ان 80 في المائة المتبقية من المعلومات واردة في الوثائق غير المهيكلة وشبه المهيكلة. ومن المسلم به ان النصوص تشكل الجزء الاكبر من هذه البيانات، لا نها اكثر الوسائل شيوعا في التعبير عن المعلومات والمعارف. ومع النمو المتفجر للبيانات النصية، سواء في المنظمات أو على شبكة الانترنت، اصبح من الضروري الذهاب الى ابعد من التحليل الفوري للبيانات المهيكلة، ودعم ايضا البيانات النصية غير المهيكلة، وبالتالي تغطية 100 في المائة من البيانات من نظام المعلومات. ومع ذلك، فإن دعم البيانات النصية لنظم اتخاذ القرار يشكل تحديا لمشكلتين أساسيتين. الأولى تتعلق بإدماج وتخزين المعلومات من وثائق غير متجانسة. والمشكلة الثانية هي تحديد المعلومات التي يجب استخلاصها من الوثائق والنصوص لخدمة مختلف عمليات الاسترداد، بما في ذلك تحليل OLAP. عملية تجميع البيانات النصية هي واحدة من التحديات التي تواجه عملية OLAP. في الواقع، مع أدوات OLAP التقليدية، فانه من المستحيل تجميع بيانات النص وفقا للوظائف الحسابية. بيئة OLAP للبيانات النصية، تحتاج تقنيات تجميع جديدة لهذا النوع من البيانات.

وفيما يلي موجز مقترحاتنا في أربع وظائف تلخيص، وذلك باستخدام تقنيات من مجال التنقيب في البيانات النصية. الغرض من التلخيص، هو استغلال المقاييس النصية في *Text cube* لتلخيص محتويات وثائق النص في شكل جديد مفهوم من قبل صانع القرار. الوظائف الاربع هي: *Classes*، *Clusters*، *Top_Keyphrases* و*Summary*. تلخص وظيفة *Classes* الوثائق النصية من خلال تقديم صانع القرار مجموعة من المجاميع في شكل طبقات. للحصول على هذه الطبقات، نستخدم تقنية *Text categorization*. وظيفة *Clusters* تلخص الوثائق النصية في مجموعة كتل، وذلك باستعمال تقنية *Clustering*. وظيفة *Top_Keyphrases* تتيح تجميع مجموعة من الوثائق في الموضوعات الأكثر تمثيلا. وأخيرا، فإن *Summary* وظيفة توفر ملخص في شكل الجمل الاكثر اهمية من الوثائق التي سيتم تجميعها. وتهدف مقترحاتنا تجاوز التحليل الكمي للوثائق النصية حيث يتم إجراء تعداد لهذه الوثائق، وتمكينها من تحليلات نوعية ودلالية لمضمون الوثائق النصية.

الكلمات المفتاحية : Text cube، OLAP، بيانات نصية، التنقيب في البيانات النصية، تلخيص الوثائق.

ABSTRACT

Decision support systems have emerged in the last decade around the use of data warehouses and On-Line Analytical Processing (OLAP). Although these technologies are gaining increasing popularity within organizations, only 20% of information extracted from structured data, can be treated by OLAP systems. Indeed, the remaining 80% of the information are contained in unstructured and semi-structured documents. It is acknowledged that the text makes up the main part of the unstructured data because it is the most common way to express information and knowledge. With the explosive growth of text data, both in the organizations as well as on the web, it becomes necessary to go beyond online analysis of structured data, to support also unstructured textual data, and cover 100% of information system data. However, the management of textual data for decision support systems is a challenge for two main problems. The first relates to the integration and storage of information from heterogeneous documents. The second problem is to determine the information to be extracted from text documents to serve the various processes of reproduction, including OLAP analysis. The aggregation of textual data is one of the challenges facing the OLAP processes. Indeed, with traditional OLAP tools, it is impossible to aggregate text data according to arithmetic functions. The OLAP environment for textual data needs new aggregation techniques for this type of data.

Our proposals are summarized into four summarization functions, using techniques from the field of text mining. The purpose of summarization is to exploit the textual measure of a text cube to summarize document's content in a novel way that is apprehensible by decision-makers. These functions are: *Classes*, *Clusters*, *Top_Keyphrases* and *Summary*. The *classes* function, summarizes textual documents by offering the decision maker a set of aggregates in the form of *n* classes. To obtain these classes, we use the technique of text categorization. The *Clusters* function summarizes the textual documents into *k* clusters, for this purpose the technique of clustering is used. The *Top_Keyphrases* function allows the aggregation of a set of documents into the most representative topics. Finally, the *Summary* function provides a summary formed of the most relevant sentences from the documents to be aggregated. Our proposals aim to go beyond quantitative analysis on the textual documents where a count of document instances is made and enable semantic and qualitative analyzes on the textual content of documents.

Keywords: text cube, OLAP, textual data, text mining, document summarization.

REMERCIEMENTS

Tout d'abord je remercie DIEU tout puissant pour m'avoir donné la force et la patience d'accomplir ce modeste travail « … Et t'a enseigné ce que tu ne savais pas. Et la grâce d'Allah sur toi est immense… » **Sourate Les femmes verset 113**. Ce fût une grande opportunité de faire ce magistère, de découvrir le monde de la recherche.

En second lieu je reste profondément reconnaissante à mes encadrants Docteur Nadjia Benblidia et Professeur Omar Boussaid pour la confiance qu'ils m'ont accordé et pour la grande liberté d'action dont j'ai pu bénéficier. En dehors du cadre scientifique, je les remercie très sincèrement pour leurs qualités humaines qui m'ont permis de remonter la pente dans les moments les plus difficiles.

Je remercie Saliha Oukid, Maître de conférences habilité à l'Université de Blida, pour l'honneur qu'elle me fait de présider le jury de ce mémoire. Je remercie également les autres membres pour avoir accepté de participer à mon jury. Je remercie tout particulièrement mon ancien enseignant de structures de données Djamel Eddine Zegour, Professeur à l'E.S.I, pour l'honneur qu'il me fait d'être membre de ce jury.

Je remercie Yacine Bouzegza, expert en recrutement, pour les informations précieuses qu'il m'a fournies sur le domaine du recrutement. Que Yacine Atif, Professeur à United Arab Emirates University, trouve ici mes remerciements les plus sincères pour ses encouragements et ses informations plus que précieuses sur le monde de la recherche et des conférences. Je ne peux oublier de remercier Salim Boutarouk, Directeur de l'I.N.S.F.P de Khemis Miliana et Yacine Mehabil, Directeur des études spécialisées du même institut, mes supérieurs hiérarchiques pour avoir aménagé mes horaires de travail pour que ce mémoire se déroule dans des conditions favorables, sans pour autant faillir à ma tâche. Merci à mon collègue Ahmed Temmam, le matheux pour son aide et sa collaboration dans la formalisation et la validation mathématique des fonctions d'agrégation.

Je remercie mon frère Mohamed d'avoir cru en mes capacités. Il a tout fait pour me convaincre de faire magistère. Et j'ai tout fait pour lui faire oublier cette idée, mais la volonté de DIEU a été plus forte que tout. Je remercie mon père pour ses prières continuelles pour moi, pour ses encouragements, ses sacrifices. Des remerciements spéciaux vont à mon frère Djillali et ma belle-sœur Nadia pour tout ce qu'ils ont fait pour moi. Ils ont toujours été là à m'offrir un environnement agréable, à me servir, dans mes moments les plus difficiles. Que tous les membres de ma famille trouvent ici ma reconnaissance pour leurs encouragements et leur patience.

« Ô Seigneur, si vous me dotez de ce que j'aime, faites en sorte qu'il soit une force pour que j'accomplisse ce que vous aimez… » **Prophète Mohammad (bénédictions et paix sur lui).**

TABLE DES MATIÈRES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION

Dans un contexte concurrentiel en constante évolution, les entreprises sont soumises à s'adapter en permanence et à être de plus en plus réactives aux variations des marchés et besoins des clients. Pour cela, il convient à ces entreprises de bien comprendre son environnement, d'ajuster ses interactions avec lui en faisant les meilleurs choix de cibles et d'actions. Face à ce besoin, les systèmes décisionnels ont été mis en place dans les entreprises. Ils fournissent aux décideurs des informations pour définir la stratégie, piloter les opérations et analyser les résultats des données du système d'information.

Les systèmes décisionnels ont émergé dans la dernière décennie autour de l'utilisation des entrepôts de données (*Data warehouses*) et l'analyse en ligne (OLAP : *On-Line Analytical Processing*). Un entrepôt de données est une collection de données orientées sujet, intégrées, non volatiles, historisées et organisées pour supporter un processus d'aide à la décision [35]. Ce dernier peut être amélioré par des systèmes d'analyse en ligne OLAP [16] qui ont pour but de fournir une réponse rapide et interactive à des requêtes analytiques de nature multidimensionnelle. Les technologies OLAP permettent aux responsables de la stratégie d'une entreprise, les décideurs, d'avoir une vision synthétique de leurs activités les aidant ainsi à orienter leurs décisions.

Bien que ces technologies (les entrepôts de données et OLAP) gagnent de plus en plus en popularité au sein des organisations, elles ne couvrent pas entièrement la portée des systèmes décisionnels. Il est estimé qu'environ 80 à 85 pour cent des données pertinentes sur des systèmes informatiques modernes ne sont pas structurées [32]. Pour les systèmes décisionnels d'une entreprise, seuls 20% des informations extraites des données, dites structurées et stockées dans des bases de données relationnelles, peuvent être traitées par un système OLAP. Pour Sullivan [76], la plupart des efforts n'ont touché que la partie visible de l'iceberg des flux informationnels. En effet, les 80% restants de l'information sont contenus dans des documents non structurés ou semi-structurés, appelés *données* ou *objets complexes* [19]. L'une des caractéristiques de ces derniers est qu'ils peuvent être représentés sous divers formats : textes, sons, images, vidéos, etc. [19]. Ces documents sont principalement des documents texte, ce type de données étant le plus répandu pour exprimer les informations et les connaissances.

Les données textuelles ne sont pas structurées, de plus la diversité de leurs sources d'information est extrêmement large, et inclut des documents techniques, des e-mails, des articles de presse, des actes de conférences, des commentaires de clients, etc. Ces données représentent la capitalisation des connaissances dans les organisations, au même titre que les transactions dans les bases de données. Elles sont partout dans l'organisation et représentent à la fois un challenge et une opportunité pour être utilisées dans le processus d'aide à la décision.

Problématique

Avec la croissance explosive des données textuelles, aussi bien dans les organisations que sur le web, il devient nécessaire d'aller au-delà de l'analyse en ligne des données structurées, pour prendre en charge également les données textuelles, non structurées, et couvrir ainsi les 100% des données d'un système d'information [76]. Cependant, un fossé existe entre les sytèmes de données structurées et les systèmes de données non structurées [36]. Dans son nouveau paradigme de data warehouse 2.0, Inmon [35] définit les caractéristiques architecturales pour une nouvelle génération d'entrepôts de données baptisé DW 2.0. Les auteurs préconisent qu'un entrepôt de données moderne doit contenir les données structurées ainsi que les données non structurées. Ces dernières qui sont principalement textuelles, contiennent une mine d'informations dont le déverrouillage constitue un véritable challenge [35].

Deux principaux problèmes sont à considérer : Le premier est relatif à l'intégration et au stockage des informations issues de documents hétérogènes. Cette hétérogénéité est aussi bien structurelle que sémantique. De plus, le contenu de documents textuels étant peu structuré, ce qui explique la difficulté de leur intégration dans les systèmes décisionnels. La tendance actuelle pour résoudre ce problème consiste à utiliser le standard XML[1]. Des travaux ont montré que XML peut être une solution permettant de représenter et d'intégrer des documents peu ou pas structurés au sein des systèmes décisionnels [10, 11, 12, 57]. XML, une norme du W3C[2], est considéré comme un standard dans la description et l'échange des données. Il représente les données de façon semi-structurée. Aussi, sa capacité d'auto-description et sa structure arborescente donnent à ce formalisme une grande flexibilité et une puissance suffisante pour décrire des données complexes, hétérogènes et provenant de sources éparpillées.

[1] Extensible Markup Language (http://www.w3.org/XML/)
[2] http://www.w3.org/

Le second problème consiste à déterminer, les informations à extraire des documents textuels pour servir aux différents processus de restitution, notamment l'analyse OLAP. Se pose alors les questions suivantes : Est-il possible d'analyser des données textuelles représentées à l'aide d'un schéma multidimensionnel ? Doit-on faire évoluer les modèles de schémas en étoiles existants ? Comment peut-on agréger les données textuelles ? Ce dernier problème constitue l'un des challenges que doit relever les processus OLAP. En effet, avec les outils OLAP classiques, il est impossible d'agréger les données textuelles selon des fonctions telles que la somme ou la moyenne. L'environnement OLAP de données textuelles, appelé aussi *Text OLAP* [55], a besoin d'outils appropriés et de nouvelles techniques d'agrégation pour ce type de données. La tendance actuelle, vise à combiner l'OLAP avec une des principales technologies manipulant les données textuelles, la Recherche d'Information (RI), et/ou la Fouille de Texte (*Text mining*, TM). En effet, avec le développement des méthodes statistiques et linguistiques, nous disposons aujourd'hui d'outils qui nous permettent d'analyser le texte pour en extraire les informations clés, catégoriser les documents, les indexer par thème ainsi que par mots clés, résumer automatiquement le texte, et grouper les documents similaires (*clustering*). Ces outils représentent les solutions à l'intégration des données textes à l'infrastructure des systèmes décisionnels. Ils représentent aussi une solution envisageable pour une analyse qualitative, sémantiquement riche des données textuelles dans un environnement OLAP.

Objectifs et contributions

L'aspect proéminent, dans l'entreposage de données et l'analyse en ligne OLAP, est l'agrégation des données. Cette dernière peut s'envisager, pour les données textuelles, en s'inspirant de techniques bien maitrisées des domaines de la recherche d'information ou de la fouille de texte.

Nous proposons dans ce mémoire quatre fonctions pour l'agrégation des données textuelles dans les cubes de textes. Un cube de textes est un cube de données utilisé pour la modélisation et l'analyse multidimensionnelle de documents textuels. La mesure utilisée dans un cube de textes est une mesure textuelle qui modélise le contenu (ou une partie du contenu) du document à analyser. En exploitant cette mesure textuelle dans le cube de textes, nous proposons la *summarization* des documents textuels. Par *summarization* de documents textuels nous entendons, un processus qui permet de résumer leur contenu sous une nouvelle forme permettant de les analyser afin d'en extraire de la connaissance. Pour

atteindre ce but, nous proposons les fonctions d'agrégation *Classes*, *Clusters*, *Top_Keyphrases* ou *Summary*.

- La fonction *Classes* résume les documents textuels en offrant au décideur un ensemble d'agrégats sous forme de *n* classes. Pour obtenir ces classes, nous utilisons la technique de la catégorisation de texte par le contenu.
- La fonction *Clusters* résume les documents textuels en *k* clusters, pour cela une technique de classification non supervisée (*clustering*) est utilisée.
- La fonction *Top_Keyphrases* permet l'agrégation d'un ensemble de documents en les thèmes les plus représentatifs.
- Enfin, la fonction *Summary* fournit un résumé sous forme des phrases les plus pertinentes extraites des documents à agréger.

Nos propositions ont pour objectifs, tout d'abord d'aller au-delà des analyses quantitatives sur les documents textuels où un comptage des instances des documents est effectué, et permettre des analyses qualitatives et sémantiques sur le contenu textuel des documents.

De plus, la richesse de nos propositions réside dans le fait que ces quatres fonctions permettront de couvrir une panoplie de scénarios d'applications. Nous présentons dans ce mémoire un ensemble de scénarios pour un cas d'étude du domaine des ressources humaines celui de l'analyse des CVs.

Organisation du mémoire

Le reste de ce mémoire est structuré selon deux parties. Dans la première partie nous présentons un état de l'art sous forme de trois chapitres. Le chapitre 1 de cette première partie est consacré aux concepts relatifs au contexte de notre étude. Nous poursuivons, dans le chapitre 2, en présentant les données textuelles dans l'organisation, les problèmes liés à leur gestion, ainsi que les approches de *summarization* de ces données. Le chapitre 3 fait l'objet des différents travaux qui proposent l'intégration des documents textuels dans les systèmes OLAP. Nous terminons ce chapitre, ainsi que cette partie, par une synthèse dans laquelle nous positionnons nos contributions.

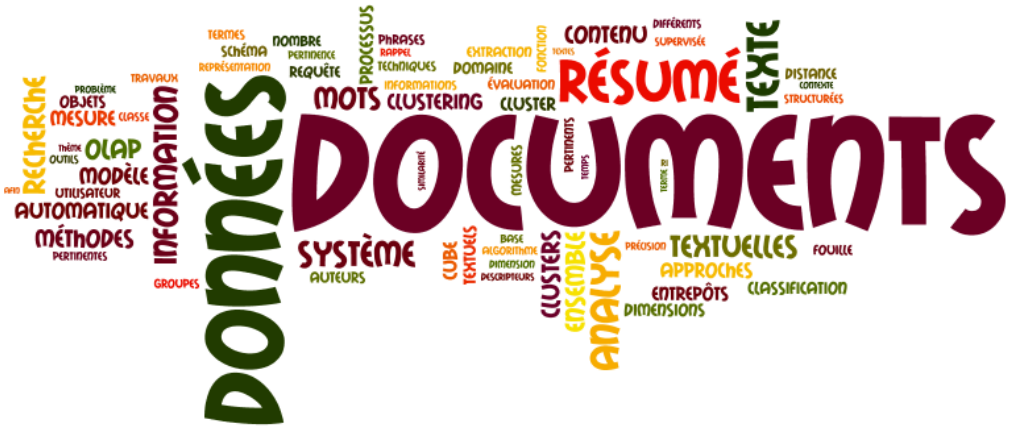
La deuxième partie fait l'objet de nos contributions. Nous abordons dans le chapitre 4 la modélisation multidimensionnelle de documents textuels. Nous présentons deux approches, celle par extension de schéma en étoile ainsi que le modèle en galaxie proposé dans [63]. Nous abordons ensuite l'approche adoptée ainsi que la présentation d'un cas

d'application réel, celui de l'analyse en ligne des CVs. Le modèle multidimensionnel pour ce dernier cas est aussi présenté. Dans le chapitre 5, nous présentons les quatre fonctions d'agrégations proposées. Pour chacune d'elles, nous présentons le principe par le biais d'un scénario d'analyse et nous décrivons un cadre formel ainsi qu'un cadre théorique. Nous évoquons dans le chapitre 6 nos expérimentations pour deux fonctions d'agrégation, et commentons les résultats obtenus sur un jeu de données que nous présentons au préalable. Nous discutons aussi dans ce chapitre, des performances computationnelles pour ces fonctions ainsi que les possibilités d'optimisations.

Nous terminons ce mémoire par une conclusion générale qui présente les apports de notre travail ainsi que les perspectives envisagées.

PREMIÈRE PARTIE : ÉTAT DE L'ART

Résumé. *Le cœur de notre travail est l'agrégation des données textuelles dans les cubes de textes. Afin de mieux cerner cette problématique, nous avons mené une étude bibliographique dans laquelle, nous allons dans le chapitre 2 donner les concepts de base liés au contexte de notre étude, les systèmes décisionnels, les entrepôts de données et les systèmes OLAP, nous abordons aussi les entrepôts de documents textuels. Nous discutons dans le chapitre 3 des problèmes liés à la gestion des données textuelles. Nous présentons aussi quelques approches de summarization de ces dernières. Le chapitre 4 présente un panorama des travaux ayant abordé l'analyse en ligne des données textuelles. Dans la dernière section de ce chapitre nous concluons ce parcours bibliographique par une synthèse et nous positionnons notre travail.*



Tag cloud réalisé en utilisant http://www.wordle.net/

CHAPITRE 1
LES SYSTÈMES DÉCISIONNELS

1.1 Introduction

Ce mémoire s'inscrit dans le domaine des systèmes décisionnels, tout particulièrement les systèmes OLAP. Il est donc naturel de commencer par présenter les concepts liés au contexte de notre étude. Ainsi, ce chapitre est organisé de la manière suivante. Dans la section 1.2 nous définissons et donnons l'architecture d'un système décisionnel. La section 1.3 aborde les concepts d'entrepôts et magasins de données. Les principes des systèmes OLAP sont abordés en section 1.4. Enfin, dans la dernière section, nous introduisons les entrepôts de documents textuels.

1.2 Le système décisionnel

Le système décisionnel est défini [67] comme étant le système dédié au support de la prise de décision (pilotage) au sein d'une organisation. Il regroupe l'ensemble des outils informatiques (matériels et logiciels) permettant :

- d'extraire, de transformer et de charger les données opérationnelles,
- de constituer un ou des espaces de stockage de données décisionnelles,
- de manipuler ces données au travers d'outils d'analyse ou d'interrogation destinés au pilotage des organisations.

Dans la dernière décennie, les systèmes décisionnels ont constitué une coalescence autour des entrepôts de données et des systèmes OLAP. Ces technologies sont nées de la nécessité d'isoler et de tenir à disposition les données pertinentes issues des systèmes d'information de production pour une analyse approfondie. Avec ces technologies, le système décisionnel est organisé autour d'une architecture comportant principalement quatre parties :

- Partie extraction, transformation et alimentation (ou ETL, *Extracting Transformating Loading*) des données issues de sources hétérogènes pour alimenter le système décisionnel. Ces sources peuvent être internes à l'entreprise (données opérationnelles) ou externes.

- L'entrepôt de données comme espace de stockage centralisé des sources de données. Les données dans l'entrepôt représentent une vision horizontale des activités de l'entreprise.
- Les bases de données multidimensionnelles (ou magasins de données) où une partie des données (thème d'analyse particulier) de l'entrepôt sont structurées via une modélisation multidimensionnelle [15] facilitant leur interrogation et analyse. Contrairement à l'entrepôt de données, le magasin de données représente une vision verticale des données de l'entreprise.
- L'analyse multidimensionnelle et la restitution où les données issues des magasins de données sont restituées sous une forme adaptée aux utilisateurs.

La figure ci-dessous illustre les quatre parties de l'architecture d'un système décisionnel.

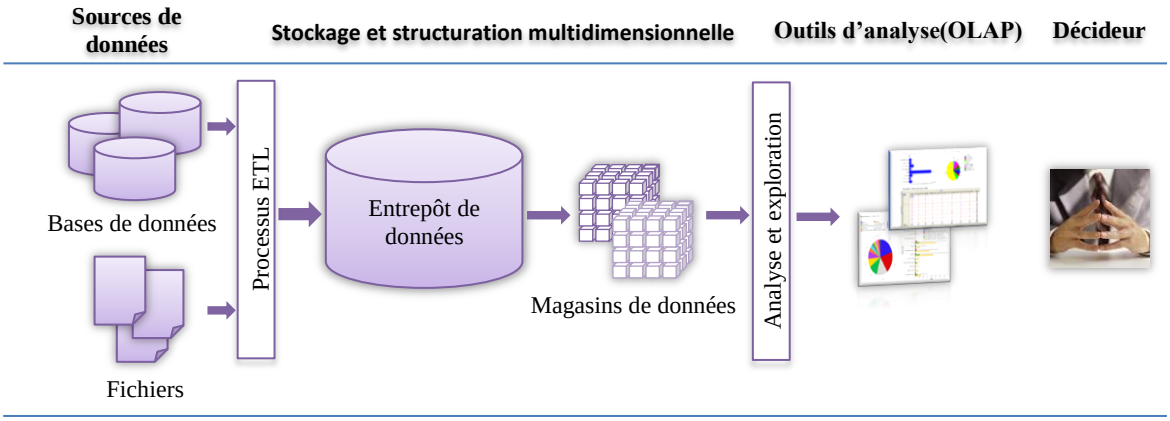


Figure 1.1 : Architecture d'un système décisionnel [67]

1.3 Entrepôts et magasin de données

Un entrepôt de données (*data warehouse*) est défini comme étant «une collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour le support d'un processus d'aide à la décision» [35]. Cette définition met l'accent sur les caractéristiques suivantes :

- ***Orientées sujet :*** Les données de l'entrepôt sont organisées par sujet plutôt que par application. Cette orientation sujet permet de mettre en avant les indicateurs de performance pour chaque thème d'analyse. Par exemple, une chaîne de magasins d'alimentation organise les données de son entrepôt par rapport aux ventes qui ont été réalisées par produit et par magasin, au cours d'un certain temps.

- ***Intégrées :*** Les données provenant de différentes sources doivent être intégrées, avant leur stockage dans l'entrepôt de données. L'intégration consiste à convertir, reformater, et nettoyer les données des systèmes de production afin d'assurer la cohérence de l'information.
- ***Non volatiles :*** À la différence des données opérationnelles, celles de l'entrepôt sont permanentes et ne peuvent pas être modifiées. Le rafraîchissement de l'entrepôt, consiste à ajouter de nouvelles données, sans modifier ou perdre celles qui existent.
- ***Historisées :*** La prise en compte de l'évolution des données est essentielle pour la prise de décision qui, par exemple, utilise des techniques de prédiction en s'appuyant sur les évolutions passées pour prévoir les évolutions futures.

En plus de sa vocation de stockage, la modélisation d'un entrepôt doit permettre l'analyse de ses données. En effet, les données d'un entrepôt sont sélectionnées pour construire des magasins de données (*data marts*). Un magasin de données est un extrait de l'entrepôt dédié à une activité particulière et organisé selon un modèle adapté aux outils d'analyse et d'interrogation décisionnelle [81]. Un modèle multidimensionnel ou en cube est généralement utilisé pour structurer les données du magasin. Ces modèles sont largement employés pour préparer les données à l'analyse, notamment l'analyse OLAP.

1.4 Les systèmes OLAP (*On-Line Analytical Processing*[4])

Un système OLAP est défini [78] comme un système décisionnel dans lequel les magasins de données suivent une organisation multidimensionnelle des données afin d'assurer un support efficace pour l'analyse OLAP. L'idée principale de cette dernière, est de permettre la navigation interactive à travers les données pour en extraire de la connaissance [2]. Cette approche décisionnelle, peut être vue en un processus en trois étapes comme le montre la figure 1.2. Les sections qui suivent, sont dédiées aux concepts liés à chacune de ces trois étapes.

1.4.1 Le modèle multidimensionnel

Les magasins de données reposent sur un modèle multidimensionnel, permettant l'expression d'analyses en ligne (OLAP). Un modèle multidimensionnel est défini par un

[4] http ://www.olapreport.com/fasmi.htm

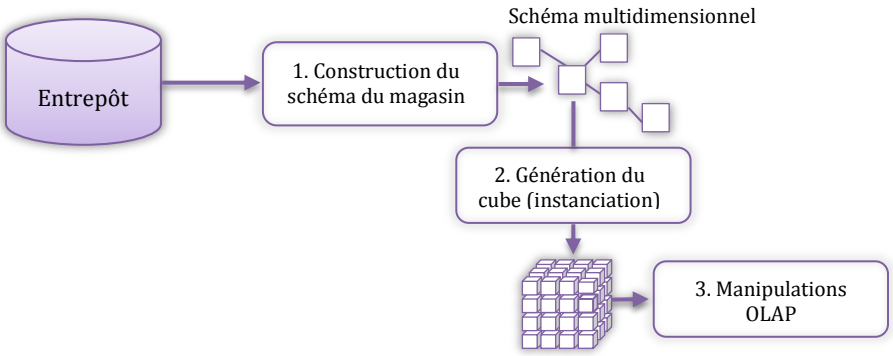


Figure 1.2 : Processus d'analyse OLAP, inspirée de [42]

sujet d'analyse, appelé *fait*. Ce dernier est observé par un ou plusieurs indicateurs d'analyse appelés *mesures***,** selon plusieurs axes d'analyse, appelés *dimensions*. Ces dernières sont composées d'attributs, appelés *paramètres***,** agencés de manière hiérarchique. Par exemple, une dimension temps peut être organisée selon une hiérarchie à quatre niveaux : Mois → Trimestre → Semestre → Année. De telles hiérarchies permettent d'observer des indicateurs selon plusieurs niveaux de granularité et de construire des agrégats à partir des faits. Un modèle composé d'un fait et de ses dimensions est appelé *schéma en étoile* [15]. Un schéma dit *en constellation* est une généralisation du schéma en étoile où plusieurs faits partagent tout ou partie de leurs dimensions [15].

Exemple : la figure 1.3 illustre un exemple de modèle en étoile pour la modélisation de l'analyse des *ventes* en fonction des *produits* et des *magasins* au cours du *temps*. Le sujet est modélisé par le fait *ventes* composé des mesures *quantité* et *montant*. Les axes d'analyses sont modélisés par les dimensions *produits*, *magasins* et *temps*. La dimension *magasins* est caractérisée par trois paramètres *villes*, *pays* et *continents* organisés de façon hiérarchique.

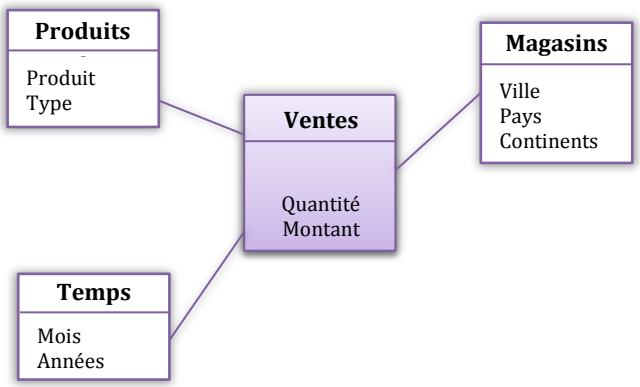


Figure 1.3 : Schéma en étoile pour l'analyse des ventes.

1.4.2 Le cube de données

La représentation multidimensionnelle des données peut aussi être considérée en utilisant la métaphore de *cube* ou d'*hypercube* [27]. Un cube de données est une structure combinant une partie d'un schéma en étoile et ses valeurs associées [67]. Il s'agit d'une instance du modèle multidimensionnel où chaque fait est représenté par une cellule du cube, décrite par un ensemble de valeurs des paramètres des dimensions. Ces dernières représentent les arrêtes du cube.

Exemple : le cube de données associé à l'exemple précédent (figure 1.3) est donné par la figure 1.4. Chaque cellule du cube contient les valeurs des mesures, *quantité* et *montant*, correspondant à la combinaison des modalités des dimensions, *produits*, *magasins* et *temps*.

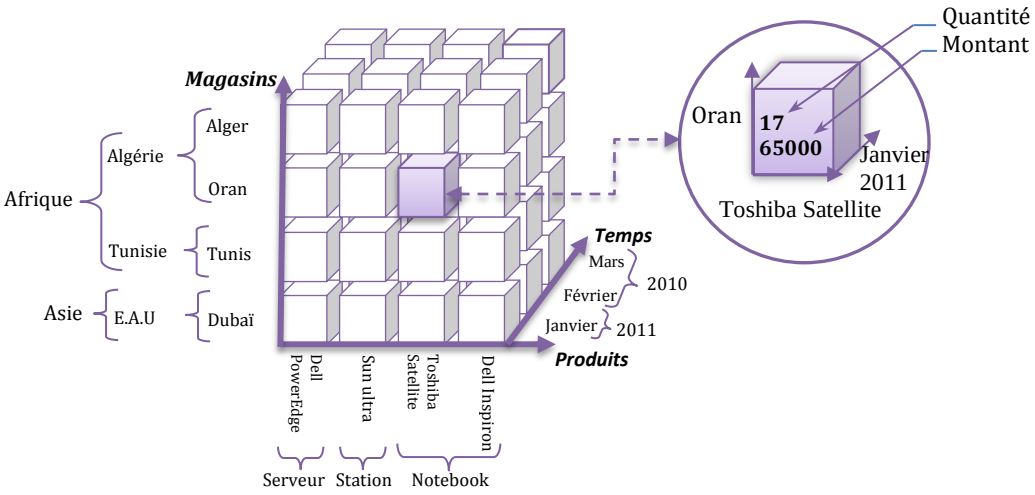


Figure 1.4 : Cube de données pour l'analyse des ventes.

1.4.3 Opérations de manipulation OLAP

Les technologies OLAP reposent sur des outils pour la visualisation, la structuration et l'exploration des cubes de données. Ils fournissent aux utilisateurs des opérateurs pour naviguer dans les données multidimensionnelles afin d'y découvrir des informations pertinentes. Ces opérateurs ont été initialement définis par l'algèbre OLAP, introduite par [16]. Les opérateurs standards sont :

- Les opérations de *forage*, permettent de naviguer au sein d'une hiérarchie de dimension pour analyser les indicateurs d'analyse avec différents niveaux de granularités. Nous trouvons, l'opérateur de forage vers le haut (*roll-up*) qui consiste à remonter d'un niveau dans une hiérarchie de dimension vers un ni-

veau plus agrégé. L'opérateur de forage vers le bas (*drill-down*) consiste quant à lui à descendre dans une hiérarchie de dimension vers un niveau plus détaillé.

- Les opérations de *sélection*, permettent de travailler sur une sous-partie des données du cube. Nous trouvons l'opérateur *slice* qui sélectionne une tranche du cube en exprimant une restriction sur une des valeurs de l'un des axes d'analyse. L'opérateur *dice* sélectionne un sous-cube en exprimant une restriction sur les données d'un indicateur d'analyse.
- Les opérations de *rotation*, permettent d'intervertir les axes d'analyse. L'opérateur *rotate* permet de réorienter la vue multidimensionnelle en tournant le cube.

1.5 Les entrepôts de documents textuels

Les entrepôts de documents textuels sont nés du besoin d'étendre la portée des systèmes décisionnels pour inclure les données textes non structurées. D'après [36], 99% des décisions sont prises sur la base des données structurées alors que le volume des données non structurées est de quatre à cinq fois plus grand que le volume de données structurées. La valeur ajoutée des données textuelles est inestimable pour le processus d'aide à la décision. Cependant, ces données sont hors de portée des systèmes OLAP, par l'absence de moyens adaptés à leur intégration, leur traitement et leur analyse.

Un entrepôt de documents est défini par [67], comme un espace de stockage centralisé d'informations (contenu, structures, métadonnées…) issues des documents sources (hétérogènes en structures et en types) pertinents pour les décideurs. Pour le même auteur, cet entrepôt doit servir de support aux processus de recherche d'information, d'interrogation et d'analyse décisionnelle. La même démarche pour la construction d'un entrepôt de données est nécessaire pour la conception d'un entrepôt de documents [41, 17, 82, 67, 57]. Il s'agit de la phase d'intégration (textual-ETL[5]) des données provenant de sources éparpillées (interne à l'entreprise et sur le web), la phase de structuration et enfin l'analyse OLAP. Cependant, des problèmes sont liés à ce processus d'entreposage, des challenges doivent être relevés.

En amont du processus d'analyse et d'exploitation des documents textuels, leur intégration demeure encore une des difficultés majeures, due principalement au problème d'hétérogénéité de ces documents. Cette hétérogénéité se trouve à deux niveaux :

[5] http://textualetl.com/

- Au niveau structurel, les documents peuvent être de structures différentes voir même des documents non structurés ou peu structurés.
- Au niveau sémantique, les documents peuvent concerner des domaines très variés et en diverses langues. En l'absence d'outils appropriés, l'utilisateur doit fouiller et parcourir le contenu des documents pour trouver l'information pertinente.

Certains travaux [42, 49, 67] se sont intéressés à la problématique de l'hétérogénéité (structurelle et sémantique) des documents qui sort du cadre de ce mémoire.

Le problème de la modélisation et de l'analyse multidimensionnelle des documents textuels est aussi posé [13, 67, 44]. Le schéma en étoile est-il vraiment adapté à la modélisation multidimensionnelle de documents textuels ? Comment faire l'agrégation des données textuelles dans un environnement OLAP où les fonctions disponibles sont principalement arithmétiques ? À ces questions, nous tenterons d'apporter des éléments de réponses dans la deuxième partie de ce mémoire.

1.6 Conclusion

Nous avons présenté dans ce chapitre, les concepts de base liés au contexte de notre étude. Nous avons défini ce que c'est qu'un système décisionnel. Ce dernier repose sur deux technologies émergentes, les entrepôts de données et l'analyse en ligne OLAP, que nous avons présenté aussi. Finalement, nous avons introduit le concept d'entrepôt de documents textuels ainsi que les problématiques sous-jacentes. Ces problématiques émanent principalement de la nature et la spécificité des données textuelles.

Nous poursuivons donc ce parcours bibliographique par aborder, dans le prochain chapitre, les problèmes liés aux données textuelles et comment résumer ces données pour faciliter leur fouille dans le but d'en extraire de la connaissance.

CHAPITRE 2
APPROCHES DE *SUMMARIZATION* DES DONNÉES TEXTUELLES

2.1 Introduction

Il est dit, et cela pendant des décennies, que l'information est devenue de plus en plus disponible. Cependant, ce n'est que récemment que cette information semble être accessible au format numérique[6]. L'être humain a atteint une capacité impressionnante pour stocker les données. D'après Fayyad et al. [22], la capacité de stockage de ces dernières a doublé tous les neuf mois pendant au moins une décennie. Cependant, la capacité humaine à traiter les données recueillies n'est pas si impressionnante. En effet, l'être humain n'est plus capable d'analyser ou d'appréhender ces informations dans leur globalité. Cette situation est qualifiée de « *data rich but information poor* » [30].

En plus de l'immense quantité de données disponibles, celles-ci ne sont pas seulement numériques ou symboliques, mais elles peuvent être représentées dans des formats différents, provenir de sources diverses ou avoir une sémantique différente [13]. De telles données sont désignées par les termes de données complexes [13]. Ces dernières peuvent être structurées, semi-structurées ou non structurées. On dit qu'elles sont multi-structures. Elles peuvent aussi être multi-format, donc de différents types : image, son, vidéo, texte. Le volume des données non structurées, est de quatre à cinq fois plus grand que les données structurées [36]. Il est admis que les textes constituent l'essentiel de ces données, soit 80%, car le texte est le moyen le plus répandu pour exprimer les informations et les connaissances. Ces données sont contenues dans des documents pouvant provenir de sources diverses.

Rappelons que, nous nous intéressons dans ce mémoire à la problématique de *summarization* des données textuelles dans le but d'en faciliter leur exploitation dans le contexte d'un processus de prise de décision, notamment dans un environnement OLAP. Par *summarization* de documents textuels nous entendons, un processus qui permet de résumer leur contenu sous une nouvelle forme permettant de les analyser afin d'en extraire de la connaissance. Pour atteindre ce but nous nous sommes inspirés de technologies qui mani-

[6] Certaines études ont tenté de chiffrer la croissance de l'information au format numérique, dont la plus citée a été menée à l'Université de Berkeley en 2003 : How much information ? http://www.sims.berkeley.edu/how-much-info-2003.

pulent les données textuelles telles que la recherche d'information, la fouille de texte et le domaine du résumé automatique de texte. Ce chapitre vise à présenter quelques approches issues de ces domaines. Il est structuré comme suit. Nous commençons en section 2.2 par situer les données textuelles dans l'organisation. Nous abordons dans la section 2.3, la problématique liée à la gestion des documents textuels. Nous présentons ensuite, les deux technologies majeures qui manipulent les données textuelles, la recherche d'information (RI) en section 2.4 et la fouille de tex*te (Text mining*) en section 2.5. Le domaine du résumé automatique de texte est abordé en section 2.6. Nous concluons ce chapitre en section 2.7 où nous discutons d'une éventuelle alliance de l'OLAP avec les technologies présentées.

2.2 Les données textuelles dans l'organisation

Les données, dans les organisations existent sous deux grandes catégories : les données structurées et les données non structurées. Les données structurées sont des données de type numérique, générées par les transactions des activités de l'entreprise et sont généralement stockées dans les bases de données et les tableurs. La deuxième catégorie de données qu'on trouve dans les organisations, est celle des données non structurées. Il est estimé que 80 à 85 pour cent des données pertinentes qui existent dans les organisations ne sont pas structurées [32]. Ces données non structurées se divisent en données textuelles et en données non textuelles. Ces dernières incluent le son, la vidéo, les images, etc. Les données textuelles sont les plus répandues et proviennent de sources diverses telles que les e-mails, les rapports, les contrats, les articles scientifiques, les articles de news, les pages web, etc.

Les données textuelles sont disponibles dans les documents électroniques. Un document électronique présente un *contenu*, une *structure logique* qui permet d'identifier les granules d'information du document, une *structure physique* qui décrit le format de restitution physique du document sur support (papier, écran) et les *attributs externes* ou *identité* du document, qui permettent de caractériser sans équivoque un document (date de création, format, etc.) [67]. Ces documents sont stockés généralement sur des systèmes de gestion de contenu, dans des bases de données texte mais peuvent également provenir du web[7].

[7] Selon une étude réalisée par S. Feldman en 2008: What are people searching for and where are they looking? 62% des répondants affirment que le Web constitue leur première source d'information.

2.3 Problématique liée aux données textuelles

La croissance explosive des données textuelles dans les réseaux des entreprises et sur le web, a engendré un problème bien connu celui de la surcharge d'information. Cependant, ce qui est moins connu, c'est le degré auquel cette explosion des données a consommé de ressources coûteuses aussi bien humaines que matérielles pour les gérer [8, 82]. En effet, les gestionnaires des connaissances en entreprise passent en moyenne entre 15% et 35% de leur temps de travail à chercher (très souvent infructueusement) de l'information[8] sur le web ou à l'intérieur d'un intranet corporatif.

La croissance, de plus en plus rapide des données textuelles, a dépassé de loin la capacité humaine pour l'analyse et la compréhension de ces données. Nous ne disposons pas d'assez de temps ni d'énergie pour tout lire et donc une demande, de plus en plus pressante de techniques et d'outils puissants, est exprimée. Ces outils et techniques devraient permettre un accès intelligent à de grands volumes de données textuelles et un gain de temps dans leur gestion. De plus, et en raison de la richesse de l'information contenue dans les données textuelles, organiser, gérer et extraire de la connaissance de ces données, constitue un challenge.

D'énormes efforts sont déployés pour développer des approches et des techniques permettant de retrouver l'information voulue effectivement et efficacement à partir de vastes collections de données textuelles. Ce qui a donné naissance à des domaines tels que la recherche d'information, la fouille de texte, l'extraction d'information, le résumé automatique de texte, etc. qui offrent tous des outils permettant l'exploitation de ces données. Ces champs de recherche, toujours actifs, représentent, sans aucun doute, la réponse actuelle au problème de la surcharge informationnelle de type textuel. Les sections qui suivent sont dédiées à la présentation des domaines de la recherche d'information (RI), de la fouille de texte et celui du résumé automatique de texte. Les approches qui seront abordées dans ces sections, serviront de base pour résoudre notre problématique de *summarization* des données textuelles, dans un contexte décisionnel.

---

http://www.kmworld.com/Articles/Editorial/Feature/What-are-people-searching-for-and-where-are-they-looking-41012.aspx

[8] Toujours selon la même étude réalisée par S. Feldman en 2008: What are people searching for and where are they looking?

2.4 Recherche d'information

La recherche d'information fût la première réponse au problème de surcharge informationnel de documents textuels. Ce domaine consiste à organiser et à rechercher l'information pertinente dans de grands volumes de documents textuels. La RI trouve ces applications dans les catalogues de bibliothèques en ligne, les systèmes de gestion de documents en ligne ou dans les moteurs de recherche sur le web, etc. Un problème de recherche d'information typique, consiste à trouver les documents pertinents dans une collection de documents en se basant sur une requête utilisateur. Cette dernière est souvent formulée sous forme de mots clés qui décrivent l'information recherchée.

Pour répondre à ce besoin informationnel, nous faisons recours à un système de recherche d'information (ou SRI). Ce dernier met en œuvre, principalement, deux processus pour faire correspondre l'information contenue dans la collection de documents et le besoin d'information exprimé par la requête. Il s'agit principalement du processus de représentation et celui de la recherche :

- Processus de représentation : c'est la transformation du document et de la requête en une représentation numérique, qui reflète leur contenu informationnel. Ce processus de conversion est appelé *indexation*. Le résultat de l'indexation est un descripteur pour chaque document qui contient, généralement, une liste des termes présents dans le document. À ces termes sont associés des poids qui caractérisent le degré de représentativité de ces termes dans le document.
- Processus de recherche : ce processus permet d'associer à une requête l'ensemble des documents jugés pertinents par le système. Ce processus est étroitement lié au modèle de représentation du document et de la requête. Il est basé sur un appariement entre la requête et les descripteurs des documents pour mesurer et évaluer un score de pertinence. Les documents retournés sont classés selon ce score de pertinence. Différents modèles de RI ont été proposés pour mesurer ces scores. Ces modèles sont au centre des travaux de recherche dans ce domaine.

Dans la suite de cette section nous présentons les concepts de base en RI. Nous commençons par évoquer l'indexation en section 2.4.1. La section 2.4.2 est consacrée à la présentation des différentes formes que peuvent prendre les descripteurs utilisés dans le processus d'indexation ainsi que les approches d'extraction de ces descripteurs. La pondération des termes est décrite en section 2.4.3. Nous abordons aussi, l'un des modèles de recherche les

plus utilisés, le modèle vectoriel. Nous terminons la partie consacrée à la RI par l'évaluation des systèmes RI.

2.4.1 L'indexation

L'indexation consiste à analyser le document, lors de l'organisation du fond documentaire afin de produire un ensemble de mots clés, appelés aussi descripteurs, utilisés ultérieurement dans le processus de recherche [83]. Pour un SRI, l'indexation est le plus souvent automatique et consiste en l'extraction des descripteurs, l'élimination des mots vides (en utilisant une liste de mots vides), la pondération des mots avant de créer l'index, etc. Le résultat de l'indexation est un ensemble de termes appelé *langage d'indexation*.

2.4.2 L'extraction des descripteurs

Comment pouvons-nous représenter un document textuel afin de pouvoir au mieux découvrir et exploiter les connaissances qu'il contient ? C'est là, le rôle des descripteurs qui sont aussi appelés *unités élémentaires* (ou *tokens* en anglais). Ils constituent l'information atomique dans la représentation d'un texte. Comme ils représentent le contenu du document, le but de l'extraction des descripteurs est de les choisir de manière à ce que l'index (qui est une représentation condensée du contenu des documents) perde le moins d'information sémantique. Ces descripteurs peuvent prendre l'une des formes suivantes :

- Les *mots simples* : les textes sont transformés en un vecteur de leurs mots simples en éliminant les mots vides (ou mots outils de la langue, tels que « de », « un », « le », etc. pour la langue française). L'approche de représentation du document en mots simples est communément connue sous le nom « sac de mots » (*Bag of Words*, *BoW*) [48]
- Les *racines lexicales* (ou *stemm* en anglais) : les textes sont représentés par les racines des mots simples extraits [48]. La racinisation (*stemming*) consiste à trouver la racine des verbes fléchis et à ramener les mots pluriels et/ou féminins au masculin singulier[9]. La racinisation est un traitement étroitement lié à la langue. L'algorithme le plus connu est celui proposé par M. Porter [59], pour la langue an-

[9] Les mots : développe, développé, développeront, développement, seront ramenés à la même forme, « développ ».

glaise. La version française (ainsi que d'autres langues) a été développée dans le cadre du projet Snowball[10].

- Les *lemmes* : les mots extraits du texte sont remplacés par leurs formes d'origine. La lemmatisation consiste à remplacer les verbes par leurs formes infinitives et les noms par leurs formes singulières. Pour cela, une analyse grammaticale est nécessaire. L'algorithme le plus connu pour l'extraction des lemmes est *TreeTagger*[11] [71] pour différentes langues (anglaise, française, etc.).
- Les *n-grammes* de caractères, qui sont une représentation du texte en séquences de *n* caractères consécutifs. Les *n-grammes* de mots représentent le texte en séquences de *n* mots consécutifs. Une représentation en bi-grammes, tri-grammes, est fréquemment utilisée [58].
- Les *concepts* qui sont des expressions contenant un ou plusieurs mots. Ces concepts peuvent être écrits de manière libre par l'utilisateur ou choisis dans une liste de concepts (on parle de vocabulaire contrôlé). Cette liste peut être décrite dans un thésaurus, une ontologie, une hiérarchie de concepts, etc. [6].
- Les *groupes de mots* (ou *phrases* en anglais), sont définis comme des mots qui, pris ensemble représentent les principaux thèmes discutés dans un document [52]. Ces groupes de mots sont plus riches sémantiquement que les mots, qui les composent, pris séparément. Ainsi, l'expression « recherche d'information » est plus précise que « recherche » et « information » pris séparément. Cet argument a conduit à considérer les groupes de mots comme unités de base, appelé aussi *keyphrases*, dans le langage d'indexation.

Trois approches existent dans la littérature pour l'extraction des descripteurs. Ces approches sont détaillées dans [7] :

- Les approches statistiques : ces approches se basent sur des formules statistiques et sur de simples calculs de fréquence, ce qui les rend simples à mettre en œuvre. Ces approches ne nécessitent aucune connaissance de la langue du corpus, ni du domaine couvert par ces corpus. Les groupes de mots, dans les approches statistiques, sont extraits en utilisant la cooccurrence. En partant de l'hypothèse que des termes (généralement une suite de deux ou trois mots) qui apparaissent ensemble dans le texte sont susceptibles de représenter un concept.

[10] http://www.snowball.tartarus.org
[11] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

- Les approches linguistiques : pour extraire les groupes de mots (appelé collocations en Traitement Automatique de la Langue Naturelle) ces approches se basent sur un étiquetage grammatical des mots des textes, en utilisant des patrons syntaxiques (*templates*). Un exemple de patron syntaxique « Nom-Préposition-Nom », « Adjectif-Nom » ou « Nom-Adjectif » etc. des outils tels que *TreeTagger* peuvent être utilisés.
- Les approches mixtes : ces approches combinent les deux informations : statistique et linguistique pour l'extraction des descripteurs.

2.4.3 Pondération des termes

La pondération est l'opération qui consiste à affecter un poids aux termes de la requête et du document. Le poids d'un terme est la détermination de l'importance de ce terme dans la requête ou le document. C'est l'évaluation de son pouvoir discriminant et son importance dans la description sémantique du contenu d'un document. Bien que des techniques linguistiques existent dans la détermination de ce pouvoir discriminant, la grande majorité des approches se basent sur des techniques statistiques. La plupart des méthodes proposées dans la littérature, combinent deux facteurs pour construite un schéma de pondération :

- Le facteur de *pondération locale* qui mesure la représentativité d'un terme dans le document. Si un terme apparaît souvent dans un même document, alors il représente bien ce document. La fonction de pondération locale la plus utilisée est tf_{ij} (*term frequency*) et qui correspond à la fréquence du terme t_i dans le document d_j.
- Le facteur de *pondération globale*, qui mesure la représentativité du terme par rapport à la collection de documents. Si un terme apparaît dans de nombreux documents il ne permet pas, à lui seul, de déterminer si le document est pertinent ou pas. De ce fait, un poids plus important doit être donné aux termes qui apparaissent moins fréquemment dans la collection. La fonction de pondération globale introduite par [70] est le plus souvent utilisée. Cette fonction, désignée par idf_{ij} (*inverse of document frequency*) est définie par l'inverse de la fréquence des documents dans lesquels le terme t_i apparaît.

La combinaison des pondérations locale et globale, a donné naissance à une métrique référencée sous le nom de *tf-idf* (*term weighting scheme* en anglais) [70]. Ce schéma tend à réaliser un compromis (équilibre) entre *spécificité* et *exhaustivité*. Le premier aspect,

cherche à ce que la représentation du document soit la plus complète possible, alors que le deuxième consiste en une meilleure discrimination entre les représentations des documents pour mieux les distinguer [7].

2.4.4 Les modèles RI

Pour formaliser le processus de recherche, un modèle de recherche d'information est utilisé. Un modèle RI fournit un cadre théorique, basé sur des concepts mathématiques, pour la modélisation de la mesure de pertinence [69, 84]. Nous trouvons les modèles RI qui reposent sur la théorie des ensembles (les modèles booléens), ceux qui reposent sur l'algèbre (les modèles algébriques) et ceux qui utilisent la théorie des probabilités (les modèles probabilistes) [48]. Ces modèles de recherche représentent ce qui diffère le plus entre les SRI. La présentation de tous ces modèles sort du cadre de ce mémoire, nous nous contentons dans la section qui suit de présenter le modèle que nous avons adopté dans nos propositions, le modèle vectoriel (*Vector Space Model*).

Le modèle vectoriel

Dans le modèle vectoriel proposé par G. Salton [68], les documents ainsi que la requête sont représentés par des vecteurs dans un espace engendré par les termes d'indexation. L'espace est de dimension *N*, *N* étant le nombre de descripteurs utilisés pour l'indexation. Dans ce modèle, une pondération dans un document est attribuée à chaque terme de l'espace d'indexation. Chaque document d_j est donc représenté par un vecteur de poids des termes dans l'espace des termes.

$$\vec{d_j} = (w_{1j}, w_{2j}, w_{3j}, \dots, w_{nj}) \tag{2.1}$$

De manière analogue, une requête Q est représentée dans l'espace d'indexation par un vecteur de poids des termes qui composent cette requête.

$$\vec{Q} = (q_1, q_2, q_3 \dots, q_n) \tag{2.2}$$

où w_{ij} est le poids du terme t_i dans le document d_j. Et q_i le poids du terme t_i dans la requête Q. Le schéma de pondération le plus couramment utilisé, est *tf-idf* (voir § 2.4.3) dont l'une de ses variantes est :

$$w_{ij} = tf_{ij} * idf_{ij} = tf_{i,j}.log\frac{N}{n_i} \tag{2.3}$$

où :

- tf_{ij} est la fréquence du terme t_i dans le document d_j ;
- idf_{ij} est l'inverse de la fréquence des documents dans lesquels le terme t_i apparaît ;
- N est le nombre total de documents dans le corpus ;
- n_i le nombre de documents dans lesquels apparaît le terme t_i.

Le processus de recherche consiste à trouver les documents qui sont les plus similaires à la requête. Une mesure de similarité appropriée, est utilisée pour calculer la similarité entre le vecteur requête et le vecteur document. Ainsi, la mesure de similarité permet d'apprécier le degré de pertinence d'un document vis-à-vis d'une requête utilisateur. Les valeurs de la mesure de similarité sont utilisées par le SRI afin de proposer à l'utilisateur des listes ordonnées selon la pertinence des documents. Les principales mesures sont [7]:

- Le produit scalaire :

$$Sim(Q,\ d_j) = \sum_{i=1}^{N} q_i * w_{ij} \tag{2.4}$$

- La mesure de jaccard :

$$Sim(Q,\ d_j) = \frac{\sum_{i=1}^{N} q_i * w_{ij}}{\sum_{i=1}^{N} q_i^2 + \sum_{i=1}^{N} w_{ij}^2 - \sum_{i=1}^{N} q_i * w_{ij}} \tag{2.5}$$

- La mesure cosinus :

$$Sim(Q,\ d_j) = \cos(\vec{Q},\ \vec{d}_j) = \frac{\vec{Q}.\overrightarrow{d_J}}{\|\vec{Q}\|.\|\overrightarrow{d_J}\|} = \frac{\sum_{i=1}^{N} q_i * w_{ij}}{\sqrt{\sum_{i=1}^{N} q_i^2} * \sqrt{\sum_{i=1}^{N} w_{ij}^2}} \tag{2.6}$$

2.4.5 Évaluation d'un système RI : Précision et Rappel

Comme pour tout système, l'évaluation d'un système de recherche d'information constitue une étape primordiale dans l'élaboration d'un modèle de recherche ainsi que pour la comparaison de différents modèles. Un système de recherche d'information idéal tente d'atteindre deux objectifs : retrouver tous les documents pertinents et rejeter tous les documents non pertinents. Dans la pratique, l'ensemble des documents retournés par un SRI contient des documents non pertinents, ce qui génère un bruit documentaire. Aussi, un SRI peut manquer de retourner des documents pertinents, ce qui engendre un silence. La précision et le rappel [83] sont deux mesures qui permettent d'évaluer les performances d'un

SRI. En supposant que, D est l'ensemble des documents de la collection, $Pert$ le sous-ensemble de D contenant tous les documents pertinents pour une requête q et $Retr$ le sous-ensemble de D contenant tous les documents retournés par le SRI. Nous définissons la précision et le rappel comme suit :

- La précision : est le taux qui mesure la capacité du système à rejeter tous les documents non pertinents à une requête. Il est exprimé par le rapport entre l'ensemble des documents pertinents retournés et l'ensemble des documents retournés.

$$Précision = \frac{|Pert \cap Retr|}{|Retr|} \tag{2.7}$$

- Le rappel : est le taux qui mesure la capacité du système à retrouver tous les documents pertinents répondant à une requête. Il est donné par le rapport entre les documents retrouvés pertinents et l'ensemble des documents pertinents de la collection.

$$Rappel = \frac{|Pert \cap Retr|}{|Pert|} \tag{2.8}$$

Un compromis, couramment utilisé, entre Précision et Rappel est la F-score (ou F-mesure) donné par la formule (2.9) [83]. Le paramètre β de la formule (2.9) permet de régler les influences respectives de la précision et du rappel. Ce paramètre est très souvent fixé à 1 pour accorder le même poids à ces deux mesures, formule (2.10). Ce qui revient à la moyenne harmonique entre la précision et le rappel.

$$F - score(\beta) = \frac{(\beta^2 + 1) * précision * rappel}{(\beta^2 * précision) + rappel} \tag{2.9}$$

$$F - score = \frac{2 * précision * rappel}{précision + rappel} \tag{2.10}$$

2.5 Fouille de texte (*Text Mining*)

Les techniques de recherche d'information classiques sont devenues inadéquates. Ceci est dû à la croissance de plus en plus grande de données textuelles, seule une petite fraction des documents, celle des résultats de la recherche, sera pertinente à l'utilisateur [30]. Ce dernier a besoin d'outils pour analyser et extraire les informations utiles des documents. Comparer différents documents, résumer leurs contenus sous une nouvelle forme, etc. Ainsi, la fouille de texte est devenue un thème essentiel qui tente de résoudre le problème de surcharge d'informations textuelles en combinant des techniques de la fouille de

données, de l'apprentissage machine, du traitement automatique de la langue naturelle, de la recherche d'information et de la gestion de connaissances [23].

La fouille de texte est définie [23] comme un processus d'analyse exploratoire qui permet de découvrir des connaissances, des données textuelles par des outils d'analyse. De manière comparable à la fouille de données, la fouille de texte cherche à extraire des informations pertinentes des sources de données grâce à l'identification et l'exploration de tendances intéressantes (inconnues ou cachées). Cependant, pour la fouille de texte, les sources de données sont de grandes collections de documents et ces tendances sont identifiées dans les données non structurées de ces documents. Parmi toutes les techniques de la fouille de texte [23, 30, 84], nous nous intéressons dans cette section aux approches de classification automatique de documents textuels. Nous pensons que cette dernière constitue une manière naturelle et intuitive de résumer les données textuelles sous une nouvelle forme, pouvant être exploité par l'utilisateur dans sa prise de décision. Les sections qui suivent sont dédiées aux approches de classification automatique. Dans la section 2.5.1, nous définissons la classification automatique des données de manière générale et des données textuelles de manière particulière. Comme deux types d'approches existent en classification automatique, la section 2.5.2 est consacrée à la classification supervisée. Dans la section 2.5.3, est abordée la classification non supervisée.

2.5.1 Classification automatique de données

La création de groupes au sein d'un ensemble d'objets est une caractéristique innée de l'être humain. Ainsi, dans la vie de tous les jours, il est habituel de classer les gens en des hommes et des femmes de manière intuitive, de distinguer les choses conformément à des formes géométriques, couleurs, etc. Il est donc naturel de faire appel aux groupes quand il s'agit de structurer, d'organiser ou de résumer un ensemble d'objets.

Un groupe peut être vu comme un ensemble d'éléments qui sont rassemblés en raison d'une relation particulière entre eux. La problématique de constituer ces groupes de manière automatique s'appelle la classification automatique; elle se pose dans de nombreux domaines. Dans les réseaux sociaux, on cherche par exemple, à regrouper les différents membres du réseau pour faire émerger des communautés. En recherche documentaire sous l'hypothèse que des documents ayant des contenus similaires sont pertinents pour les mêmes requêtes; ce qui permet de mieux cibler la recherche, etc.

Deux types d'approches existent en classification automatique. La première approche consiste à faire émerger des groupes au sein d'un ensemble d'éléments sans aucune information *a priori*. Dans ce cas, cette tâche est appelée, *classification par apprentissage non supervisée*, ou en utilisant l'anglicisme *clustering*. Les groupes créés sont appelés *clusters*. L'objectif de cette approche est de découvrir la structure sous-jacente des données pour en extraire de l'information. La seconde approche intervient quand les groupes existent *a priori*, et le problème est de créer un modèle permettant d'assigner des éléments à ces groupes. Dans ce cas, la tâche est appelée *classification par apprentissage supervisée* et les groupes sont appelés des *classes* et possèdent une étiquette qui correspond au nom de la classe. La classification supervisée nécessite cependant, contrairement à la classification non supervisée, un ensemble d'exemples, c'est-à-dire un ensemble d'éléments dont la classe est connue a priori. L'objectif, à partir de ces exemples, est de découvrir un modèle des classes pouvant être généralisé à un ensemble de données plus large sous la forme d'un modèle prédictif. Ce processus comporte deux étapes : une étape de construction du modèle à partir des exemples dont la classe est connue, suivie d'une étape de classement des objets dont la classe est inconnue.

On peut donc dire que, l'apprentissage supervisé est exploité afin de réaliser des tâches de décision ou de prévision là où les approches non supervisées sont souvent employées à des fins exploratoires.

Pour les documents textuels, leur volume de plus en plus croissant, la classification automatique a suscité une plus grande attention. La classification de textes a pour objectif de regrouper les textes similaires, c'est-à-dire thématiquement proches, au sein d'un même ensemble. L'intérêt d'une telle démarche est d'organiser les connaissances de façon à pouvoir effectuer, par la suite, une recherche ou une extraction d'information efficace. Les sections suivantes sont consacrées à la classification supervisée et non supervisée de documents textuels.

2.5.2 Classification supervisée de documents textuels

La classification supervisée de documents, aussi appelée catégorisation automatique de documents, est définie comme étant la tâche d'assigner un document texte à une ou plusieurs classes prédéfinies [72, 18]**.** La catégorisation de texte est l'un des sujets de recherche les plus importants dans le domaine de la fouille de texte. En effet, avec la crois-

sance de plus en plus grande de documents textuels, il devient indispensable d'organiser ces documents dans des classes afin d'en faciliter leur recherche et leur analyse ultérieure.

Cette tâche est à la croisée des chemins entre le domaine de la recherche d'information et celui de l'apprentissage automatique. En effet, un processus général inductif construit automatiquement un classifieur, en apprenant depuis un ensemble de documents pré classés; il s'agit des caractéristiques de la catégorie [72]. L'élaboration de ce classifieur nécessite un processus en trois étapes, qui constitue chacune un sous-domaine de recherche à part entière [1]:

1. l'extraction des descripteurs pertinents,
2. la phase d'apprentissage à partir d'un corpus d'entraînement,
3. l'évaluation du classifieur.

2.5.2.1 L'extraction des descripteurs pertinents

Dans un système de classification supervisée de document, chaque document est généralement représenté dans une approche communément appelé « sac de mots » (*Bag of Words, BoW*), où la plus petite unité pour représenter le document est le mot simple [1, 72] en utilisant le schéma de pondération *tf-idf* [70]. L'extraction des mots pertinents consiste à effectuer des opérations de filtrage comme par exemple l'élimination des mots vides, la racinisation, la lemmatisation, etc. Ce qui réduit considérablement la dimension de l'espace des termes et fait face au problème de la « malédiction de la dimension »[12]. Avec ces filtrages il est constaté que la matrice *documents* × *mots* reste encore très éparse, une stratégie connue sous le nom de *sélection d'attributs* est utilisée pour réduire encore l'espace de représentation des documents. Cette stratégie se base sur l'utilisation de mesures du domaine de la théorie de l'information telles que le gain d'information, l'entropie, l'information mutuelle, etc. [1, 72].

2.5.2.2 Méthodes de classification supervisée

Plusieurs méthodes de catégorisation de texte existent, dont nous citons, celles à base d'instances comme la méthode du plus proche voisin ou *k-NN* (*k Nearest Neighbors*), les méthodes probabilistes comme le classifieur bayésien naïf, la méthode des SVM (*Support Vector Machine*), les arbres de décision et les réseaux de neurones, etc. [72, 1]. La plupart de ces algorithmes d'apprentissage supervisé tentent de trouver un modèle, une

[12] The curse of dimensionality

fonction mathématique, qui explique le lien entre des données d'entrée (les Documents) et les classes de sortie (leurs Catégories). Cette phase est appelée phase d'apprentissage. Grâce à ce modèle, on peut alors déduire les classes de nouveaux documents. On dit que le modèle est utilisé pour prédire. Le modèle est bon s'il permet de bien prédire. Nous présentons dans les sections suivantes quelques-uns de ces algorithme, ceux les plus connus.

k-plus proches voisins, k-ppv Connue en anglais sous le nom de *k-nearest neighbor* (*k-NN*), cette méthode diffère des traditionnelles méthodes d'apprentissage car aucun modèle n'est induit à partir des exemples. Les données restent simplement stockées en mémoire. Pour prédire la classe d'un nouveau document, l'algorithme cherche les *k* plus proches voisins de ce document. Après avoir déterminé quels étaient les *k* plus proches voisins, il faut définir une méthodologie afin d'attribuer une classe au nouveau document. Dans k-NN de base, la classe majoritairement représentée par les *k* plus proches voisins est choisie. Une autre solution est de pondérer la contribution de chacun des *k* plus proche voisin en fonction de sa distance avec le nouveau cas à classer. La méthode utilise donc deux paramètres : le nombre *k* et la fonction de similarité pour comparer le nouveau document à ceux déjà classés. Ces paramètres sont importants car des résultats très différents résultent de leurs choix. La valeur de *k*, par exemple, est déterminée empiriquement par plusieurs essais. En effet, dans l'exemple de la figure 2.1 ci-dessous, si k=1, alors x_q est classé « + ». Si k=5, alors x_q est classé « – ».

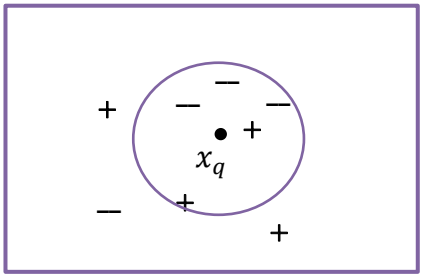


Figure 2.1 : illustration des k-plus proches voisins.

Pour la mesure de similarité, la plus couramment utilisée est le calcul du cosinus (voir formule 2.6, § 2.4.4) de l'angle formé par les deux vecteurs de documents.

Notons aussi que, si le temps d'apprentissage est inexistant, puisque les données sont stockées telles quelles, la classification d'un nouveau cas est par contre coûteuse puisqu'il faut comparer ce cas à tous les exemples déjà classés. Pour remédier à ce problème, plusieurs variantes de k-NN ont été proposées dans la littérature. Dans [88] par exemple, basé sur la stratégie du centroide, les auteurs proposent une nouvelle variante de k-NN,

centroid k nearest neighbors ou *CkNN*. Dans cette variante de l'algorithme k-NN, le nouveau cas est comparé aux centroides des *k* exemples, Ce qui permet d'améliorer les performances de l'algorithme de base.

Classifieurs Bayésiens naïfs Ces classifieurs se fondent sur le théorème de Bayes. Soit $\vec{d}_j = (w_{j1}, \dots, w_{jk}, \dots, w_{j|\Gamma|})$ un vecteur de termes représentant un document d_j ($|\Gamma|$ représente la taille du vocabulaire) et C un ensemble de classes. En s'appuyant sur le théorème de Bayes, la probabilité que le document d_j appartienne à la classe $c_i \in C$ est définie par :

$$P(c_i|\vec{d}_j) = \frac{P(c_i)P(\vec{d}_j|c_i)}{P(\vec{d}_j)}$$

La variable aléatoire w_{jk} du vecteur $\vec{d}_j$ représente l'occurrence du terme k retenue pour la classification dans le document d_j. Le classifieur naïf de Bayes, est fondé sur l'hypothèse de l'indépendance de ces occurrences d'où :

$$P(\vec{d}_j|c_i) = \prod_k P(w_{jk}|c_j)$$

La classe c_k d'appartenance de la représentation vectorielle $\vec{d}_j$ d'un document d_j est définie par :

$$c_k = arg\ max\ P(c_i|\vec{d}_j) = \arg max\, P(c_i) \prod_k P(w_{jk}|c_j)$$

Qui veut dire que le classifieur Bayésien naïf affecte au document d_j la classe ayant obtenue la probabilité d'appartenance la plus élevée. $P(c_i)$ est définie par :

$$P(c_i) = \frac{nombre\ de\ documents \in c_i}{nombre\ total\ de\ documents}$$

Arbres de décisions Un arbre de décision est un arbre tel qu'un nœud correspond à un attribut, les branches issues de ce nœud ont des valeurs possibles pour cet attribut. Les feuilles correspondent à une classe. Le chemin parcouru par un nouvel exemple de la racine de l'arbre jusqu'à la feuille détermine sa classe. La figure 2.2 donne un exemple d'un arbre de décision qui permet de prédire l'état d'un malade en fonction des symptômes relevés sur celui-ci.

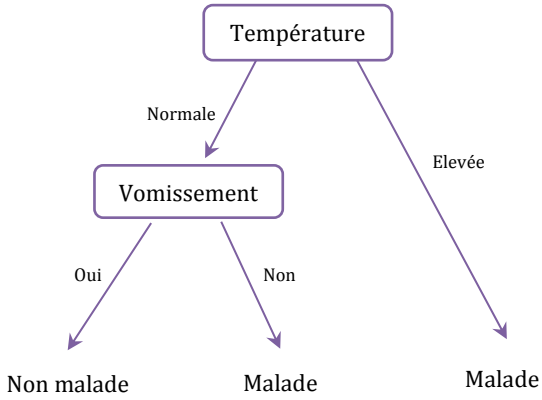


Figure 2.2 : Exemple d'un arbre de décision binaire.

Un classifieur de texte basé sur la méthode d'arbre de décision, est un arbre dont les nœuds internes sont marqués par les termes, les branches qui partent des nœuds sont étiquetées par des tests sur les poids des termes et les feuilles de l'arbre sont marquées par les catégories [72]. Dans ce classifieur, un nouveau document d_j est classé en testant recursivement les poids des nœuds internes du vecteur $\vec{d}_j$, jusqu'à ce qu'une feuille soit atteinte. L'étiquette de ce nœud est alors attribuée à d_j. CART (*Classification And Regression Tree*), ID3, C4.5 [72, 1] sont des algorithmes connus, permettant de construire les arbres de décisions. Ces algorithmes utilisent des règles de décisions sous forme de tests, « Si conditions alors Conclusions », pour construire l'arbre. Ces règles permettent à chaque branche de l'arbre de subdiviser l'ensemble des données en deux sous-ensembles plus homogènes. Cette homogénéité est une opération très importante qui est calculée en se basant sur des mesures telles que, par exemple, le gain d'information pour l'algorithme ID3 [60].

Machines à vecteurs supports (*Support Vector Machine SVM*) Le principe des machines à vecteurs supports (aussi appelés séparateur à vaste marge) suppose que l'on peut séparer linéairement les classes dans l'espace de représentation des éléments à classer [39]. Autrement dit, l'objectif est de trouver une surface linéaire de séparation (hyperplan) maximisant la marge entre les exemples positifs et négatifs d'un jeu d'apprentissage. L'idée est de garantir que la marge entre le plus proche des positifs et des négatifs soit maximale. Les points se situant précisément sur l'hyperplan sont appelés les *vecteurs supports* (voir figure 2.3).

Ce classifieur binaire est un des algorithmes les plus performants en classification textuelle. Un nouveau texte est classé dans une des deux catégories en fonction de sa position par rapport à l'hyperplan trouvé en phase d'apprentissage par la SVM.

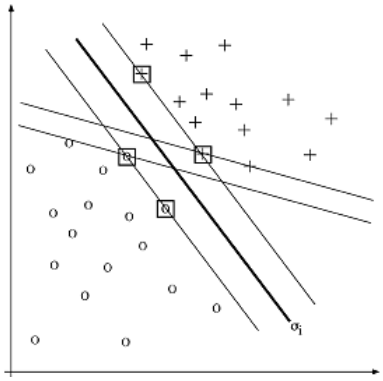


Figure 2.3 : Apprentissage par une SVM – les cercles et les croix représentent respectivement les exemples positifs et négatifs, les lignes représentent les surfaces de décision. La surface de décision σ_i montre le meilleur cas [72].

Nous avons présenté dans cette section les méthodes les plus connues en catégorisation de texte. Notre but n'est pas d'être exhaustif mais de présenter une vue d'ensemble de ces approches. Cependant, d'autres méthodes et algorithmes permettent également d'effectuer une classification avec un apprentissage supervisé. Nous pouvons citer les réseaux de neurones, les algorithmes génétiques, les modèles de markov cachés, etc. Un état de l'art complet de ces méthodes peut être trouvé dans [72, 1].

2.5.2.3 Évaluation d'un classifieur

L'évaluation d'un classifieur est menée de façon expérimentale [72]. Cette évaluation mesure l'efficacité du classifieur et qui est définie par sa capacité à prendre les bonnes décisions de classification [72]. Les mesures que nous allons présenter dans les sections suivantes, se basent sur les tables de contingence (aussi appelées tables de confusion) des tableaux 2.1 et 2.2. Les notations vp, vn, fp et fn désignent, pour une classe c_i donnée, respectivement :

- « vrais positifs », le nombre de documents correctement classés dans la classe c_i.
- « vrai négatifs », le nombre de documents correctement identifiés comme n'appartenant pas à la classe c_i.
- « faux positifs », le nombre de documents négatif classés dans la classe c_i.
- « faux négatifs », le nombre de documents positif classés comme étant négatif par le système.

Appartenance à la classe c_i		Jugement expert	
		Oui	Non
Jugement classifieur	Oui	vp_i	fp_i
	Non	fn_i	vn_i

Tableau 2.1: table de contingence pour la classe c_i

Appartenance aux classes $C = \{c_i\}_{i=1..\|C\|}$		Jugement expert	
		Oui	Non
Jugement classifieur	Oui	$VP = \sum_{i=1}^{\|C\|} vp_i$	$FP = \sum_{i=1}^{\|C\|} fp_i$
	Non	$FN = \sum_{i=1}^{\|C\|} fn_i$	$VN = \sum_{i=1}^{\|C\|} vn_i$

Tableau 2.2 : table de contingence globale

Précision et rappel. L'efficacité d'un classifieur est souvent mesurée en termes de *précision* et *rappel* de la RI adaptés à la tâche de catégorisation. Pour une classe c_i donnée, la précision P_i mesure la portion de documents correctement classés parmi ceux classés dans c_i, tandis que le rappel R_i mesure la portion de documents classés dans c_i, parmi ceux appartenant à la classe c_i. La précision et le rappel sont définis par la formule 2.11.

$$P_i = \frac{vp_i}{vp_i + fp_i} \; ; \; R_i = \frac{vp_i}{vp_i + fn_i} \tag{2.11}$$

La précision et le rappel peuvent ensuite être évalués sur l'ensemble des classes $C = \{c_i\}_{i=1..|C|}$ par l'une des méthodes suivantes :

- Macro-précision/rappel :

$$P^M = \frac{\sum_{i=1}^{|C|} P_i}{|C|} \; ; \; R^M = \frac{\sum_{i=1}^{|C|} R_i}{|C|} \tag{2.11'}$$

- Micro-précision/rappel :

$$P^\mu = \frac{VP}{VP + FP} \; ; \; R^\mu = \frac{VP}{VP + FN} \tag{2.11''}$$

F-mesure. Comme déjà cité (en § 2.4.5) la F-mesure représente la moyenne harmonique de la précision et du rappel. En catégorisation, on distingue la macro moyenne (*macro average*) et la micro moyenne (*micro average*) selon que la macro précision/rappel ou la micro précision/rappel, sont utilisées [72]. Ces deux mesures sont définies par la formule 2.12.

$$F_1{}^\mu - mesure = \frac{2 * P^\mu * R^\mu}{P^\mu + R^\mu} \; ; \; F_1{}^M - mesure = \frac{2 * P^M * R^M}{P^M + R^M} \tag{2.12}$$

D'autres mesures. D'autres ratios peuvent être calculés à partir de la table de contingence du tableau 2.2. Nous citons principalement :

- Taux de réussite (*accuracy*) :

$$TS = \frac{VP + VN}{VP + FP + VN + FN} \tag{2.13}$$

- Taux d'erreur :

$$TE = \frac{FP + FN}{VP + FP + VN + FN} \tag{2.14}$$

2.5.3 Classification non supervisée de documents textuels

La classification non supervisée ou partitionnement de documents (*clustering* en anglais), est la tâche dont l'objectif est de trouver des groupes au sein d'un ensemble de documents (ou des segments de documents). Les groupes recherchés, communément appelés clusters, forment des ensembles homogènes de documents du jeu de données partageant des caractéristiques communes. Un cluster est donc un ensemble de documents qui sont supposés être similaires entre eux (homogénéité dans les clusters) et être différents des documents appartenant aux autres clusters (hétérogénéité entre clusters).

Contrairement à la classification supervisée, dans la classification non supervisée le nombre et la définition des clusters n'étant pas donné a priori. De plus, aucune donnée étiquetée n'est disponible. Ainsi, le *clustering* vise à regrouper des documents en ne considérant que leur contenu. Il s'agit d'une opération exploratoire et descriptive [24]. Dans la suite de ce document nous utiliserons le terme de *clustering* pour qualifier un processus de classification non supervisée. De même, le résultat d'un tel processus sera appelé *schéma de clustering*. Aussi, les notations suivantes seront adoptées :

- Un objet désigne la représentation vectorielle du document à classer ;
- Soit $X = \{x_1, \dots, x_n\}$ l'ensemble des n objets à classer ;
- Soit $SC = \{C_1, \dots, C_K\}$ un schéma de clustering comportant K clusters ;
- Soit $|C_i|$ le nombre d'objet contenus dans le cluster C_i ;
- Soit $d(x_1, x_2)$ une distance entre les objets $x_1$et x_2 ;
- Soit $D(C_1, C_2)$ une distance entre les clusters C_1 et C_2.

2.5.3.1 Mesure de similarité

Pour réaliser l'opération de *clustering*, on fait fréquemment appel à la notion de similarité entre les objets. Il s'agit d'évaluer à quel point deux objets ou éléments sont similaires (ou dissimilaires) pour les regrouper ou les séparer. Le choix de la mesure de simila-

rité permettant de comparer les objets entre eux va induire la façon de les regrouper. En utilisant deux mesures de similarité différentes, les objets ne seront pas comparés, et de ce fait regroupés ou non, de la même façon. Les mesures les plus utilisées, notamment pour les données textuelles, sont la distance de Minkowski et la mesure du cosinus, définies par :

- Distance de Minkowski :

$$d(x_i, x_j) = \left(\sum_{k=1}^{p} \left| w_{ik} - w_{jk} \right|^{l} \right)^{1/l} \tag{2.15}$$

où w_{ik} (w_{jk}) correspond au poids du terme k dans le vecteur x_i (x_j). Selon les valeurs du paramètre l, on parle de distance Euclidienne ($l = 2$), ou de Manhattan ($l = 1$).

- Mesure du cosinus : déjà définie en section 2.4.4 (formule 2.6). Elle correspond au cosinus de l'angle formé par les deux vecteurs x_i et x_j.

$$d(x_i, x_j) = \cos(x_i, x_j) = \frac{x_i . x_j}{\|x_i\| . \|x_j\|} = \frac{\sum_{k=1}^{p} w_{ik} * w_{jk}}{\sqrt{\sum_{k=1}^{p} {w_{ik}}^2} * \sqrt{\sum_{k=1}^{p} {w_{jk}}^2}} \tag{2.16}$$

2.5.3.2 Structure des résultats de *clustering*

Différentes approches de *clustering* existent. Suivant ces approches, le type de résultat obtenu peut être différent. Ainsi, les clusters obtenus peuvent être des ensembles *durs* (*hard*) ou *flous* (*soft*). De plus, le résultat n'est pas forcément plat, et peut se présenter sous la forme d'une hiérarchie. Ces différents types de résultats sont présentés dans les sections ci-dessous.

Clustering dur vs. clustering flou les méthodes de *clustering dur* permettent de positionner un document dans un seul groupe. Par contre, les techniques de *clustering flou* permettent de situer le document dans plus d'un ensemble en précisant, pour le document, son degré d'appartenance à chacun de ces ensembles [24]. Le tableau 2.3 donne un exemple de *clustering* dur à gauche et un exemple de *clustering* flou à droite.

	C_1	C_2	C_3
D_1	1	0	0
D_2	0	1	0
D_3	0	0	1
D_4	0	0	1

	C_1	C_2	C_3
D_1	0,9	0,1	0
D_2	0	0,8	0,2
D_3	0	0,3	0,7
D_4	0	0	1,0

Tableau 2.3 : Exemple de résultats pour un *clustering* dur (à gauche) et un *clustering* flou (à droite).

Clustering plat vs. clustering hiérarchique De plus, pour les méthodes de *hard clustering*, certains effectuent des regroupements plats (*flat* en anglais), d'autres effectuent des partitionnements hiérarchiques. Les regroupements plats (figure 2.4) sont caractérisés par le fait qu'aucune relation précise n'est déterminée entre les différents clusters générés. Les

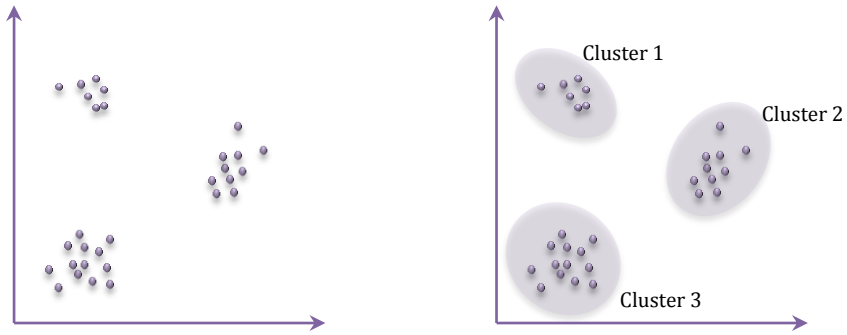


Figure 2.4: Illustration d'un clustering plat.

méthodes de cette catégorie sont très souvent de nature itérative. En effet, elles procèdent d'abord en déterminant un nombre fini de groupes, ensuite en raffinant chacun des regroupements effectués.

Pour certaines applications, il est cependant naturel de représenter le résultat sous la forme d'une hiérarchie de clusters. Les techniques effectuant le *clustering* hiérarchique, se caractérisent par la production de nœuds qui représentent chacun un sous cluster d'un nœud de niveau supérieur. Plus un cluster sera bas dans la hiérarchie plus il contiendra un faible nombre de documents mais qui seront plus similaires. L'ensemble des clusters étant représenté par un arbre (figure 2.5).

Deux grands types d'approches de *clustering* hiérarchique existent : les approches par agglomération (ou ascendantes) et les approches par division (ou descendantes). Les approches par agglomération, partent des documents à classer et ceux-ci sont ensuite regroupés jusqu'à obtenir un cluster unique contenant tous les documents. Les approches par division partent elles, de l'ensemble des documents, et les divisent en clusters qui sont ensuite divisés à leur tour de manière récursive.

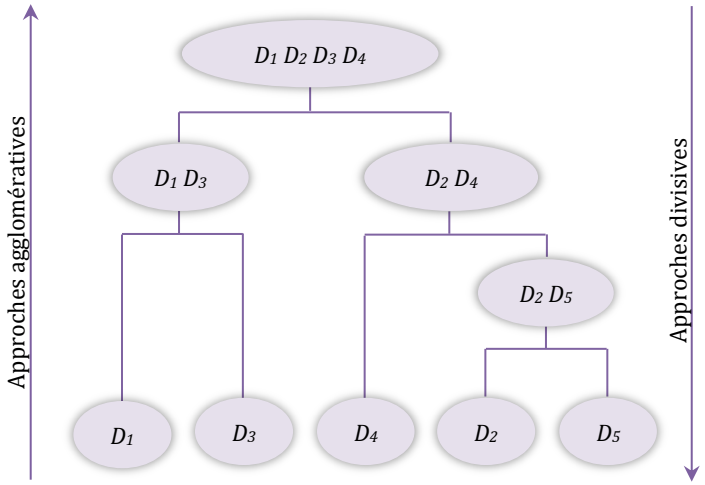


Figure 2.5 : Illustration d'un *clustering* hiérarchique.

2.5.3.3 Méthodes de *clustering*

Dans la section précédente nous avons présenté les principales structures de résultats obtenues en *clustering*. Sans vouloir être exhaustif[13], nous présentons dans cette section quelques méthodes de partitionnement. Une méthode de *clustering* est une stratégie générale employée pour résoudre un problème de *clustering*. Un algorithme de *clustering* est une instance d'une ces méthodes [38].

Les méthodes basées sur une distance ces méthodes sont parmi les premières méthodes de clustering à avoir été proposées et sont toujours très populaires aujourd'hui [38]. Ces méthodes se basent sur la notion de distance entre objets du jeu de données, en supposant que si deux objets sont proches suivant cette distance, ils doivent être regroupés ensemble dans un même cluster. Les principales méthodes utilisant une distance sont les méthodes à base de prototypes, les méthodes neuronales et les méthodes à base de densité.

Les méthodes basées sur les prototypes comme mentionné précédement, ces méthodes appartiennet à la classe des méthodes basées sur une distance. Ils définissent pour chaque cluster un objet représentant de ce cluster. Si ce représentant est calculé comme la moyenne des éléments du cluster, il est alors appelé un *centroide*. S'il s'agit d'un objet particulier du cluster alors celui-ci est appelé un *médoide*. Les algorithmes K-means et Fuzzy-C-Means sont les algorithmes les plus connus de cette famille de méthodes.

L'algorithme K-means L'algorithme des K-means (ou k-moyennes) [46] est sans aucun doute la méthode de partitionnement la plus connue et la plus utilisée dans divers domaines

[13] Le lecteur pourra se référer à un état de l'art détaillé, disponible dans [37].

d'application. Ce succès est dû à sa simplicité et son coût algorithmique relativement faible. L'algorithme K-means cherche un partitionnement des données en se basant sur les centroides des clusters, tout en optimisant une fonction objective. Cette dernière vise à évaluer la qualité du résultat pour faire converger l'algorithme vers la solution optimale. L'algorithme est donc itératif et consiste en la répétition de deux étapes :

1. Redéfinition des centres des clusters : le centre μ_i est défini comme le centroide des objets du *i*-ième cluster ;
2. Affectation des objets aux clusters : un objet x est affecté au cluster dont le centre μ est le plus proche selon une distance donnée.

À l'issue de ces deux étapes la fonction objective est localement optimisée. La fonction utilisée par K-means est la somme de l'erreur au carré (*SSE*) définie par :

$$SSE(SC) = \sum_{i=1}^{K} \sum_{x \epsilon C_i} d(x - \mu_i)^2 \qquad (2.17)$$

Avec μ_i le centroide du cluster C_i et d une mesure de distance entre les objets. Plus la valeur de *SSE* est petite plus les clusters sont compacts. L'algorithme K-means cherche donc à minimiser cette valeur. Les centres sont initialisés aléatoirement en prenant des objets aléatoirement dans X.

La version floue de K-means est appelée Fuzzy-C-Means [38]. Elle consiste à considérer une appartenance floue à chaque cluster plutôt qu'une appartenance dure comme dans K-means. Ainsi, un degré d'appartenance de chaque objet à chaque cluster est calculé en fonction de la distance de cet objet aux prototypes des différents clusters. Le degré d'appartenance μ_i d'un objet x au cluster C_i est calculé comme suit :

$$\mu_i\,(x) = \frac{1}{\sum_{j=1}^{K} \left(\frac{d(x,\, \mu_i)}{d(x,\, \mu_j)} \right)^{\frac{2}{f-1}}} \qquad (2.18)$$

où f est un paramètre qui définit le degré de flou et d une distance entre objets.

Les méthodes probalistiques ces méthodes supposent que les données suivent une certaine loi de probabilité. L'objectif est d'estimer les paramètres de cette loi de probabilité et de définir un mélange de lois (généralement guaussiennes) pour représenter les différents clusters. Ces méthodes font l'hypothèse qu'à chaque cluster C_i est associé une loi de probabilté $P(x, \theta_i)$ de paramètres θ_i qui permet de déterminer la probabilité

d'appartenance de x à C_i. Cette probabilité d'appartenance au cluster peut être interprétée comme un degré d'appartenance k à ce cluster. Un des algorithmes probabilistes les plus répandu est l'Espérance-Maximisation (en anglais Expectation-Maximization, EM) [20] dont le principe est d'alterner itérativement les phases dites d'espérance et de maximisation. L'espérance consiste à calculer l'espérance de vraisemblance en tenant compte des variables observées. La maximisation estime le maximum de vraisemblance des paramètres en maximisant la vraisemblance trouvée à l'étape d'espérance. Les paramètres résultant de la maximisation sont alors utilisés afin de réitérer l'opération.

Les méthodes hiérarchiques ces méthodes construisent une hiérarchie de clusters, c'est-à-dire un arbre de clusters pouvant se présenter sous forme d'un dendrograme. Chaque nœud contient ses clusters enfants, et les nœuds frères partitionnent les objets contenus dans leurs parents. Ces hiérarchies permettent d'explorer les données à différents niveaux de granularité. On distingue deux approches pour parvenir à de tels arbres hierarchiques : Les algorithmes agglomératifs (ou ascendants) et les algorithmes divisifs (ou descendants). Un partitionnement agglomératif construit l'arbre en partant des feuilles et procède par fusions successives des plus proches clusters jusqu'à obtenir un unique cluster racine, contenant l'ensemble des objets. Les algorithmes divisives, par contre, considèrent d'abord la racine contenant tous les objets, puis procèdent par divisions successives de chaque nœud jusqu'à obtenir des singletons.

Il existe plusieurs stratégies de regroupement de clusters pour les algorithmes agglomératifs. Les plus connues étant : *single-link*, *complete-link* et *average-link*. La stratégie *single-link* compare les deux clusters en considérant la distance minimale entre les objets des deux clusters :

$$D_s(C_i, C_j) = \min_{x \in C_i,\ y \in C_j} d(x, y) \tag{2.19}$$

La stratégie *complete-link* considère la distance maximale entre les objets des deux clusters :

$$D_c(C_i, C_j) = \max_{x \in C_i,\ y \in C_j} d(x, y) \tag{2.20}$$

Enfin, la stratégie *average-link* considère la moyenne des distances des objets des deux clusters :

$$D_a(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y) \tag{2.21}$$

2.5.3.4 Évaluation des résultats du *clustering*

Nous avons présenté dans la section précédente quelques approches de *clustering* dont le but final est de fournir un bon schéma de partitionnement. Chaque algorithme, selon les paramètres d'entrée peut produire différents résultats. Ainsi, l'évaluation et la validation des résultats d'un *clustering* constituent une difficulté majeure. Trois stratégies sont envisageables pour comparer entre eux des résultats de *clustering* [29, 38] :

L'évaluation externe : aussi connue sous le nom de l'évaluation supervisée. Il s'agit de confronter un schéma de *clustering* obtenu avec un partitionnement prédéfini des données. L'évaluation porte donc sur le degré de correspondance entre le schéma obtenu et le schéma attendu (connaissances externes).

L'évaluation interne : aussi appelée, évaluation non supervisée. Aucune connaissance externe n'est utilisée dans ce genre d'évaluation. Les critères d'évaluation internes se basent sur des informations internes au *clustering* comme par exemple la distance entre les objets d'un cluster et le centroide de celui-ci. Ces mesures se basent sur deux principaux critères, celui de la cohésion intra-clusters ou compacité des clusters (à minimiser) et la distance inter-clusters ou séparabilité des clusters (à maximiser). Plusieurs indices de qualité permettent d'évaluer la compacité ainsi que la séparabilité des clusters. Les plus utilisés, notamment pour les données textuelles, sont [25] :

L'indice de Dunn : pour un schéma de *clustering* SC, l'indice de Dunn, est défini par :

$$Dunn(SC) = \min_{i=1,\dots,K} \left\{ \min_{j=1,\dots,K;\, i \neq j} \left\{ \frac{\left(D_s(C_i, C_j)\right)}{\max_{i=1,\dots,K}\left(D_c(C_i)\right)} \right\} \right\} \tag{3.22}$$

où :

- $D_s(C_i, C_j) = \min_{x \in C_i,\ y \in C_j} d(x, y)$ correspond à la distance entre les clusters C_i et C_j, définie comme la plus petite distance entre les objets des deux clusters.
- $D_c(C_i) = \max_{x,y \in C_i} d(x, y)$ et correspond à la distance maximale entre deux objets du même cluster.

Si le jeu de données contient des clusters compacts et bien séparés, les distances entre les clusters sont supposées être grandes et les diamètres des clusters sont supposés être petits. Une grande valeur de l'indice de Dunn indiquera donc la présence de clusters compacts et bien séparés.

L'indice de Davies–Bouldin : cet indice mesure la compacité et la séparation des clusters en se basant sur le calcul de la moyenne de la similarité entre les clusters. Il est défini par la formule (2.23).

$$DB(SC) = \frac{1}{K}\sum_{i=1}^{K} \max_{j=1,\dots,K;i\neq j}\left(\frac{S(C_i)+S(C_j)}{d(\mu_i,\mu_j)}\right) \tag{2.23}$$

où $d(\mu_i,\mu_j)$ représente la distance entre les centroides de C_i et C_j et $S(C_i)$ la distance moyenne entre chaque objet de C_i et son centroide μ_i :

$$S(C_i) = \frac{1}{|C_i|}\sum_{x\in C_i} d(x,\mu_i) \tag{2.24}$$

Plus les clusters sont compacts, plus la moyenne de la distance au centroide $S(C_i)$ est petite. De même, plus les clusters sont séparés, plus la distance entre les groupes $d(\mu_i,\mu_j)$ est grande. Une faible valeur de l'indice de Davies–Bouldin indique donc, un *clustering* de bonne qualité.

L'indice Silhouette : L'indice silhouette peut être calculé pour chaque objet, pour chaque cluster et pour la partition toute entière. L'indice objet permet de vérifier si celui-ci a été bien classé. Pour cela, et pour chaque objet x de la partition, la valeur suivante est calculée :

$$S(x) = \frac{b_x - a_x}{max(a_x, b_x)} \tag{2.25}$$

Avec a_x la distance moyenne entre l'objet x et tous les autres objets appartenant au même cluster que x, et b_x la distance moyenne entre x et tous les objets n'appartenant pas à ce même cluster. La valeur de $S(x)$ varie entre -1 et 1. Une valeur positive ($a_x < b_x$) signifie que les objets appartenant au même cluster que x sont plus proches que les objets des autres groupes.

Pour un cluster, l'indice silhouette est la moyenne des indices des objets appartenant à ce cluster :

$$S(C_i) = \frac{1}{|C_i|}\sum_{x\in C_i} S(x) \tag{2.26}$$

Pour un schéma de *clustering*, l'indice silhouette est égal à la moyenne des indices de ces clusters :

$$S(SC) = \frac{1}{K}\sum_{i=1}^{K} S(C_i) \tag{2.27}$$

L'indice pour le schéma de *clustering* varie aussi entre -1 et 1, une valeur positive indique que les clusters sont très compacts et bien séparés. Cependant, il est à noter que le calcul de cet indice est très coûteux en temps car de nombreux calculs de distances sont nécessaires à son évaluation.

L'évaluation relative : cette stratégie est souvent utilisée pour comparer plusieurs schémas de *clustering* obtenus par un même algorithme avec différents paramétrages. Les mesures relatives sont simplement l'utilisation de critères d'évaluation internes ou externes pour faire un choix parmi plusieurs résultats produit par un même algorithme.

2.6 Systèmes de résumé automatique de texte

Le résumé automatique de texte est un sous-domaine du domaine du traitement automatique de la langue naturelle (TALN). Les recherches dans ce domaine remontent à la fin des années cinquante [45] et une renaissance des approches après les années 1990 due aux développements technologiques et l'explosion de la masse d'informations au format électronique. La technologie du résumé automatique de texte est devenue un thème majeur ainsi qu'un domaine de recherche des plus actifs. En effet, les recherches dans ce domaine ont été intensifiées, aussi les budgets qui lui sont consacrés ont été largement augmentés [47]. Plutôt que de diffuser des documents entiers, n'est-il pas préférable de diffuser seulement des résumés, dans lesquels les informations pertinentes du contenu original sont préservées ? C'est là le but d'un système de résumé automatique. Le résumé de texte qui permet au lecteur de décider rapidement s'il est intéressant de lire le texte source semble une réponse naturelle. Ainsi, avec ces résumés, des décisions effectives peuvent être prises en moins de temps [28].

2.6.1 Définition et Types de résumés

Le résumé automatique de texte est défini [47] comme étant le processus qui consiste à extraire l'information la plus importante d'une (ou plusieurs) source(s) afin d'en produire une version abrégée pour un (ou plusieurs) utilisateur(s) particulier(s) et pour une (ou plusieurs) tâches (applications) particulière(s).

Les résumés diffèrent selon leurs fonctions et selon le lecteur cible ou l'application auxquels ils sont destinés. Dans [28], les auteurs distinguent principalement trois fonctions pour un résumé :

- Les *résumés indicatifs* : ils offrent suffisamment de contenu pour prévenir l'utilisateur des sources pertinentes, présentes dans la source, qu'il peut lire avec plus de détails. Ils peuvent s'apparenter à une table de matières mais ne peuvent en aucun cas se substituer à la source.
- Les *résumés informatifs* : se substituent à la source en assemblant les informations pertinentes (informations quantitatives ou qualitatives apportées par l'auteur) ou nouvelles, en une forme concise.
- Les *résumés évaluatifs* ou *critiques* (*reviews*) : en plus de comporter un fond d'information, les résumés critiques évaluent les sujets traités par la source, exprimant ainsi des points de vue sur la qualité des travaux de l'auteur.

Par ailleurs, un résumé peut être générique ou orienté utilisateur :

- les *résumés génériques* s'adressent à une vaste communauté. Comme l'accent n'est pas mis sur des besoins particuliers, aucun groupe particulier n'est visé. De plus ces résumés traitent tous les sujets d'un document avec le même poids.
- Les *résumés orientés utilisateur*, en revanche, sont adaptés aux besoins spécifiques d'un individu ou un groupe particulier. Ces besoins, généralement exprimés au moyen d'une requête, doivent permettre au système d'isoler les parties du document concernant une (ou plusieurs) thématique(s) précise(s) pour ensuite produire un résumé n'incluant que ces dernières. Ce type de résumés sont aussi appelés résumés orientés requête.

2.6.2 Méthodes de création de résumé automatique

Les méthodes de création de résumé automatique de texte sont classées selon deux approches : *extraction* et *abstraction* (ou *génération*). Le résumé produit par *abstraction* est le résultat d'un processus qui consiste à paraphraser (recomposer) en des termes plus généraux, le contenu de la source en se basant sur la compréhension de celui-ci. Tandis que l'*extraction* consiste en la sélection de portions de texte, les plus pertinentes, depuis le document source en les concaténant ensemble pour obtenir un texte plus court [28]. Un paramètre important pour le résumé automatique est le taux de compression (TdC) et qui correspond au rapport (ratio) entre le document original et sa version abrégée.

Chacune de ces approches constitue un domaine de recherche à part entière. En plus, les méthodes par abstraction, se basent sur la compréhension du texte source pour en construire une représentation sémantique qui va servir à la génération du résumé. Bien que cette approche donne des résumés plus cohérents, son inconvénient majeur est qu'elle nécessite des ressources lourdes pour la représentation et la génération de texte. De plus, cette approche est généralement dépendante des domaines abordés et il est difficile d'étendre les systèmes construits à d'autres domaines. De nos jours, le but est de produire des résumés suffisamment utiles pour les utilisateurs dans leurs prises de décisions avec l'utilisation de méthodes pratiques et efficaces, ce qui va en faveur des méthodes par extraction. Pour ces raisons nous nous abstenons de présenter les méthodes par abstraction et nous donnons dans la section qui suit, les principes fondamentaux des méthodes par extraction.

2.6.3 Approches par extraction

L'objectif de cette approche est de pouvoir fournir rapidement et sans analyse en profondeur du texte, un résumé à l'utilisateur. Pour créer un résumé par extraction, il faut d'abord repérer les unités textuelles (généralement les phrases) considérées comme saillantes, ensuite le système va sélectionner les unités textuelles qui portent les idées principales du contenu du document, en respectant le taux de compression désiré. Le travail principal se situe alors dans l'évaluation de la pertinence des segments textuels suivant un ou plusieurs critères. Les techniques numériques/statistiques (c'est-à-dire essentiellement quantitatives) sont le plus souvent utilisées. Les prochaines sections visent à décrire les principales méthodes de sélection de phrases pour la production de résumé par extraction.

2.6.3.1 Méthodes par extraction avec analyse en surface

Fréquence des mots Initialement introduit par Luhn [45], le critère de fréquence de mots suppose que les mots les plus fréquents dans le texte à résumer, sont importants/pertinents. Cependant, quelques mots se produisent très souvent, peu de mots se produisent quelques fois, et beaucoup de mots se produisent rarement. Dans [70], les auteurs ont résolu ce problème de la pertinence des mots en proposant la métrique *tf.idf* (voir section 2.4.3). Rappelons que le *tf.idf* d'un terme dans un document est la fréquence de ce terme dans le document considéré, multipliée par l'inverse de la fréquence des documents dans lesquels le terme apparaît.

Une fois les mots pondérés, un score est attribué à chaque unité textuelle (généralement la phrase) par simple addition des scores de chacun des mots contenus dans celle-ci.

Position des phrases À cause de la structure des textes, la position d'une phrase dans un document peut, dans certains cas, se révéler un bon indicateur de son importance. Edmundson [21] utilise ce critère sous l'hypothèse que les phrases dont la position est sous certaines rubriques sont positivement pertinentes. Dans les articles de presse par exemple, la première phrase sert de point d'appui à l'ensemble du document, et est considérée comme la plus importante. Dans un article scientifique, ce sont les phrases de la conclusion qui sont les plus représentatives du document [34].

Mots des titres Edmundson [21] utilise ce critère sous l'hypothèse que si les principaux thèmes sont véhiculés dans les titres, les phrases importantes sont donc celles qui contiennent les mots des titres. Les scores attribués aux phrases sont fonction de l'occurrence de mots dans les titres et de leur regroupement.

Expressions prototypiques (Cue Phrases) L'approche de Edmundson [21] utilise des *expressions courantes* ou *cue phrases* sur la base de l'hypothèse que la pertinence probable d'une phrase est affectée par la présence de mots-repères (ou *cue words*) (exemple : « Significant », « Greatest », « Impossible », « Hardely », « in this paper we show »). Edmundson distingue trois types de *cue words*, les mots bonus affectant positivement la pertinence d'une phrase (exemple : « Significant », « Greatest »), les mots malus affectant négativement la pertinence d'une phrase (exemple : « Impossible », « Hardely ») et les mots nuls considérés comme non pertinents. Dans [79], l'auteur rapporte un taux de 54% de rappel et précision avec l'utilisation d'une liste de 1423 *cue phrases* identifiés manuellement, dans le cas d'articles scientifiques.

Pour exprimer la valeur de pertinence globale d'une phrase (mais aussi d'un paragraphe, etc.), Il suffit seulement de combiner toutes les valeurs des critères vus précédemment, par une fonction linéaire employant des coefficients pour différencier l'importance de certains critères par rapport à d'autres. Pour un segment textuel *S* dans un texte *T*, la fonction de score de *S* peut être exprimée comme suit :

$$Score(S) = \alpha_1 . c_1(S) + \alpha_2 . c_2(S) + \cdots + \alpha_k . c_k(S) \qquad (2.28)$$

où :

- $c_1, c_2, \ldots, c_k$ sont des fonctions calculant la valeur numérique d'un critère appliqué à un segment textuel.

- k étant le nombre total de critères retenus pour calculer le score de pertinence des segments du texte.
- $\alpha_1,\ \alpha_2, \ldots \alpha_k$ sont des coefficients associés à chaque critère, suivant le cas où certains doivent peser plus dans le score que d'autres.

Ainsi, les segments qui ont les scores de pertinence les plus élevés sont ceux qui seront extraits les premiers pour construire le résumé.

2.6.4 Résumé automatique de multi-documents

Initialement, les recherches dans le domaine du résumé automatique se sont concentrées sur le résumé d'un seul document. Le but était d'extraire des unités dites saillantes d'un seul texte. Néanmoins, et suite à l'explosion documentaire et à l'abondance de collections de documents traitant des thèmes similaires, on a ressenti la nécessité de mettre au point de nouveaux outils automatiques permettant la réduction et le filtrage des informations utiles à partir de documents multiples : c'est ce qu'on appelle le résumé multi-documents. Le résumé automatique multi-documents a pour objectif de s'appliquer sur un ensemble de documents. Le résumé à construire doit pouvoir contenir les informations les plus pertinentes contenues parmi tous les textes. Les phrases extraites peuvent alors être complémentaires, redondantes ou contradictoires. Il faudra donc veiller à la cohésion des phrases mais surtout à la cohérence du résumé (absence de contradiction ou de redondance dans l'enchaînement des phrases).

Certains traitements dans la construction du résumé restent les mêmes que dans le cas du résumé automatique sur un seul document, toutefois de nouveaux traitements deviennent nécessaires en raison des nouvelles difficultés. Trois problèmes majeurs ont été décrits, par Radev et al. pour le résumé multi-documents [61] :

- la reconnaissance des unités saillantes redondantes;
- l'identification des différences entre les documents;
- la cohérence du résumé.

Ces problématiques ont été traitées de différentes manière à travers des systèmes de résumé automatique par extraction [62, 26, 14, 9] ou par génération [51, 5].

Pour identifier la redondance dans les documents texte, plusieurs mesures de similarité sont utilisées dont la plus connue est *MMR-MD* (*Maximale Marginale Relevance – Multi-document*) [14]. La valeur MMR d'une phrase dépend de sa similarité par rapport à une

requête de l'utilisateur et de sa dissimilitude avec les phrases qui ont été déjà sélectionnées dans le résumé. MMR a prouvé d'être une méthode efficace pour réduire la redondance tout en maximisant la diversité des passages sélectionnés. L'algorithme est itératif et sélectionne à chaque passe la phrase qui maximise la similarité à la requête tout en minimisant la similarité aux phrases déjà sélectionnées, selon la formule (2.29) :

$$MMR = argmax_{s \epsilon S}\left[\lambda . Sim_a(s, Q) - (1 - \lambda) . max_{s_j \in E} Sim_b(s, s_j)\right] \qquad (2.29)$$

où :

- *S* est l'ensemble des phrases à résumer ;
- *Q* la requête ;
- *E* l'ensemble des phrases déjà sélectionnées ;
- Sim_a la mesure de similarité entre une phrase et la requête ;
- Sim_b la mesure de similarité entre les phrases ;
- λ est un coefficient, dans l'intervalle [0,1], permettant de régler l'importance de la redondance vis-à-vis de la pertinence.

Pour Sim_a *et* Sim_b, la mesure standard du cosinus (voir formule 2.6) telle que utilisée dans le modèle vectoriel en RI, est adoptée.

L'une des méthodes de résumé automatique la plus couramment utilisée est le résumé par partitionnement (*clustering-summarization*). L'un des avantages de cette approche est qu'elle est indépendante du domaine d'application [9, 50, 62]. Dans une 1ère étape, l'ensemble des phrases des documents sont regroupées dans des clusters où chaque cluster représente un thème pour la collection de documents. Dans une 2ème étape, les phrases du même cluster sont ensuite ordonnées selon leur distance au centre du même groupe, représentant sa similarité au thème. De la même manière, les clusters sont ordonnés selon leur distance au centre de la collection, représentant l'importance du thème à inclure dans le résumé. La figure 2.6, illustre ce processus.

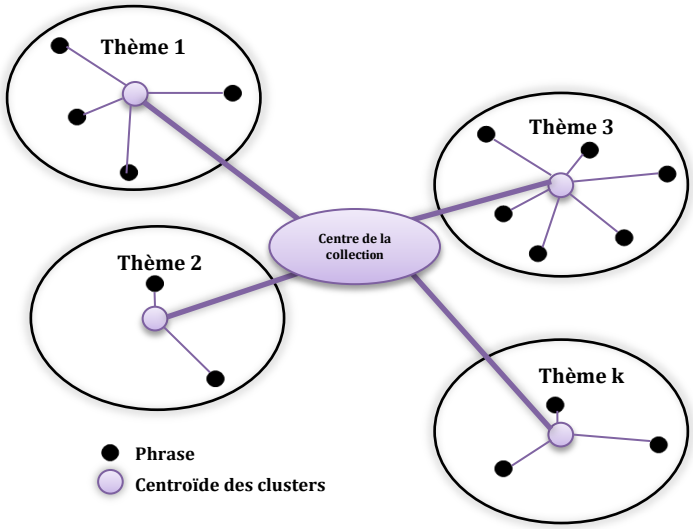


Figure 2.6 : *Clustering* pour la génération de résumé multi-documents

Le résumé est ensuite généré en prenant, de façon itérative, une phrase de chaque cluster jusqu'à ce que le taux de compression soit atteint.

2.6.5 Évaluation d'un système de résumé automatique

L'évaluation de la qualité d'un résumé a été prouvée d'être un problème difficile, principalement parce qu'il n'existe pas de résumé « idéal ». En effet, des résumés écrits par des personnes différentes, pour le même document, ne sont pas toujours convergents au niveau du contenu car peuvent être produits en utilisant un vocabulaire totalement différent. Spärk-Jones et al. [75] proposent de classer les méthodes d'évaluation en deux catégories :

- Les évaluations *intrinsèques*, où les résumés sont jugés directement en se basant sur leur analyse. Cette tâche peut être réalisée manuellement par des juges humains, évaluant la qualité d'un résumé comme la lisibilité, la cohérence etc., ou automatiquement en calculant des mesures de similarité entre le résumé candidat et un ou plusieurs résumés de référence.
- Dans les évaluations dites *extrinsèques*, les résumés sont évalués en se basant sur leurs impacts et fonctions sur des tâches annexes, comme par exemple l'utilisation des résumés à la place des documents sources, ou dans les systèmes question/réponse.

2.6.5.1 Évaluation intrinsèque

Dans le contexte du résumé, l'évaluation intrinsèque vérifie le système et la qualité du résumé lui-même. Il s'agit d'examiner la pertinence des phrases à l'intérieur même du résumé en regardant le résumé comme un objet. Pour ce genre de test, on peut comparer le contenu du résumé automatique avec un résumé modèle écrit par un humain. L'évaluation intrinsèque peut être mesurée avec les critères suivants :

- qualité de la langue,
- lisibilité,
- cohérence entre les segments du résumé.

Pour comparer un résumé machine avec un résumé modèle, on peut calculer la précision et le rappel. La précision mesure la proportion des unités pertinentes parmi toutes les unités produites par le système. Le rappel mesure la proportion des unités pertinentes parmi toutes les unités pertinentes. Ces mesures inspirées du domaine de la recherche d'information (voir § 2.4.5) peuvent être transposées dans le domaine du résumé de la façon suivante :

Soit,

- P, le nombre de phrases non pertinentes dans le résumé ;
- Q, le nombre de phrases pertinentes dans le résumé ;
- R, le nombre de phrases pertinentes présentes dans le résumé modèle et non trouvées dans le résumé automatique.

Le rappel et la précision sont alors calculés comme suit :

$$Rappel = \frac{Q}{(Q + R)} \tag{2.30}$$

$$Précision = \frac{Q}{(Q + P)} \tag{2.31}$$

Les valeurs de précision et rappel sont définies entre 0 pour le pire et 1 pour le meilleur des cas. Dans le contexte du résumé automatique, un système avec une précision plus élevée signifie que la plupart des phrases extraites dans le résumé produit sont pertinentes, alors qu'un rappel élevé indique l'identification d'un nombre plus élevé de phrases pertinentes dans le texte. Pour l'évaluation, il est nécessaire de considérer les deux mesures ensemble. La moyenne harmonique entre précision et rappel, définie par la F-Mesure (formule 2.10, § 2.4.5) est le plus souvent utilisée.

2.6.5.2 Évaluation extrinsèque

Les méthodes extrinsèques cherchent à évaluer un résumé en fonction de l'utilité et de la capacité qu'il apporte à effectuer certaines tâches. À la suite de la lecture d'un résumé par un sujet, on demande à ce dernier d'exécuter certaines tâches à partir de celui-ci dans l'objectif de voir comment seront affectées ces exécutions suivant la qualité du résumé. Par exemple, pour un document qui contient les réponses à des questions prédéfinies, on demande à des lecteurs de répondre aux questions en utilisant plutôt le résumé que le document source. Si les lecteurs répondent correctement aux questions demandées, le résumé est considéré comme bon car il couvre les informations essentielles du document.

2.7 Conclusion

Nous avons présenté dans ce chapitre la problématique de la gestion des grandes quantités de données textuelles de plus en plus disponibles, sur le web et dans les organisations. Nous avons présenté quelques approches de *summarization* de ces données contenues dans les documents textuels. Par *summarization* de documents textuels nous entendons un processus qui permet de résumer leur contenu les rendant ainsi analysables. Ceci permet de répondre au problème de surcharge informationnelle de type textuel, offrant à l'utilisateur des outils pour appréhender les éléments significatifs de ces documents.

Le résultat escompté est d'avoir, par exemple, les documents les plus pertinents par rapport à une requête exprimée par un utilisateur. C'est l'objectif de la recherche documentaire, présentée en section 2.4. Cependant, avec la croissance des documents textuels, le nombre de documents retournés par un SRI peut être grand où seule une petite partie de ces documents sera pertinente à l'utilisateur. Les SRI à venir se doivent de répondre à des besoins plus précis que ceux existants pour satisfaire au mieux les utilisateurs. Pour cela, d'autres techniques du domaine de la fouille de texte sont venues pallier aux inconvénients des SRI classiques. Nous avons vu que la catégorisation de texte, que nous avons abordée en section 2.5.2 est l'une de ces techniques. Avec la catégorisation de texte par exemple, il est possible de gérer automatiquement d'énormes quantités de données en les classant dans des catégories, comme dans le cas du filtrage du courrier électronique. De manière similaire, le regroupement de documents ou *clustering* (c.f section 2.5.3), permet de regrouper ensemble et de manière automatique des documents qui partagent les mêmes thématiques. L'intérêt d'une telle démarche est de résumer des volumes importants de documents textuels, de les organiser de façon à pouvoir effectuer, par la suite, une recherche ou une ex-

traction d'information efficace. Enfin, pour mieux analyser le contenu d'un ensemble de documents dans un cluster ou dans une classe, il est possible de résumer en quelques lignes les informations les plus pertinentes contenues dans les textes de celui-ci. C'est là, la tâche du résumé automatique que nous avons abordée en section 2.6. Dans toutes ces techniques, une étape primordiale consiste à représenter le contenu des textes par des descripteurs. Nous avons vu en section 2.4.2 que ces descripteurs sont considérés comme une représentation condensée du contenu des documents. Ils se présentent sous forme de mots simples (*keywords*) ou groupes de mots (*keyphrases*) et doivent être choisis de manière à perde le moins d'information sémantique de ces documents. Dans [52, 53, 43] les auteurs considèrent que les descripteurs représentent un résumé bref mais précis du contenu du document textuel.

Toutes ces techniques et approches sont le sujet du domaine de la fouille de texte qui fait partie d'un processus plus global, appelé « extraction des connaissances à partir de textes ». L'objectif de ce mémoire n'est pas de décrire ce dernier, mais dans le cadre d'un processus d'aide à la décision, notamment dans celui d'un environnement OLAP, nous pensons que fouille de texte et l'OLAP sont deux domaines qui se complètent et leur association serait une solution envisageable pour une analyse, sémantique riche, des données textuelles. Comment réaliser cette association, dépasse le cadre de ce mémoire. Cependant il y a lieu de citer que ces deux disciplines ont en commun de vouloir synthétiser de gros volumes d'informations pour en extraire celles les plus pertinentes, permettant ainsi à un utilisateur de prendre des décisions efficaces. C'est cet aspect de *summarization* des données, qui nous intéresse dans ce mémoire. En effet, nous nous sommes inspirés des techniques abordées dans ce chapitre pour proposer des fonctions d'agrégation pour les données textuelles. Avant de voir ces propositions nous terminons ce parcours bibliographique par présenter, dans le prochain chapitre un panorama de travaux ayant traité la problématique de l'analyse OLAP des données textuelles.

CHAPITRE 3
TEXT OLAP

3.1 Introduction

À la fin de ce parcours bibliographique, nous proposons de voir dans ce chapitre, un panorama de travaux qui ont proposé l'analyse des données textuelles dans un environnement OLAP. Nous présentons ces différents travaux dans la section 3.2. Une synthèse est faite dans la section 3.3, où nous catégorisons ces travaux. Nous terminons ce chapitre, ainsi que cette première partie de ce mémoire par une conclusion où nous positionnons nos contributions par rapport à l'existant.

3.2 Les travaux sur *Text OLAP*

L'analyse multidimensionnelle repose sur la capacité à résumer et à synthétiser des données très volumineuses. Bien que ces technologies ont montré leur utilité pour l'analyse et l'exploration des données structurées, elles doivent relever des challenges dans la prise en charge des données textuelles. Trois niveaux de difficulté sont liés à l'analyse multidimensionnelle de données textuelles :

1. Comment spécifier des *mesures* appropriées aux données textuelles ?
2. Comment définir des *hiérarchies de dimensions* spécifiques aux documents textuels pour permettre des analyses selon différents niveaux de granularités ?
3. Comment *agréger* les données textuelles ?

Parmi tous ces problèmes, l'agrégation des données textuelles est l'un des plus grand défis que doit relever les systèmes OLAP. En effet, avec les outils OLAP classiques, il est impossible d'agréger les données textuelles car les fonctions sont principalement arithmétiques. L'environnement OLAP pour les données textuelles, aussi appelé *Text OLAP* [55], a besoin d'outils appropriés et de nouvelles techniques d'agrégation pour ce type de données.

Des travaux récents se sont intéressés à intégrer les données textuelles dans les modèles multidimensionnels. Des dimensions spécifiques aux données textuelles, ainsi que

des mesures et méthodes d'agrégation adaptées à ces données sont proposées. Dans la suite de cette section nous présentons les principaux travaux ayant touché à ces aspects.

3.2.1 Le système BIKM (2002)

Pour Cody et al., deux technologies sont à l'origine de la croissance de la valeur quantitative et qualitative des connaissances, disponibles à la prise de décision dans une organisation. Le *business intelligence* (BI) qui représente la prise en charge des données structurées et le *knowledge managment* (KM) pour la prise en charge des données textes non structurées. Les auteurs soulignent l'importance de l'intégration à l'environnement OLAP, des outils d'analyse de texte en un système qu'ils appellent le système BIKM [17].

Pour cela les auteurs présentent un cadre d'analyse de texte (*framework*) qu'ils intègrent au data warehouse en introduisant un document warehouse (entrepôt de documents) et en mettant les deux en relation à travers des dimensions partagées.

Le modèle multidimensionnel de documents est donc un schéma en étoile, organisé autour d'une table de faits *documents* et où les dimensions sont les attributs des documents. Ce modèle est en relation avec un modèle multidimensionnel de données à travers des dimensions partagées, dans un schéma en constellation (voir figure 3.1 ci-dessus).

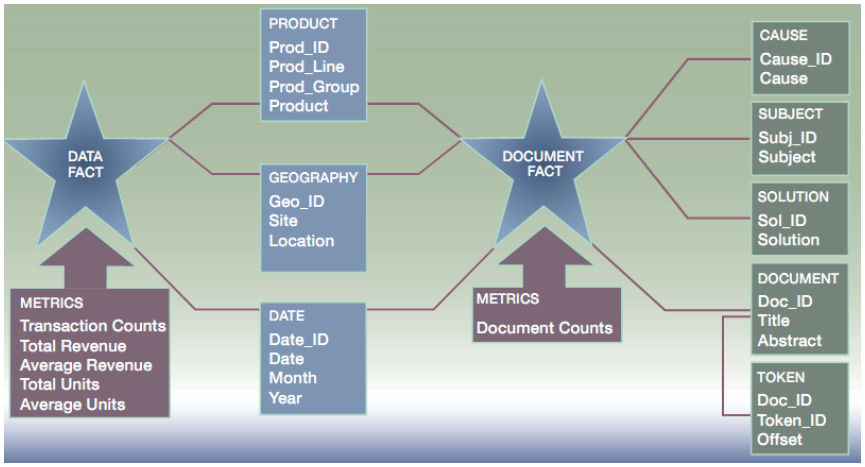


Figure 3.1 : Schéma en constellation du système BIKM [17]

La mesure utilisée pour le modèle de document, est une mesure numérique. Il s'agit du nombre de documents. Deux types de dimensions sont utilisés :

- Dimensions statiques :
  - Les dimensions qui sont partagées avec l'entrepôt de données (exemple : dimension *product)*.

- Les métadonnées des documents sont aussi utilisées comme dimensions.

– Dimensions dynamiques, elles sont appelées ainsi car elles sont déduites au moment de l'analyse, en invoquant un outil d'analyse de texte : *eClassifier*. Il s'agit principalement de la dimension catégorie issue d'un processus de classification ascendante hiérarchique, pour naviguer à travers les différentes catégories et approfondir ainsi la compréhension des données factuelles.

Toute analyse sur le cube de données peut être effectuée sur le cube de documents en utilisant les mêmes contraintes. Par exemple, si les données montrent une baisse dans le revenu dans certaines régions pendant une période de temps, les mêmes contraintes peuvent être utilisées sur le cube de documents pour identifier les documents qui peuvent mieux expliquer cette baisse. Ces documents sont analysés de manière plus approfondie en utilisant l'outil *eClassifier* qui permet de faire une classification ascendante hiérarchique de la collection de documents dans des catégories et permettre des *drill-down* et des *Roll-up* sur les différents niveaux des catégories.

3.2.2 XML-OLAP (2005)

Park et al., proposent une plateforme pour l'analyse en ligne de documents XML (orienté document) nommée XML-OLAP [54]. Dans cette plateforme les auteurs définissent des fonctions inspirées de la fouille de texte pour l'analyse du contenu des documents textuels : *SUMMARY*, permet la génération d'un résumé du texte à agréger ; *TOP-KEYWORDS*, sélectionne les *n* principaux mots-clés du texte à agréger ; *TOPIC*, extrait les thèmes d'un bloc de texte et *CLUSTER* partitionne le texte en fonction du contenu. Seulement, aucune définition ni détails d'implémentation ne sont donnés pour ces fonctions.

3.2.3 Document Cube (2006)

Tseng et Chou, définissent formellement les concepts relatifs à un entrepôt de document [82]. Ils proposent trois types de dimensions :

– Dimensions ordinaires : elles sont constituées à partir de données issues du contenu des documents analysés. Ces données sont extraites puis organisées de manière hiérarchique.

- Dimensions de métadonnées : elles représentent les informations relatives au contexte du document texte, telles que celles spécifiées dans les recommandations du Dublin Core[14] : les auteurs, l'éditeur, la date, le type, le format, etc.
- Dimensions catégories : les données de ces dimensions sont externes au document qui permettent sa catégorisation.

Un exemple de schéma multidimensionnel pour l'analyse des publications de journaux scientifiques, est donné dans la figure 3.2. L'analyse des documents s'appuie sur un schéma en étoile classique où seule la dimension catégorie est utilisée. La mesure utilisée est une mesure numérique, le nombre de documents.

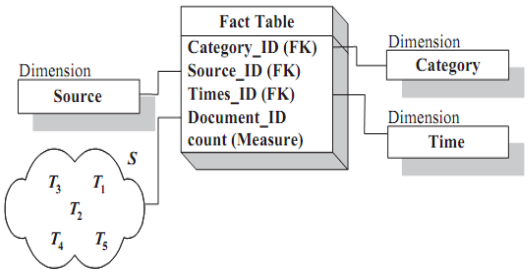


Figure 3.2 : Schéma en étoile pour l'analyse des publications [82].

3.2.4 R-cube (2007)

Pérez et al. [56, 57], proposent de coupler la modélisation multidimensionnelle et les techniques de recherche d'information pour fournir les documents pertinents relatifs à une analyse en cours. Pour cela, les auteurs proposent une architecture pour intégrer un entrepôt de données à un entrepôt de documents [56, 57]. Le résultat est un entrepôt contextualisé. L'approche de contextualisation consiste à considérer les documents comme une source complémentaire d'informations relative à un contexte d'analyse en cours afin d'expliquer les faits. Du moment que l'entrepôt de documents comprend des documents concernant différents thèmes, un SRI est utilisé pour sélectionner le contexte d'analyse depuis l'entrepôt de documents. Pour cela, dans un premier temps, l'utilisateur soumet une requête, sous forme de mots clés, suite à laquelle les documents pertinents vis-à-vis du contexte d'analyse sont restitués. Une fois le contexte défini, *R-cube* (*Relevance Cube*) est matérialisé. Chaque fait dans *R-cube* est relié à un ensemble de documents décrivant son

[14] http://dublincore.org/, Dublin Core Metadata Initiative.

contexte et lui sera assigné une valeur numérique représentant sa pertinence vis-à-vis de ce contexte. La figure 3.3 illustre l'architecture de l'entrepôt contextualisé.

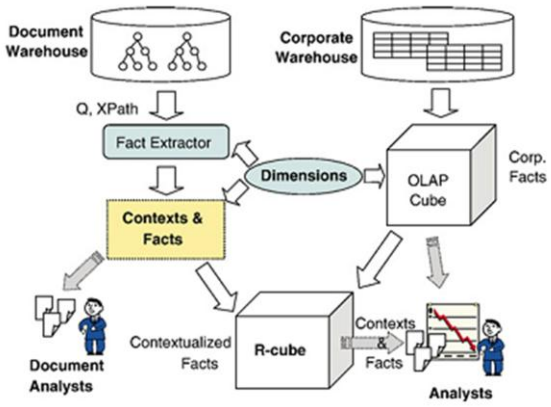


Figure 3.3 : Architecture de l'entrepôt contextualisé [57]

R-cube est donc un cube OLAP contextualisé par des documents texte et possédant deux dimensions : une dimension *contexte* et une dimension *pertinence*. Du moment que cette dernière dimension représente la pertinence du fait vis-à-vis du contexte, cette dimension peut servir à explorer la partie du cube la plus pertinente. Cependant, aucune fonctionnalité d'analyse de texte n'est offerte au décideur qui doit traiter lui-même (sélectionner et lire) les documents.

3.2.5 Travaux de Ravat et al., 2007, 2008

Pour les auteurs, la prise en compte de l'hétérogénéité des sources de données d'un système décisionnel impose à intégrer non seulement les données factuelles mais également les documents utiles pour un processus de prise de décision. Les contributions des auteurs portent sur les aspects modélisation et manipulation OLAP des documents [63, 64, 66].

Pour l'aspect modélisation, les auteurs ont proposé une extension du modèle en étoile pour l'analyse multidimensionnelle des données textuelles qu'ils appellent *constellation textuelle*. Pour cela, les auteurs proposent une dimension documentaire construite sur la base du contenu et de la structure logique des documents. La hiérarchisation de ce type de dimension représente la structure logique générique d'une collection de documents. Dans le cas où les documents ne partagent pas la même structure, plusieurs dimensions de structures peuvent être contenues dans une constellation où un fait ne peut être relié qu'à une unique dimension structure. Une extension du concept classique de mesure, est aussi pro-

posée pour les documents textuels, dans laquelle les auteurs distinguent la mesure *textuelle* composée de texte (un mot, un paragraphe, un document complet) et qui est non-additive. Les auteurs différencient entre deux types de mesures textuelles : la *mesure textuelle brute* qui est une mesure dont le contenu correspond au contenu complet d'un document ou d'un fragment de document. La mesure textuelle élaborée, est une mesure obtenue après application de traitements sur une mesure textuelle brute (exemple d'élimination des mots vides). La figure 3.4 illustre le modèle en étoile étendu pour l'analyse des données d'un laboratoire de recherche.

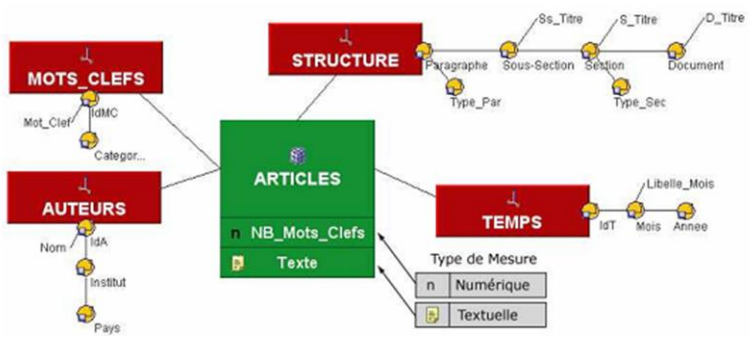


Figure 3.4 : Exemple de constellation textuelle pour l'analyse de documents.

Pour l'aspect manipulation OLAP, les auteurs proposent deux fonctions d'agrégation :

- La fonction *TOP_KW* [66] qui fournit les *n* principaux mots-clés, en utilisant la fonction de poids des termes *tf.idf* issue de la recherche d'information. Ce schéma de pondération a été adapté au contexte des systèmes décisionnels, où la représentativité d'un terme n'est plus recherchée vis-à-vis de la collection de documents mais vis-à-vis des fragments à agréger par la fonction.
- La fonction *AVG_KW* [64] qui combine plusieurs mots clés en un mot-clé plus général en utilisant une ontologie de domaine.

3.2.6 Multidimensional Content eXploration (2008)

Simitsis et al. [74], posent le problème de l'exploration et l'analyse de l'énorme quantité de données non structurées contenue dans les systèmes de gestion de contenu. Pour les auteurs, les techniques de la recherche d'information, jusque-là utilisées pour faire face à ce problème, s'avèrent insuffisantes. Ils pensent qu'une analyse avancée de l'information stockée semble être requise. Ils proposent donc de combiner OLAP et RI pour une exploration multidimensionnelle du contenu des documents.

Les auteurs proposent MCX (*Multidimensional Content eXploration*) [74], un *framework* dans lequel les concepts : document, contenu, métadonnées, liens entre documents, sont formellement mappés sur un schéma multidimensionnel : table de faits, dimensions statique ou dynamique, mesures telles que le montre la figure 3.5.

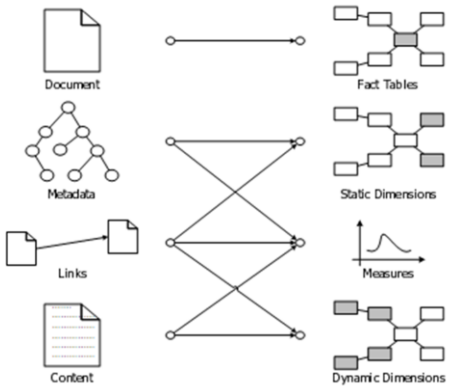


Figure 3.5 : Modélisation multidimensionnelle dans MCX [74].

Dans ce schéma, le document représente la table de faits, les métadonnées représentent des dimensions statiques, d'autres dimensions dites dynamiques sont déduites de l'analyse du contenu du document. La mesure utilisée dans MCX est la centralité du document, aussi appelée importance du document, sur la base de ses liens avec les autres documents. Les auteurs utilisent *PageRank* comme méthode de classement des documents. *PageRank* est une méthode utilisée, à l'origine sur le web. Elle permet de classer efficacement des documents ayant des liens entre eux.

L'idée de base dans MCX, est d'explorer les résultats de la recherche, suite à une requête utilisateur, en utilisant les techniques OLAP. Ceci ce fait en deux étapes :

- Dans un premier temps les documents les plus pertinents, par rapport à une requête utilisateur, sont extraits en utilisant un système classique de recherche d'information.
- Dans un deuxième temps, les métadonnées ainsi que les groupes de mots (*phrases*, suite de cinq mots au plus apparaissant dans au moins cinq documents), sont extraits de l'analyse du contenu des documents. Ces groupes de mots, qui représentent les thèmes les plus importants, sont utilisés comme dimensions pour construire le cube de texte.

Le framework MCX a été validé avec DBPubs [4] qui est un système pour analyser et explorer une base de données de publications. Notons que, l'idée des dimensions dynamique

est intéressante, vu qu'il est difficile de prévoir des dimensions à la conception sur le contenu textuel. Rappelons qu'elle a été aussi utilisée dans [17].

3.2.7 Topic Cube (2009)

Pour Zhang et al. [85], il est crucial de permettre à l'utilisateur, l'analyse des données textuelles, en utilisant la technologie OLAP avec l'analyse du contenu de texte de manière intégrée. Les auteurs préconisent donc de combiner OLAP et fouille de texte. Ils proposent un nouveau cube de données appelé : *Topic cube* [85] pour combiner OLAP avec un modèle probabiliste de thèmes et permettre l'analyse en ligne sur des dimensions de données textes. *Topic cube* est construit sur la base d'une base de données texte multidimensionnelle[15] et d'une arborescence hiérarchique de thèmes (*Topic*). Cette hiérarchie est définie au préalable par les experts du domaine. En plus des dimensions standards de la base de données, les documents textes sont mappés sémantiquement, sur la hiérarchie de thème (*Topic*) pour construire *Topic cube*. Ce qui permettra à l'analyste de procéder à des drill-down et des roll-up tout au long de la dimension thème et de visualiser les documents textuels selon différents niveaux de granularités. Le schéma multidimensionnel pour *Topic cube* est donné dans la figure 3.6.

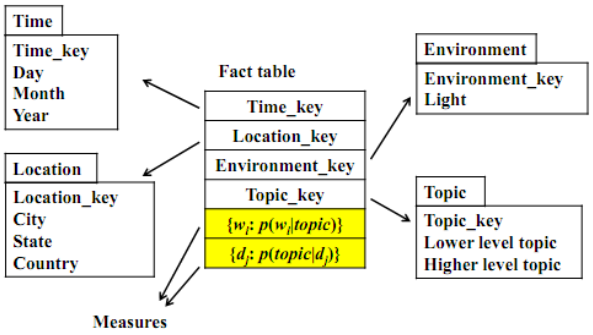


Figure 3.6 : Schéma en étoile de *Topic cube* [85]

Deux types de mesures sont stockées dans *Topic cube* :

- Une distribution de mots caractérisant le contenu textuel des documents, par rapport au thème : $p(w_i|topic)$;
- La couverture du thème par le document : $p(topic|d_j)$, ce qui permettra de prédire quel thème est dominant dans un ensemble de documents en agrégeant la mesure « couverture du thème » de tous ces documents.

[15] Une base de données texte multidimensionnelle, est définie [85, 86] comme une base de données ayant deux parties. Une partie constituée de champs standards (attributs structurés) et une partie constituée d'un champ texte. Les champs standards sont considérés comme le contexte du champ texte.

Ces probabilités sont calculées en estimant un modèle d'analyse sémantique latente probabiliste (PLSA) [33], basé sur le contenu textuel des cellules du cube. Les auteurs proposent aussi une matérialisation efficace (compromis entre stockage et temps de calcul) du cube en utilisant des heuristiques.

3.2.8 Micro-Text-Cluster Cube (2009)

Dans leurs travaux, Zhang et al., proposent un nouveau cube de texte appelé : *Micro-Text-Cluster Cube* [86]. Dans ce nouveau cube, les documents dans une cellule sont représentés de manière condensée en *k* partitions appelées *micro-clusters*. Ces micro-clusters sont pré-calculés, en appliquant la technique du *clustering* (l'algorithme *k-Means* est utilisé) et stockés dans le cube, dans une 1ère étape *offline*. Dans une 2ème étape *online*, c.-à-d. au moment de l'analyse, les auteurs proposent de faire la *summarization* du contenu des cellules en offrant à l'analyste deux types de résumés, un résumé *standard* ou *neutre* et un résumé *par thème* :

1. Dans le *résumé standard*, les micro-clusters dans une cellule sont regroupés en *P* partitions, appelées *macro-clusters*. Le résumé est constitué de *P* documents représentants les centroides de ces macro-clusters. *P* est un paramètre défini par l'utilisateur et représente le nombre de documents qui vont constituer le résumé de la cellule texte. Il s'agit des documents les plus représentatifs de cette cellule.
2. Dans le résumé par thème, les micro-clusters sont ordonnés selon la pertinence des documents dans le micro-cluster par rapport à une requête utilisateur (décrivant le thème). Le résumé est constitué des documents des micro-clusters les mieux ordonnés.

Dans *MiTexCluster cube*, chaque micro-cluster est représenté par un vecteur des poids des termes présents dans le micro-cluster selon la fonction de poids *tf.idf*, ainsi que le nombre de documents dans le micro-cluster. Cette représentation est utilisée comme mesure dans le cube. Le schéma en étoile de *MiTexCluster cube* est donné en figure 3.7.

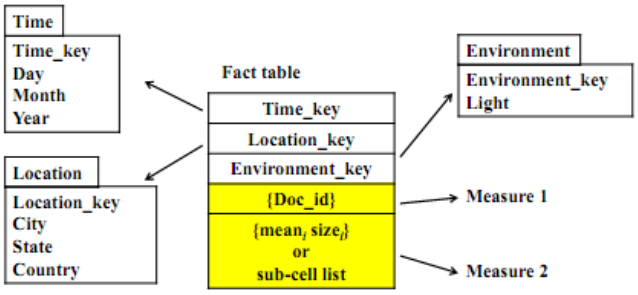


Figure 3.7 : Schéma en étoile de *MiTexCluster cube* [86]

Les opérations de forages ne sont pas définies. Les auteurs parlent de niveau de précision du résumé et qui dépend du nombre de micro-clusters, de macro-clusters et du nombre de documents fournis comme résumé. Dans [87], les auteurs reprennent les mêmes propositions mais en prenant en considération l'aspect qualitatif des résumés fournis.

3.3 Conclusion et positionnement

Dans ce chapitre nous avons tenté de donner une vision globale des solutions proposées pour la problématique de l'analyse des documents textuels dans un système OLAP. L'ensemble des travaux présentés dans la section précédente ne constitue pas une liste exhaustive, mais nous pensons qu'elle est représentative des contributions dans ce domaine. Nous avons vu que ces propositions vont du simple dénombrement d'instances de documents à des analyses plus approfondies sur le contenu de ces documents, en passant par une simple recherche documentaire. De plus nous pensons que ces travaux ont pris différentes tendances, et ont utilisé différentes technologies pour faire face au problème de l'analyse en ligne des données textuelles. Ils diffèrent principalement, selon que la mesure est numérique ou textuelle, que des opérateurs OLAP adaptés soient proposés, comment le document a été modélisé et est-ce que cette modélisation est classique ou adaptée. Le tableau 3.1, résume ces différents aspects.

	Type de Mesure				Dimension sur le contenu	Fonction d'agrégation adaptée
	Numérique	Numérique élaborée[16]	Textuelle brute	Textuelle élaborée		
Cody et al., 2002	✓				✓	
Park et al., 2005			✓			✓
Tseng et Chou, 2006	✓				✓	
Pérez et al., 2007	✓				✓	
Ravat et al., 2007	✓			✓	✓	✓
Ravat et al., 2008			✓		✓	✓
Simitsis et al., 2008		✓			✓	
Zhang et al., 2009		✓			✓	
Zhang et al., 2011		✓				✓
Nos propositions			✓		✓	✓

Tableau 3.1 : Comparaison des propositions pour l'analyse OLAP des documents textuels.

Par ailleurs, nous avons choisi de classer ces travaux en trois catégories, selon la technologie choisie pour analyser les données textuelles :

- Dans la première catégorie de travaux, un environnement OLAP classique est utilisé pour analyser les documents textuels. Les travaux les plus représentatifs sont ceux de [17, 82]. En effet, dans les (03) trois exemples présentés dans [82] un simple dénombrement des documents est réalisé.
- Dans la deuxième catégorie, un couplage entre la recherche d'information et la technologie OLAP est réalisé. Les travaux représentatifs de cette catégorie, sont ceux de [56, 57, 74, 4]. En effet, dans tous ces travaux, les auteurs préconisent de coupler RI et OLAP pour un environnement OLAP adapté aux données textuelles.
- Dans la troisième catégorie, l'utilisation des outils de la fouille de texte (*Text mining*) est prépondérante. Les travaux les plus représentatifs sont ceux de [17, 85, 86, 64, 65, 66, 65].

Dans certains travaux plus d'une technologie est utilisée, telle que dans [74]. Dans ce sens, nous pensons que RI et fouille de texte sont à coupler à l'OLAP dans un même environnement et de façon intégrée, afin de tirer profit de la richesse de la sémantique véhiculée par les données textuelles. Tel que nous l'avons vu dans le chapitre précédent OLAP et fouille de données sont deux domaines qui se complètent et leur association serait une solution envisageable pour une analyse des données textuelles plus élaborée dépassant la simple exploration des cubes de données. Nous pensons aussi que le manque de struc-

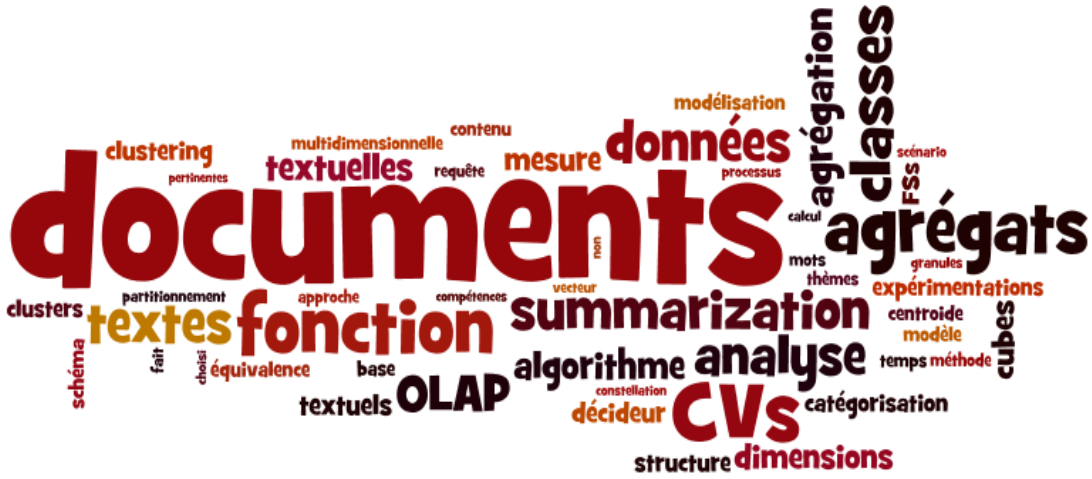
[16] Une mesure numérique élaborée est une mesure numérique calculée telle qu'elle est définie dans [81] comme les vecteurs des centroides des micro-clusters utilisés dans [85]. Ce type de mesure permet des analyses poussées mais nécessite des outils de visualisations conviviales car elles sont difficilement interprétables.

ture dans les documents textuels nous oblige à nous pencher sur les techniques d'analyse de contenu du texte issues du domaine de la fouille de texte.

C'est ainsi que nous avons basé nos propositions sur des outils de fouille de texte pour résoudre la problématique qui nous a été confiée, celle de l'agrégation des données textuelles dans un environnement OLAP. Ces propositions (voir tableau 3.1) se résument en (04) fonctions d'agrégation sur une mesure textuelle, il s'agit du contenu textuel du document ou des fragments de document. C'est ce que nous présentons dans la partie suivante de ce mémoire.

DEUXIÈME PARTIE : AGRÉGATION DES DOCUMENTS DANS LES CUBES DE TEXTES

Résumé. *Cette partie du mémoire, fait l'objet de nos contributions. Dans le chapitre 4, nous abordons le problème de la modélisation multidimensionnelle des documents textuels. Nous exposerons les deux approches existantes dans la littérature, ainsi que celle adoptée dans notre travail. Nous finirons ce chapitre en présentant une étude de cas pour la validation de nos propositions. Le chapitre 5, est consacré à la présentation de nos quatre fonctions d'agrégation. Une définition formelle pour chacune de ces fonctions est donnée, ainsi qu'un cadre théorique. Dans le chapitre 6, nous présentons les différentes expérimentations menées ainsi que les résultats obtenus.*



CHAPITRE 4
MODÉLISATION MULTIDIMENSIONNELLE DE DOCUMENTS TEXTUELS

4.1 Introduction

La modélisation conceptuelle multidimensionnelle a pour but de représenter les besoins utilisateurs en termes d'analyse. Cette modélisation vise à présenter les données sous une forme intuitive qui a pour objectif de se rapprocher de la manière dont les décideurs perçoivent les données d'analyse [16]. La modélisation multidimensionnelle permet d'effectuer des analyses OLAP.

Un modèle multidimensionnel est défini par un sujet d'analyse, le fait. Ce dernier est observé par un ou plusieurs indicateurs d'analyses, les mesures**,** selon plusieurs axes d'analyses, les dimensions. Ces dernières sont composées de paramètres**,** agencés de manière hiérarchique. Il est admis que la modélisation multidimensionnelle est bien maitrisée sur les données numériques. Cependant, des questions se posent quand il s'agit de prendre en charge les données textuelles par ces modèles. Est-il possible d'analyser des données textuelles représentées à l'aide d'un schéma multidimensionnel ? Les modèles de schémas en étoiles existants sont-ils adaptés à ce genre de données ? Comment doit-on les faire évoluer ?

Ce chapitre est consacré au problème de la modélisation multidimensionnelle des données textuelles. Nous présentons dans la section 4.2, les approches de modélisation pour ces dernières. Nous avons pu dégager deux approches dans la littérature, l'approche par extension de schémas que nous abordons en section 4.2.1 et la nouvelle modélisation en galaxie proposé par Ravat [63] que nous présentons en section 4.2.2. La section 4.3, est consacrée à l'approche adoptée. Comme la modélisation multidimensionnelle cible un problème d'analyse précis, nous présentons en section 4.3.1, un cas d'analyse, il s'agit de l'analyse des CVs. La section 4.3.2 donne quelques scénarii d'analyses des CVs. Le modèle proposé pour ce cas d'étude est abordé en section 4.3.3. En section 4.3.4, nous définissons le cube de textes pour l'analyse de CVs. Nous terminons ce chapitre par une conclusion, en section 4.4.

4.2 Approches de modélisation multidimensionnelle de documents

Les documents textuels présentent des caractéristiques qu'il sera intéressent de prendre en compte dans la modélisation multidimensionnelle. En effet, nous pouvons associer à un document textuel plusieurs vues [67] :

- Le contenu du document.
- La structure logique du document, qui permet d'identifier les granules d'information d'un document et de définir un découpage de l'information d'un point de vue hiérarchique. Une structure logique peut être décomposée en structure générique et spécifique. La structure générique exprime l'organisation générique commune à toute une classe de documents. La structure spécifique d'un document est une instance d'une structure générique.
- Les attributs externes ou identité du document, permettent de caractériser sans équivoque un document (date de création, format, n° version). Il s'agit des méta-données associées au document.

La prise en charge de toutes ces caractéristiques dans un modèle multidimensionnel constitue une problématique en soi. Les modèles existants sont-ils réellement adaptés pour la prise en compte des documents textuels ainsi organisés ? Nous avons pu recenser dans la littérature principalement deux approches pour la modélisation multidimensionnelle de documents textuels. Dans la première, la plupart des travaux proposent une modélisation multidimensionnelle par utilisation des schémas existants, à savoir le schéma en étoile ou en constellation. Dans la deuxième approche, Ravat et al. [63] proposent une nouvelle modélisation qu'ils appellent, *modèle en galaxie*. Les sections qui suivent sont dédiées à la présentation de ces deux approches.

4.2.1 Schéma en étoile pour la modélisation des documents textuels

Dans la plupart des propositions existantes dans la littérature (voir chapitre 3), un schéma en étoile ou en constellation classique [15] est utilisé. Cette solution se limite au dénombrement des documents où généralement la structure et le contenu de ces documents sont ignorés. Les analyses qui en résultent sont ainsi sémantiquement pauvres. Or, la spécificité des données textuelles fait que leur prise en charge dans un schéma en constellation doit prendre en compte les deux aspects suivants :

1. Spécifier des mesures appropriées aux données textuelles, afin de déterminer les fonctions d'agrégations possibles.

2. Définir des hiérarchies de dimensions spécifiques aux documents textuels pour permettre des analyses sur le contenu textuel, selon différents niveaux de granularités.

Le schéma en étoile ou en constellation a donc été adapté aux documents textuels dans certains travaux en prenant en compte l'un (ou les deux) de ces aspects. À notre connaissance, Ravat et al. sont les seuls auteurs qui aient défini formellement cette adaptation qu'ils ont appelé « constellation textuelle » [64]. Dans ce qui suit, nous définissons les concepts liés à une constellation textuelle.

4.2.1.1 Définition d'une constellation textuelle

Un schéma en constellation textuel est employé pour modéliser une analyse de contenus de documents où ce contenu est modélisé en tant que sujet d'analyse. Comme dans un schéma en constellation classique, un fait modélise un sujet d'analyse et une dimension modélise un axe d'analyse [64]. Cette définition décrit de manière classique la constellation au travers des concepts de fait, de dimension et de hiérarchie. Cependant, pour la prise en compte des spécificités des documents textuels, les auteurs proposent une extension du concept de mesure classique et de dimension.

4.2.1.2 Mesures pour une constellation textuelle

Les auteurs distinguent deux types de mesures : les *mesures numériques* et les *mesures textuelles*.

- Une mesure numérique est composée de données numériques. Elle est soit additive, soit semi-additive. Exemple : nombre de mots.
- Une mesure textuelle est une mesure composée de données textuelles qui sont à la fois non numériques et non additives. Le contenu d'une mesure textuelle peut représenter des termes (ou mots), un ensemble de mots, un paragraphe voire un document complet. Les auteurs distinguent deux types de mesures textuelles :
    - Une mesure textuelle brute : est une mesure dont le contenu correspond au contenu textuel d'un document ou d'un fragment de document.
    - Une mesure textuelle élaborée : est une mesure dont le contenu est issu d'une mesure textuelle brute et ayant subi un certain nombre de prétraitements. Exemple : les mots-clés obtenus, après application de traitements sur une mesure textuelle brute tel que le retrait des mots vides.

À cette typologie nous rajoutons un autre type de mesures qualifiées de mesures numériques élaborées [81]. La mesure numérique élaborée est une mesure numérique calculée telles que des vecteurs, des probabilités, etc. En effet, un document peut être représenté non pas par son contenu textuel brut, mais par son modèle de représentation vectorielle, par exemple[17].

4.2.1.3 Dimensions pour une constellation textuelle

Plusieurs dimensions pour la modélisation des documents textuels ont été proposées et qui sont résumées dans les définitions formelles de Ravat et al. [65] et Tseng et Chou [82]. Nous distinguons principalement quatre types de dimensions :

- Dimensions de métadonnées [82] : elles représentent les informations relatives au contexte du document texte, telles que celles spécifiées dans les recommandations du Dublin Core. Exemple : les auteurs, l'éditeur, la date, le type, le format, etc.
- Dimension de structure [65] : il s'agit d'une dimension qui modélise la structure commune des documents contenus dans une collection à analyser.
- Dimensions ordinaires [82] : ces dimensions sont constituées à partir de données extraites du contenu des documents analysés. Ces données sont extraites puis organisées de manière hiérarchique. Par exemple pour les auteurs, une dimension composée des principaux mots-clés extraits des documents est une dimension ordinaire.
- Dimensions catégories [82] : les données de ces dimensions sont des données externes au document qui permettent sa catégorisation. Par exemple, des catégories de documents d'un thésaurus tel que *Wordnet*.

4.2.2 Nouvelle modélisation : Galaxie

L'approche par extension de modèles existants a l'avantage de supporter directement les opérations classiques de manipulation OLAP, évitant ainsi de les redéfinir. Néanmoins, cette approche présente l'inconvénient de la dualité que jouent certaines données au sein des documents textuels. En effet, en fonction de l'analyse, la même donnée a été définie comme sujet dans certains travaux et axe d'analyse dans d'autres. C'est préci-

[17] Voir dans le chapitre 3, les travaux de Zhang [85, 86]

sément le cas des mots-clés qui peuvent jouer le rôle de paramètre dans une dimension ordinaire [82] ou mesure textuelle élaborée [65] selon le cas.

Pour pallier à ce problème, Ravat et al. [63] ainsi que Tournier [81] proposent un nouveau modèle appelé modèle en « Galaxie » dans lequel un concept unique, celui de dimension, est utilisé pour représenter les données qui peuvent être employées de manière symétrique en tant que sujet ou axe d'analyse. La notion de fait est ainsi supprimée.

4.2.2.1 Définition d'une Galaxie

Une galaxie est définie [63, 81] comme un regroupement de dimensions liées entre elles par un ou plusieurs nœuds centraux. Chaque nœud modélise les dimensions compatibles pour une même analyse. La figure 4.1 ci-dessous donne un exemple de galaxie pour l'analyse des publications scientifiques et des rapports de projets, d'instituts de recherches. La galaxie ci-dessous, décrit les *articles* publiés au sein d'une *conférence* à une *date* donnée et écrits par des *auteurs*. La même galaxie décrit aussi, les *rapports* de projets obtenus à une certaine *date*, pilotés par des *instituts* et employant des personnels scientifiques (les auteurs d'articles) [81].

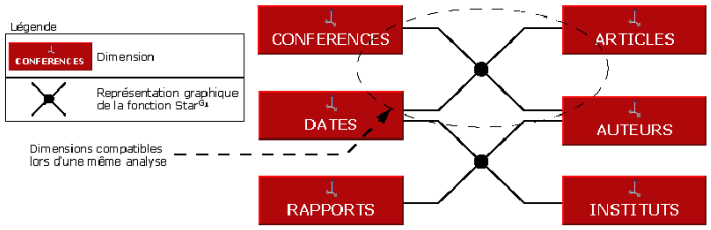


Figure 4.1 : Exemple de galaxie [81]

4.2.2.2 Autres concepts d'une Galaxie

Le concept de dimension dans une galaxie représente à la fois un axe d'analyse et un sujet d'analyse. Le concept de hiérarchie organise les attributs d'une dimension selon des niveaux de granularités, comme pour les approches classiques. Cependant, certains auteurs proposent un nouveau concept, celui de *liens de navigation* [63, 81]. Partant du fait que les documents comportent des liens hypertextes permettant de naviguer entre eux. Ces liens sont modélisés pour être exploités lors des analyses OLAP. Un lien de navigation est défini comme étant une liaison entre deux attributs par une relation « correspond à » entre les valeurs de ces deux attributs.

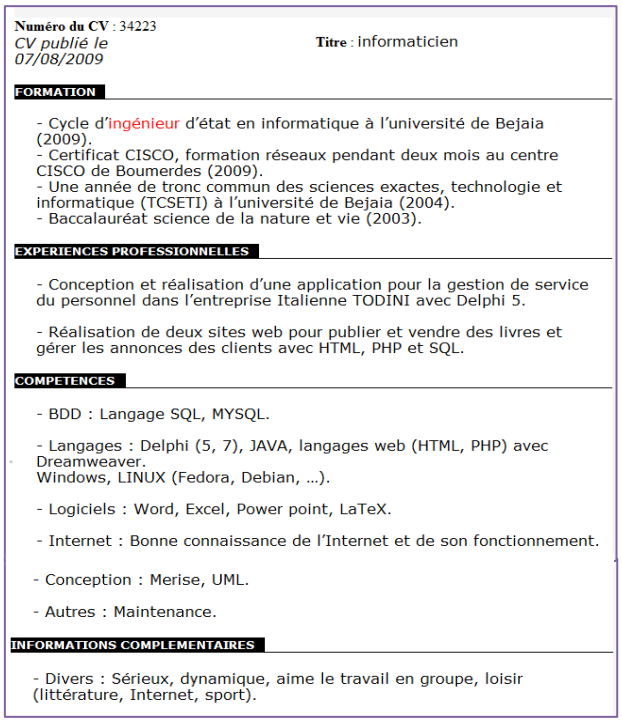
4.3 Approche adoptée

Bien que la galaxie semble pallier à certains problèmes des schémas classiques ou même de l'extension de ceux-ci, nous avons choisi d'adopter dans nos propositions l'approche par extension de schéma en étoile. La principale raison pour ce choix est que, ces derniers modèles sont bien maitrisés et les concepts qui leurs sont liés sont bien établis. D'autre part, la modélisation multidimensionnelle des documents textuels est une problématique à part entière et reste une question de recherche ouverte.

Pour illustrer nos propositions (dans ce chapitre et dans le suivant) nous avons ciblé un problème d'analyse précis : l'analyse des CVs (curriculum vitae) du domaine des ressources humaines. Cette partie sera dédiée tout d'abord à la présentation de l'étude de cas, section 4.3.1. Des scénarii d'analyse pour les CVs sont proposés dans la section 4.3.2. Le modèle multidimensionnel proposé pour ce cas d'analyse est présenté en section 4.3.3. La définition du cube de textes ainsi que les concepts qui lui sont liés, sont abordés en section 4.3.4.

4.3.1 Étude de cas : Analyse de CVs

L'analyse de CVs est le cas d'étude sur lequel s'est porté notre choix pour valider nos propositions dans ce mémoire. Le curriculum vitae ou CV, constitue la base du processus de recrutement. Il représente pour l'organisme d'embauche le document principal d'une candidature qui résume le parcours professionnel de la personne de façon concise. Comme le montre la figure 4.2, le CV est un document relativement court, syntaxiquement pauvre mais d'apparence structurée. Les informations qu'il comporte sont groupées de façon thématique au sein de sections. On retrouve généralement, les sections "Formation", "Expériences professionnelles", "Compétences", "Informations complémentaires" ou encore "Divers".



Numéro du CV : 34223
CV publié le 07/08/2009

Titre : informaticien

FORMATION

- Cycle d'ingénieur d'état en informatique à l'université de Bejaia (2009).
- Certificat CISCO, formation réseaux pendant deux mois au centre CISCO de Boumerdes (2009).
- Une année de tronc commun des sciences exactes, technologie et informatique (TCSETI) à l'université de Bejaia (2004).
- Baccalauréat science de la nature et vie (2003).

EXPERIENCES PROFESSIONNELLES

- Conception et réalisation d'une application pour la gestion de service du personnel dans l'entreprise Italienne TODINI avec Delphi 5.

- Réalisation de deux sites web pour publier et vendre des livres et gérer les annonces des clients avec HTML, PHP et SQL.

COMPETENCES

- BDD : Langage SQL, MYSQL.

- Langages : Delphi (5, 7), JAVA, langages web (HTML, PHP) avec Dreamweaver.
Windows, LINUX (Fedora, Debian, …).

- Logiciels : Word, Excel, Power point, LaTeX.

- Internet : Bonne connaissance de l'Internet et de son fonctionnement.

- Conception : Merise, UML.

- Autres : Maintenance.

INFORMATIONS COMPLEMENTAIRES

- Divers : Sérieux, dynamique, aime le travail en groupe, loisir (littérature, Internet, sport).

Figure 4.2 : Exemple d'un CV.

4.3.2 Quelques scénarii pour l'analyse de CVs

La croissance exponentielle d'Internet a permis le développement d'un grand nombre de sites d'emplois et d'un marché du recrutement en ligne en expansion[18]. Ceci implique des volumes d'informations (majoritairement sous la forme de texte libre) qu'il n'est plus possible de traiter manuellement. Une analyse assistée nous semble pertinente en réponse à cette problématique.

La sélection de candidatures sur la base du document CV constitue la tâche la plus courante effectuée par un recruteur. Cependant, d'autres analyses peuvent être faites sur ce document afin d'observer certaines tendances sur le marché du recrutement. Dans la section qui suit nous proposons quelques scénarii qui peuvent servir de base à une analyse approfondie pour un décideur.

Scénario 1, présélection des candidatures : comme cité plus haut, la sélection de candidatures est la tâche la plus importante pour un recruteur. Seulement, les réponses des candidats à une offre d'emploi représentent une grande quantité d'information difficile à gérer rapidement et efficacement par ces derniers. Il devient nécessaire de traiter cette masse d'information de façon automatique ou assistée pour une meilleure prise de décision. Par

[18] En 2011, près de 60% des offres d'emploi ont été publiées sur internet, ce qui fait du net le premier média de recrutement en Algérie, annonce le site de l'e-recrutement, Emploitic.com.

rapport à une offre d'emploi et sur la base d'un document CV du candidat, le recruteur voudrait faire une présélection des candidatures en les classant en trois catégories[19]. Les candidatures pertinentes (classe A), celles susceptibles d'être pertinentes (classe B) et enfin celles qui sont non pertinentes (classe C). Ce classement se fait sur la base d'informations telles que : la localité du candidat, son niveau d'étude, sa spécialité, son expérience professionnelle, ainsi que les compétences recherchées. Si le recruteur spécifie en plus la partie ou les parties (sections) du CV qui vont servir à constituer les classes, les résultats seront plus précis.

Scénario 2, regroupement des candidatures : dans ce deuxième scénario, le décideur voudrait découvrir des groupes homogènes dans un ensemble de CVs en procédant à un partitionnement de cet ensemble. Des candidats d'une même région et de même niveau d'étude, peuvent partager des expériences, des formations ou même des intérêts personnels. Le résultat de ce partitionnement sera d'autant plus intéressent que si l'utilisateur spécifie la section du CV selon laquelle portera ce partitionnement. Des groupes peuvent être constitués en utilisant, par exemple la section « formation » des documents CVs. Ces groupes reflèteront les tendances en matière de formation dans les universités algériennes, par exemple.

Scénario 3, analyse des compétences des candidats : dans un troisième scénario, le décideur voudrait par exemple connaître les compétences les plus courantes dans un secteur donné et une région donnée. Il est donc judicieux d'utiliser la section « compétences » des CVs pour extraire les groupes de mots les plus pertinents. Ces multi-mots clés, c.-à-d. les compétences présentes dans les CVs, sont-ils en rapport avec les compétences attendues ?

Scénario 4, analyse de l'expérience professionnelle des candidats : enfin prenant ce dernier scénario, où le décideur voudrait avoir un résumé des meilleures expériences des candidats dans une région donnée et un niveau d'étude précis. Les segments textuels les plus pertinents sont extraits des CVs en utilisant la partie « Expériences professionnelles ». Quels secteurs d'activités sont prédominants pour cette région ? s'agit-il de secteurs privés ou étatiques.

[19] Pour certains recruteurs, le nombre de classes peut dépasser les trois classes. Pour d'autres, les classes sont au nombre de deux : classe des candidatures pertinentes et la classe de candidatures non pertinentes.

4.3.2 Modèle proposé pour l'analyse de CVs

Les scénarios présentés dans la section précédente constituent des besoins d'analyse sur la base de laquelle nous avons construit notre modèle. Ce dernier doit prendre en considération les caractéristiques du document CV (contenu et structure) ainsi que les informations de contexte (localité, niveau d'étude, etc.), hiérarchisées ou non.

Ainsi, dans le modèle multidimensionnel que nous proposons, nous considérons le CVs comme fait que nous observons selon plusieurs dimensions. Nous retenons pour notre modèle deux types de dimensions, des dimensions de métadonnées et une dimension structure :

- Dimensions de métadonnées :
  - Secteur d'activité : il s'agit du secteur pour lequel le candidat postule sa demande d'emploi. Exemples : Informatique, Éducation, Commercial, Enseignement, etc.
  - Localité : cette dimension présente deux niveaux hiérarchiques, permettant de regrouper les villes en régions.
  - Niveau d'étude : qui permet de connaître à quelle qualification correspond la candidature. Exemples : Bac, Bac+2, Bac+5 et plus, etc.
- Dimension de structure : cette dimension modélise la structure logique ainsi que le contenu du CV, ce qui permet des analyses sémantiques sur les CVs.

La figure 4.3 illustre le modèle proposé.

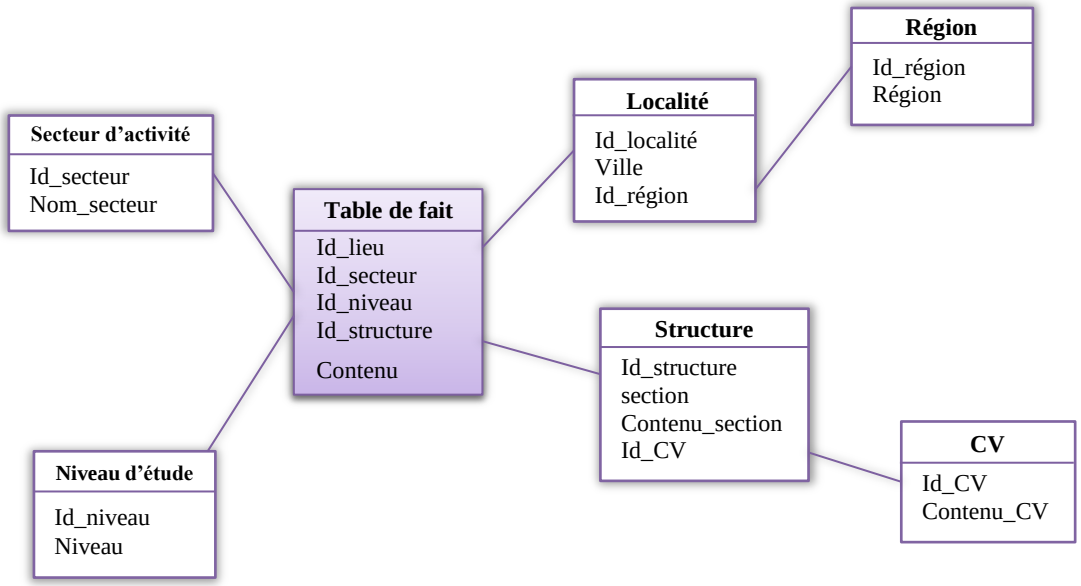


Figure 4.3 : Modèle multidimensionnel pour l'analyse de CVs.

4.3.3 Cube de textes pour l'analyse de CVs

L'instanciation du modèle présenté dans la section précédente, permet de construire le cube de textes pour l'analyse de CVs. Dans cette partie nous allons tout d'abord définir les concepts liés aux cubes de textes. Nous donnerons un exemple de ce type de cube pour l'analyse de CVs, qui va servir de base pour le prochain chapitre.

Définition 1 (Cube de textes). Un cube de textes est un cube de données utilisé pour la modélisation et l'analyse multidimensionnelle de documents textuels. La mesure utilisée dans un cube de texte est une mesure textuelle qui modélise le contenu (ou une partie du contenu) du document à analyser.
Nous définissons TC un cube de textes comme étant le couple suivant : $TC = \langle D, m \rangle$
Où $D = \{D_1, D_2, \dots, D_n\}$ est un ensemble non vide de n dimensions, m une mesure textuelle. Les dimensions dans un cube de textes peuvent être structurés ou textuelles, hiérarchisées ou non (voir typologie de dimensions, section 4.2.1.3).

Définition 2 (Cellule). Une cellule d'un cube de textes consiste en un ensemble de n documents (ou granules de documents), obtenus de l'agrégation sur un sous-ensemble non vide de dimensions structurées D' tel que $D' \subset D$.

Définition 3 (Requête). Nous utiliserons sur le cube de textes, une requête de la forme $(a_1, a_2, \dots, a_n, s)$. où a_i représente la valeur de la i-ième dimension, s la valeur de la dimension structure.

Exemple : le cube pour l'analyse des CVs peut être noté : $TC = \langle \{L, SA, NF\}, m \rangle$

$L =$ Dimension *Localité.*
$SA =$ Dimension *Secteur d'activité.*
$NF =$ Dimension *Niveau* de *formation.*
m, une mesure textuelle qui représente le contenu du CV.
La figure 4.4, ci-dessous illustre le cube de textes pour l'analyse de CVs.

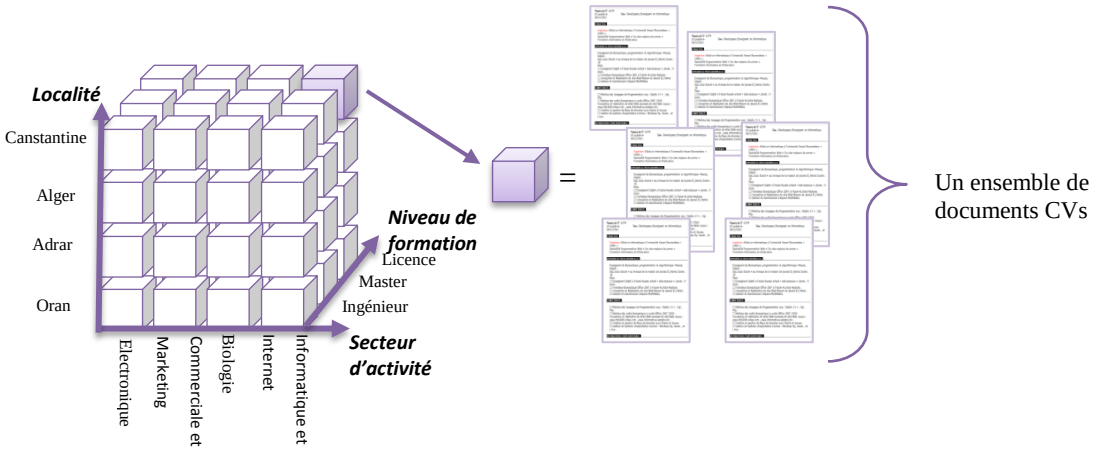


Figure 4.4 : cube de Textes pour l'analyse de CVs

Un exemple de requête sur le cube de la figure 4.4, (Lieu = "Alger", secteur d'activité = "Informatique", niveau d'étude = "Bac + 5 et plus", structure = "Formation").

4.4 Conclusion

Nous avons présenté dans ce chapitre la problématique de la modélisation multidimensionnelle des documents textuels. En effet, la prise en compte de ces derniers dans les modèles existants n'est pas une chose évidente. Nous avons vu que les documents textuels présentent des spécificités qu'il est souhaitable de les prendre en considération dans la modélisation. À cet effet, une extension des concepts de base des schémas en étoile a été proposée [64]. Malgré cela des problèmes subsistent, d'où la proposition d'une nouvelle modélisation, la galaxie [63, 81]. Cette dernière semble pallier aux problèmes de la constellation textuelle, néanmoins cette nouvelle modélisation doit être examinée de plus près pour déduire ces réels apports vis-à-vis des schémas en étoile ou en constellation, bien maitrisés. Ce qui sort du cadre de ce mémoire. Nous avons donc adopté une constellation textuelle pour le cas d'étude que nous avons choisi : l'analyse de CVs. Le schéma proposé prend en compte la structure visible ainsi que le contenu du document CV. La mesure que nous avons choisie, est une mesure textuelle, à laquelle des fonctions d'agrégation adaptées doivent être proposées. C'est l'objet du prochain chapitre où nous proposons quatre fonctions de *summarization* pour l'agrégation des documents, ce qui constitue le cœur de nos contributions dans ce mémoire.

CHAPITRE 5
AGRÉGATION DES DOCUMENTS DANS LES CUBES DE TEXTES

5.1 Introduction

L'agrégation des données est la pierre angulaire dans l'entreposage de données et l'analyse en ligne, OLAP. Contrairement aux données numériques, les données textuelles ne s'agrègent pas par des fonctions arithmétiques. Seules les fonctions LIST et COUNT s'appliquent à ces données. L'agrégation des données textuelles peut s'envisager à travers la *summarization* de texte. Rappelons que, par *summarization* de documents textuels nous entendons, un processus qui permet de résumer leur contenu sous une nouvelle forme permettant de les analyser afin d'en extraire de la connaissance. La *summarization* peut donc s'envisager en utilisant des techniques bien maitrisées des domaines manipulant les données textuelles et que nous avons présenté dans le chapitre 2. En s'inspirant de ces techniques, nous proposons quatre fonctions d'agrégation *Classes*, *Clusters*, *Top_Keyphrases* et *Summary*. Ces fonctions permettent de résumer les données textuelles en exploitant la mesure textuelle dans les cubes de textes. Les sections 5.2 à 5.5 sont consacrées à chaque fonction. Comme la construction d'un cube de données cible un problème d'analyse précis, nous présentons le principe de chaque fonction sur la base d'un scénario d'analyse parmi ceux que nous avons donné dans le chapitre 4 (*cf.* section 4.3.1). Pour chaque fonction aussi, nous définissons un cadre formel ainsi qu'un cadre théorique. Enfin nous terminons ce chapitre par une conclusion.

5.2 Fonction *Classes*

L'idée de base de la *summarization* par catégorisation est d'exploiter la mesure textuelle contenue dans le cube de textes pour présenter au décideur un résumé sous forme de *n* classes. Les applications sont nombreuses, nous présentons dans la section suivante un scénario de l'étude de cas présenté dans le chapitre 4.

5.2.1 Exemple

Notre exemple se base sur le scénario 1 présenté en section 4.3.2 du chapitre 4. Dans ce scénario, le recruteur voudrait faire une présélection d'un ensemble de CVs sur la base d'une offre d'emploi. Par exemple, la requête peut être (Localité = "Alger", secteur d'activité = "Informatique", niveau d'étude = "Bac + 5 et plus"). Nous proposons au recruteur un modèle multidimensionnel. Ainsi, l'indicateur à observer est une mesure textuelle, le CV. Trois axes d'analyses sont proposés, la *localité* du poste, le *niveau d'étude* et le *secteur d'activité*. Un ensemble de documents est renvoyé dont le nombre peut être très grand. Il serait souhaitable de pouvoir fournir au recruteur un résumé de cet ensemble de documents sous forme de trois classes. Ces trois classes vont permettre au recruteur de catégoriser les candidatures pertinentes (classe A), celles susceptibles d'être pertinentes (classe B) et enfin celles qui sont non pertinentes (classe C). La figure 5.1, illustre le principe de la *summarization* par catégorisation sur l'exemple des CVs.

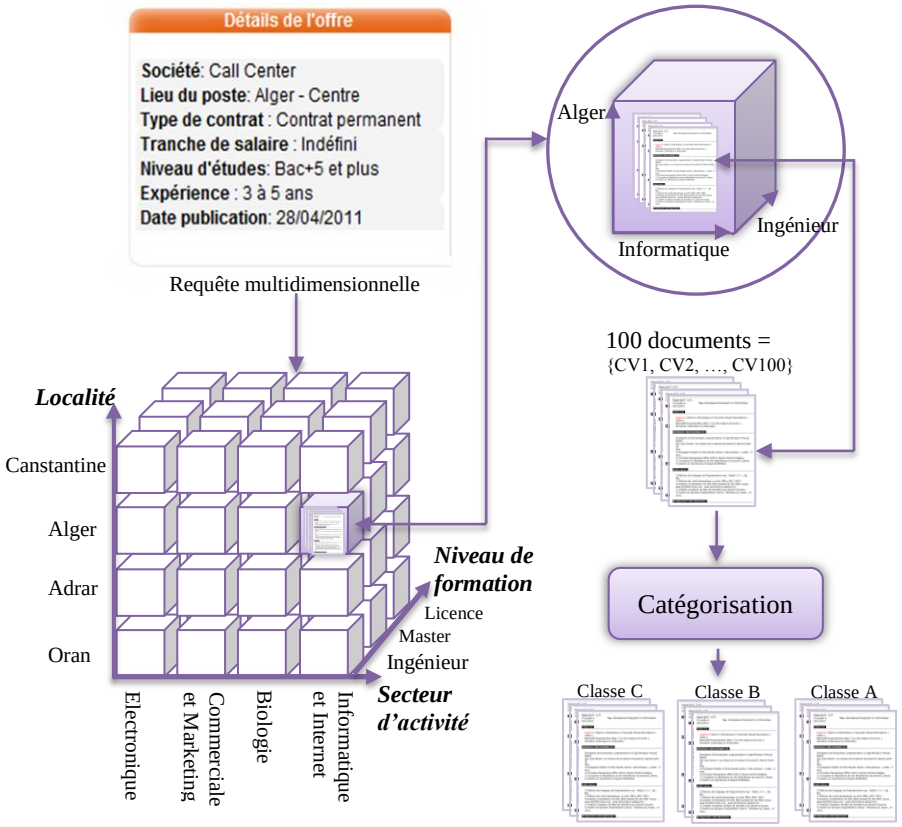


Figure 5.1 : Principe de la *summarization* par catégorisation appliquée aux CVs

5.2.2 Cadre formel

Pour déterminer les *n* agrégats, nous avons adapté au contexte OLAP la technique de la catégorisation de texte par le contenu (c.f section 2.5.3). La catégorisation de texte est une approche supervisée où un expert dirige le processus d'apprentissage. Elle peut être formellement définie comme suit. Étant donné (1) $X^n = \{x_1, x_2, \dots, x_n\}$ l'ensemble des représentations vectorielles des n documents à catégoriser, et (2) $C^m = \{c_1, c_2, \dots, c_m\}$ l'ensemble des classes ou catégories prédéfinies par l'expert pour les besoins d'une application donnée, étant donné aussi (3) un ensemble d'apprentissage S d'objets tés $\langle x, c\rangle \in X^n \times C^m$. En utilisant un algorithme d'apprentissage nous voulons apprendre une fonction dénommée *Classes* qui permet de mapper les objets de l'ensemble X^n sur les classes de l'ensemble C^m.

Définition 1. La fonction *Classes* est définie par

$$\begin{array}{rcll} Classes: & X^n & \longrightarrow & C^m \\ x_i & \longmapsto & c_j & i := 1,..,n\ ;\ j := 1,..,m \end{array}$$
$$\text{avec, } m \text{ et } n \in \mathbb{N}$$

- $X^n = \{x_1, x_2, \dots, x_n\}$ est l'ensemble des représentations vectorielles des n documents à agréger.
- $C^m = \{c_1, c_2, \dots, c_m\}$ est l'ensemble des m classes.
- Si nous notons Γ l'algorithme d'apprentissage utilisé, $Classes = \Gamma(S)$.
- l'algorithme d'apprentissage Γ prend en entrée l'ensemble d'apprentissage S pour retourner la fonction de catégorisation *Classes.*

5.2.3 Cadre théorique

Cette partie présente les différents choix adoptés pour réaliser la tâche de catégorisation. Comme présenté en chapitre 2 (c.f section 2.5.2), plusieurs méthodes de catégorisation de texte existent. La question qui se pose est, quel algorithme choisir pour construire un modèle adapté au problème à résoudre ? Il n'y a pas de méthode générique ayant donné la preuve de sa supériorité, dans tous les cas de catégorisation de texte [80]. Néanmoins, la classe d'algorithmes basés sur la similarité permet de produire de bonnes performances pour les données textuelles [73]. Cette catégorie contient les algorithmes des classifieurs tels que k-NN, ou « basé Centroide » et leurs variantes. Pour ces classifieurs, la classe d'un nouveau document est déterminée en calculant la similarité entre le document test et les

instances individuel (comme dans le cas de k-NN) ou agrégats (comme dans le cas de « basé centroide ») de l'ensemble d'apprentissage. La classe est déterminée en se basant sur la distribution de la classe des plus proches instances ou agrégats.

Notre choix s'est porté sur une méthode simple, efficace et peu couteuse. C'est la méthode « basé centroide» dont les performances ont surpassé celles des algorithmes, k-NN, Bayésien naïf et C4.5, sur un large éventail de jeux de données, d'après l'étude expérimentale réalisée par Han et Karypis [31].

Trois facteurs sont impliqués dans la catégorisation du texte : le modèle de classement utilisé, la représentation du document et la mesure de similarité [72, 40]**.** Avant de voir le modèle de classement utilisé et qui sera présenté dans la section 5.2.3.4, nous allons tout d'abord aborder la préparation des données dans la section 5.2.3.1. La section 5.2.3.2 donne la représentation choisie pour les documents CVs; la mesure de similarité adoptée sera abordée en section 5.2.3.3.

5.2.3.1 Prétraitements des documents CVs

Avant de penser à une représentation pour le CV en vue de le classer, un ensemble de traitements doivent être enchainés sur le texte du document, afin de filtrer les termes pertinents en éliminant les segments de texte (tokens) non pertinents. Nous avons retenu la chaîne ci-dessous, où chaque traitement est considéré comme un filtre. D'autres filtres peuvent être enchaînés pour une analyse plus sophistiquée :

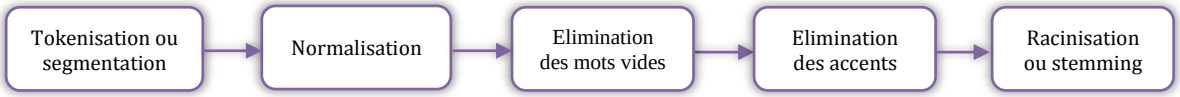


Figure 5.2 : Chaine de traitements d'analyse de texte.

Les principales tâches effectuées concernent :

1. La *tokenisation* ou segmentation du texte ; elle consiste à parcourir le texte en vue de récupérer les termes et supprimer les caractères spéciaux et la ponctuation.
2. La *normalisation* qui réalise la conversion du texte en minuscule.
3. L'*élimination des mots vides***,** consiste à éliminer les mots outils de la langue en utilisant une liste de mots vides pour le français (ex : de, en, dans, etc.).
4. L'élimination des accents, car une partie considérable de la source des erreurs d'orthographe, réside dans l'oubli des accents. En effet, les mots « ingénieur » et

« ingenieur » sont considérés comme étant deux termes différents, ce qui augmente la dimension de l'espace des caractéristiques.

5. La *Racinisation* ou *stemming,* consiste à trouver la racine des verbes fléchis et à ramener les mots pluriels et/ou féminins au masculin singulier.

5.2.3.2 La représentation des documents CVs

Le problème primordial dans la représentation des documents, est le choix des unités textuelles pertinentes; celui-ci dépend du niveau d'analyse sur ces documents. Nous avons choisi de représenter les documents CVs avec une approche « *bag of words* ». L'unité textuelle choisie est donc le terme ou groupe de termes, en utilisant le modèle vectoriel (*Vector-Space Model* ou *VSM*). Dans ce dernier, chaque document d_j est un vecteur des poids des termes dans l'espace des termes.

$$\vec{d}_j = (w_{1j}, w_{2j}, w_{3j} \dots, w_{nj}) \tag{5.1}$$

Où : w_{ij} est le poids du terme t_i dans le document d_j.

Nous avons choisi le schéma de pondération $tf.idf$ [70] pour la pertinence du terme dans le document :

$$w_{ij} = tf_{ij} * idf_{ij} = tf_{i,j}.log\frac{N}{n_i} \tag{5.2}$$

où :

- tf_{ij} est la fréquence du terme t_i dans le document d_j ;
- idf_{ij} est l'inverse de la fréquence des documents dans lesquels le terme t_i apparaît ;
- N est le nombre total de documents dans le corpus ;
- n_i le nombre de documents dans lesquels apparaît le terme t_i.

Pour pallier au problème de la longueur des documents, et ne pas pénaliser les documents courts au dépend des documents longs, chaque vecteur document a été normalisé de sorte que sa longueur $\|\vec{d}_j\| = 1$.

Après prétraitement, l'ensemble des CVs utilisés pour l'expérimentation (décrite en chapitre 6) ont été transformés en des vecteurs de poids des termes en utilisant le schéma de poids $tf.idf$ et ont été normalisés de sorte que leur longueur devienne unitaire.

5.2.3.3 Mesure de similarité

Dans le modèle vectoriel, la similarité entre deux documents d_i et d_j est communément mesurée en utilisant la fonction cosinus donnée par la formule (5.3) :

$$\cos(\vec{d}_i, \vec{d}_j) = \frac{\vec{d}_i . \vec{d}_j}{\|\vec{d}_i\| * \|\vec{d}_j\|} \quad (5.3)$$

Où $\vec{d}_i . \vec{d}_j$ est le produit vectoriel des vecteurs des documents d_i et d_j. Notons que, du moment que les vecteurs ont été normalisés de sorte que leurs longueurs est unitaire la formule (5.3) devient :

$$\cos(\vec{d}_i, \vec{d}_j) = \vec{d}_i . \vec{d}_j \quad (5.3)'$$

5.2.3.4 Principe de l'algorithme « basé centroide »

Comme il a déjà été cité, notre choix s'est porté sur un algorithme simple, peu couteux mais aussi efficace [31, 77, 73], il s'agit de l'algorithme « basé centroide ». Ce dernier a montré des performances qui ont surpassé celles des algorithmes, k-NN, Bayésien naïf et C4.5, sur un large éventail de jeux de données, d'après l'étude expérimentale réalisée par Han et Karypis [31]. Basé sur modèle vectoriel, le principe de ce classifieur est le suivant :

- Pour chaque ensemble de documents appartenant à la même classe, un vecteur centroide est calculé. Si k classes existent dans la base d'apprentissage, k vecteurs centroides sont calculés $\{\vec{C}_1, \vec{C}_2, \vec{C}_3, \dots, \vec{C}_k\}$, où $\vec{C}_i$ est le centroide de la $i^{ème}$classe. Le vecteur centroide, représentant la classe, est aussi appelé prototype de la classe. Le calcul du centroide est donné par la formule (5.4).
- Les similarités entre un nouveau document x et les k centroides sont calculées, en utilisant la mesure du cosinus. Le calcul de ces similarités est donné par formule (5.5).
- Basé sur ces similarités, x est assigné à la classe dont le vecteur centroide est le plus similaire au vecteur document. La classe du document x est donnée par la formule (5.6).

Pour une classe i, étant donné un ensemble S de documents et leurs représentations vectorielles, le vecteur centroide $\vec{C}_i$ est calculé par :

$$\vec{C}_i = \frac{1}{|S|} \sum_{d \in S} \vec{d} \quad (5.4)$$

Où $|S|$ représente le nombre de documents de l'ensemble S.

La similarité entre le vecteur du document d et le centroide $\overrightarrow{C}$ d'une classe, est calculée par la formule ci-dessous :

$$\cos(\vec{d}, \vec{C}) = \frac{\vec{d}.\vec{C}}{\|\vec{d}\| * \|\vec{C}\|} = \frac{\vec{d}.\vec{C}}{\|\vec{C}\|} \tag{5.5}$$

La classe d'un nouveau document x est définie par :

$$arg\ max_{j=1,\dots,k}(\cos(\vec{x}, \vec{C_j})) \tag{5.6}$$

5.2.4 Processus de *Summarization*

En adoptant les choix présentés dans la section précédente et en prenant comme algorithme d'apprentissage Γ, l'algorithme « basé centroide », la fonction *Classes* devient :

Définition 2.

$$Classes : \begin{array}{ccc} X^n & \longrightarrow & C^m \\ x_i & \longmapsto & c_j \end{array} \quad i := 1,..,n\ ;\ j := 1,..,m$$
$$\text{avec, } m\ et\ n \in \mathbb{N}$$

- $X^n = \{x_1, x_2, \dots, x_n\}$ est l'ensemble des représentations vectorielles des n documents à agréger.
- $C^m = \{c_1, c_2, \dots, c_m\}$ est l'ensemble des classes préalablement initialisées par des vecteurs prototypes.
- Soient $\{\overrightarrow{Ctd}_1, \overrightarrow{Ctd}_2, \dots, \overrightarrow{Ctd}_m\}$ les vecteurs prototypes des m classes, où chaque vecteur prototype $\overrightarrow{Ctd}_i$ identifie la classe c_i de façon unique.

$$x_i \longmapsto c_j$$
$$avec\ l'index \quad j = arg\ max_{k=1,\dots,m}\left(\cos(x_i, \overrightarrow{Ctd}_k)\right)$$

où : $\cos(x_i, \overrightarrow{Ctd}_j) = \frac{x_i.\overrightarrow{Ctd}}{\|x_i\|.\|\overrightarrow{Ctd}\|}$

Le processus de *summarization* peut être vu principalement en trois étapes. Étant donné un ensemble de documents des cellules d'un cube de textes correspondant à une requête multidimensionnelle. (1) Pour chaque document il y a lieu de faire : une étape de prétraitements où certains filtrages sont effectués. (2) Une étape de représentation vectorielle où le document est modélisé. (3) Une fois tous les documents vectorisés, on applique

un algorithme de catégorisation où en sortie nous avons les classes A, B et C. L'algorithme 5.1, donne les étapes de *summarization* par catégorisation.

Entrées : S = un ensemble de n documents (ou sections de documents) des cellules d'un cube de textes *TC* correspondant à une requête *R*, multidimensionnelle.
Sorties : les classes : C_1 (classe A) , C_2 (classe B) et C_3 (classe C).

$X := \emptyset$ // Ensemble des vecteurs des documents $d_i \in S$

Pour chaque document $d_i \in S$, $i := 1,..,n$
Faire

Étape 1 : Prétraitements

Appliquer le filtre de $tokensisation$ au document d_i ;
Appliquer le filtre de $normalisation$ au document d_i;
Appliquer au document d_i le filtre qui supprime les mots vides en utilisant une liste de mots vides ;
Appliquer au document d_i le filtre qui supprime les accents ;
Appliquer au document d_i le filtre qui réalise l'opération de *stemming* ;

Étape 2 : Vectorisation des documents

Pour chaque terme t_j de d_i $j := 1,..,m$ où m est la taille de l'espace des caractéristiques.
Faire

Estimer tf_j, la pondération locale du terme t_j ;
Estimer idf_j, la pondération globale du terme t_j ;
Estimer le poids du terme t_j en utilisant formule 5.2 ; $w_j = tf_j * \log(\frac{n}{idf_j})$;
Ajouter le couple $\langle t_j, w_j \rangle$ à la représentation vectorielle x_i de d_i

Fin_faire
$X := X \cup \{x_i\}$

Fin_Faire

Étape 3 : Catégorisation

Calculer $\overrightarrow{Ctd}_1, \overrightarrow{Ctd}_2$ *et* $\overrightarrow{Ctd}_3$ qui sont respectivement les vecteurs centroides des classes 1, 2 et 3, en utilisant la formule 5.4.

Pour chaque vecteur x_i de X
Faire

Calculer l'index $j := arg\ max_{k=1,..,3}(\cos(x_i, \overrightarrow{Ctd}_k))$;
Classes $(x_i) := C_j$; //définition formelle 2.

Fin_faire

Retourner C_1, C_2 *et* C_3

Algorithme 5.1 : Calcul des agrégats par catégorisation

5.3 Fonction *Clusters*

Le principe de base de la *summarization* par partitionnement est d'exploiter la mesure textuelle contenue dans le cube de textes pour présenter au décideur un résumé sous forme de *k* partitions. Le paramètre *k* est défini par l'utilisateur selon l'application et les besoins d'analyse. Nous verrons dans le prochain chapitre qu'il est possible d'aider le décideur à définir le paramètre *k* optimal, permettant de donner des agrégats de qualité. Nous présentons dans la section suivante un scénario de l'étude de cas déjà vue dans le chapitre 4, pour ce type de *summarization*.

5.3.1 Exemple

Notre exemple pour cette fonction se base sur le scénario 2 (c.f section 4.3.2). Dans ce scénario, le décideur voudrait découvrir des groupes homogènes dans un ensemble de CVs. Par exemple, le décideur voudrait analyser les tendances en matière de formation dans les universités algériennes. La requête peut être (Localité = "Centre", secteur d'activité = "Informatique", niveau d'étude = "Bac + 5 et plus", section = "Formation"). Dans le modèle multidimensionnel proposé, l'indicateur à observer est toujours le CV. Un ensemble de granules (sections « Formation ») des documents CVs, est renvoyé. Cet ensemble est le résultat d'une agrégation classique sur les axes d'analyses *localité, niveau d'étude* et *secteur d'activité*, en ne considérant que la section « Formation » du CV. Un résumé de cet ensemble de granules de documents est fourni au décideur sous forme de *k* groupes (clusters). Les groupes obtenus sont le résultat de la technique de *clustering* appliquée aux textes retournés, correspondant à une requête multidimensionnelle.

5.3.2 Cadre formel

Pour déterminer les *k* agrégats, nous avons adapté au contexte OLAP la technique de *clustering* de documents (c.f section 2.5.3). Nous définissons la tâche de *clustering* comme suit. Étant donné (1) un ensemble des représentations vectorielles des n ments $X^n = \{x_1, x_2, \dots, x_n\}$, (2) un nombre souhaité de clusters *k*, (3) une fonction objective qui évalue la qualité du *clustering*, nous cherchons à calculer l'affectation suivante :

Définition 3.La fonction *Clusters* est définie par

$$Clusters : \begin{array}{ll} X^n \longrightarrow \{1, 2, \dots, k\} & \text{avec, } n \text{ et } k \in \mathbb{N} \\ x_i \longmapsto j & \text{et } i := 1,..,n \; ; j := 1,..,k \end{array}$$

- $X^n = \{x_1, x_2, \dots, x_n\}$ est l'ensemble des représentations vectorielles des n documents à agréger.
- $\{1, 2, \dots, k\}$ les numéros de clusters.
- La fonction *Clusters* doit maximiser (ou minimiser selon l'algorithme utilisé) une fonction objective.
- La fonction objective est le plus souvent définie en termes de similarité ou distance entre les documents.

5.3.3 Cadre théorique

Cette partie présente les différents choix adoptés pour la tâche de *clustering*. Nous avons vu au chapitre 2, que selon la méthode de *clustering* utilisé, le type de résultat obtenu peut être différent. En effet, les clusters obtenus peuvent être des ensembles durs ou flous, le résultat peut être plat ou peut se présenter sous la forme d'une hiérarchie. Nous avons vu aussi que plusieurs familles de méthodes existent. Pour valider la fonction *Clusters*, nous avons choisi un algorithme de *clustering* plat donnant des ensembles durs. Cet algorithme fait partie des méthodes basées sur les prototypes, de la famille de méthodes utilisant une distance. Il s'agit de l'algorithme des K-Means [46]. Comme déjà cité, l'algorithme des K-Means est la méthode de partitionnement la plus connue et la plus utilisée dans divers domaines d'application. Ce succès est dû au fait que cet algorithme présente un rapport coût/efficacité avantageux.

Le processus de *clustering* se divise en trois étapes majeures : (1) la préparation des données, (2) l'algorithme de clustering et (3) l'évaluation du schéma de clustering obtenu. Avant de voir l'algorithme choisi en section 5.3.3.2, nous allons décrire la préparation des données en section 5.3.3.1, l'évaluation du résultat de clustering obtenu sera discutée dans le chapitre 6.

5.3.3.1 Préparation des données

Comme pour la catégorisation de texte, la préparation des données suppose l'application de deux étapes : les prétraitements et la vectorisation des documents.

Pour les prétraitements, nous avons retenu les mêmes filtres qu'en catégorisation de texte (voir section 5.2.3.1), à savoir la segmentation du texte, la normalisation, l'élimination des mots vides, l'élimination des accents et le *stemming*. Pour la vectorisation des documents nous avons aussi gardé la même représentation qu'en catégorisation de texte (section 5.2.3.2), à savoir l'approche « *bag of words* » dans un espace vectoriel. Le schéma de pondération utilisé est $tf.idf$ (formule 5.2). De plus, les vecteurs documents ont été normalisés.

5.3.3.2 Principe de l'algorithme K-Means

L'algorithme K-means cherche un partitionnement des documents en se basant sur les centroides des clusters. Plusieurs variantes de cet algorithme existent dans la littérature. La variante de K-means utilisant la mesure de similarité cosinus, aussi appelée K-means sphérique (Spherical K-Means, SPKM) [89, 90], est une méthode populaire pour le clustering de données textuelles. En effet, pour les données à grande dimensionnalité, telle que c'est le cas pour les documents textuels, la mesure cosinus a prouvé d'être plus efficace que la distance euclidienne [90]. De plus, comme les vecteurs documents et centroides sont normalisés, l'algorithme est aussi appelé K-means sur une hypersphère unitaire dans R^m où m est la taille de l'espace des caractéristiques. Le principe de SPKM est le suivant :

Soit $X = \{x_1,\ x_2, \dots,\ x_n\} \in R^m$ l'ensemble des vecteurs unitaires des n documents à partitionner, et k le nombre de clusters désiré.

1. Initialisation par tirage aléatoire dans X, de k centroides $\mu_{,1} \dots, \mu_k$
2. Constitution d'un schéma de partition $\mathsf{C}_t = \{C_1, \dots, C_k\}$ par allocation de chaque objet $x_i \in X$ au centre le plus proche, où : $C_j = \{x_i \in X | \ j = arg\ \max_{h=1,\dots,k} x_i\, \mu_h)\}$
3. Recalculer les centroides μ_j des k classes, en utilisant la formule ci-dessous :

$$\mu_j = \frac{s_j}{\|s_j\|} \ avec \quad s_j = \sum_{x_i \in C_j} x_i \qquad 1 \leq\ j \leq k \qquad (5.7)$$

4. Répéter les étapes 2 et 3 jusqu'à ce que la fonction objective est localement optimisée. Ou jusqu'à un nombre τ d'itérations.
5. Retourner la partition finale C_{finale}

Comme déjà cité en chapitre 2 (section 2.5.3.3), K-means standard utilise la somme de l'erreur au carré (formule 2.17) comme fonction objective, qu'il doit minimiser pour avoir

des clusters compacts. De façon similaire, en utilisant la mesure cosinus, SPKM utilise la fonction objective définie par la formule ci-dessous à maximiser [89].

$$\varphi(\mathrm{C}) = \sum_{j=1}^{k} \sum_{x_i \in C_j} x_i \, \mu_{k(x)} \tag{5.8}$$

Où $k(x) = arg\max_k x_i \, \mu_k$ est l'index du centroide le plus proche au vecteur document x_i.

5.3.4 Processus de *Summarization*

En prenant en considération les choix de la section précédente, la définition 3 (section 5.3.2) devient :

Définition 4.

$$Clusters : \begin{array}{ll} X^n \longrightarrow \{1, 2, \dots, k\} & \text{avec, } n \, et \, k \in \mathbb{N} \\ x_i \longmapsto j & \text{et } i := 1, .., n \, ; j := 1, .., k \end{array}$$

- $X^n = \{x_1, x_2, \dots, x_n\}$ est l'ensemble des représentations vectorielles des n documents à agréger.
- $\{1, 2, \dots, k\}$ les numéros de clusters.
- Soit $\mu_1, \dots, \mu_k$ les centroides des k classes.

$$x_i \longmapsto arg\, max_{j=1,\dots,k} \, x_i \, \mu_j$$

Comme pour la *summarization* par catégorisation, le processus de *summarization* par *clustering* peut être vu principalement en trois étapes. Étant donné un ensemble de documents (ou granules de documents) des cellules d'un cube de textes correspondant à une requête multidimensionnelle. Soit k le nombre de clusters désiré. Une étape de préparation des données, qui consiste en (1) une étape de prétraitements et (2) une étape de représentation vectorielle. (3) Une fois tous les documents vectorisés, on applique un algorithme de *clustering* dans notre cas K-means sphérique qui, rappelons-le est itératif. En sortie nous avons un schéma de *clustering* final :

$$\mathrm{C}_{finale} = \{C_1, \dots, C_k\} \Big| \bigcup_{j=1}^{k} C_j = \{x_1, x_2, \dots, x_n\} \quad et \quad \mu_j \cap \mu_l = \emptyset, \quad j \neq l$$

Le processus de *summarization* par *clustering* est donné par l'algorithme 5.2.

Entrées : S = un ensemble de n documents (ou sections de documents) des cellules d'un cube de textes *TC* correspondant à une requête *R*, multidimensionnelle. k le nombre de clusters désiré.
Sorties : schéma de *clustering* $\mathrm{C}_{finale} = \{C_1, \dots, C_k\}$

$X := \emptyset$ // Ensemble des vecteurs des documents $x_i \in X$

Pour chaque document $d_i \in S$, $i := 1, .., n$
Faire
Étape 1 : Prétraitements
Appliquer le filtre de $tokensisation$ au document d_i ;
Appliquer le filtre de $normalisation$ au document d_i;
Appliquer au document d_i le filtre qui supprime les mots vides en utilisant une liste de mots vides ;
Appliquer au document d_i le filtre qui supprime les accents ;
Appliquer au document d_i le filtre qui réalise l'opération de *stemming* ;

Étape 2 : Vectorisation des documents
Pour chaque terme t_j de d_i $j := 1, .., m$ où m est la taille de l'espace des caractéristiques.
Faire
Estimer tf_j, la pondération locale du terme t_j ;
Estimer idf_j, la pondération globale du terme t_j ;
Estimer le poids du terme t_j en utilisant formule 5.2 ; $w_j = tf_j * \log(\frac{n}{idf_j})$;
Ajouter le couple $\langle t_j, w_j \rangle$ à la représentation vectorielle x_i de d_i
Fin_faire
$X := X \cup \{x_i\}$
Fin_Faire

Étape 3 : *Clustering*

Initialisation par tirage aléatoire dans X, des k centroides $\mu_1, \dots, \mu_k$
Répéter
Calculer $\mu_1, \dots, \mu_k$ //en utilisant la formule 5.7.
Pour chaque vecteur x_i de X
Faire
Clusters$(x_i) := arg\ max_{j=1,\dots,k}(\cos(x_i, \mu_j))$; //définition 4
Mettre x_i dans C_j ;
Fin_faire

Jusqu'à maximiser $\varphi(\mathrm{C}) = \sum_{j=1}^{k} \sum_{x_i \in C_j} x_i \mu_j$

Retourner $\mathrm{C}_{finale} = \{C_1, \dots, C_k\}$;

Algorithme 5.2 : calcul des agrégats par *clustering*

5.4 Fonction *Top_Keyphrases*

La fonction *Top_KeyPhrases* permet l'agrégation d'un ensemble de documents en principaux thèmes. En effet, dans une collection[20] de documents, une information primordiale qui pourrait rajouter de la connaissance pour l'utilisateur serait de connaître les différents thèmes présents dans cette collection. Ces thèmes définissent ce qui est aussi appelé la macrostructure du document [96]. Il est intéressent à noter ici que même les documents peu structurés (ou même non structurés) possèdent une macrostructure qu'il est possible de décrire en identifiant les principaux thèmes véhiculés par le contenu de ces documents. Nous présentons dans la section suivante un scénario de l'étude de cas présenté dans le chapitre 4, pour ce type de *summarization*.

5.4.1 Exemple

Notre exemple pour cette fonction se base sur le scénario 3 présenté en section 4.3.2. Dans ce scénario, le décideur voudrait connaître les compétences les plus courantes dans un secteur donné et une région donnée. Il voudrait, par exemple, voir si les compétences présentes dans les CVs, sont en rapport avec celles attendues. La requête soumise par le décideur peut être (Localité = "Centre", secteur d'activité = "Informatique", section = "compétences"). Dans le modèle multidimensionnel proposé, l'indicateur à observer est toujours le CV. Un ensemble de granules (sections "compétences") des documents CVs, est renvoyé. Celui-ci est le résultat d'une agrégation classique sur les axes d'analyse *localité* et *secteur d'activité*, en ne considérant que la section « compétences » du CV. Un résumé de cet ensemble de granules est fourni sous forme de *k* multi-mots (en anglais *keyphrases*) les plus pertinents. *k* est un entier qui peut être défini par le décideur ou automatiquement.

5.4.2 Cadre formel

Pour déterminer les *k* agrégats, nous faisons appel à la tâche d'extraction de descripteurs en utilisant des méthodes statistiques ou linguistiques (c.f section 2.4.2). Nous avons choisi comme descripteurs les *keyphrases* aussi appelés multi-mots clés ou encore concepts clés, en utilisant une méthode statistique. Nous définissons formellement l'extraction des multi-mots par la fonction $Keyphrases$ comme suit. Étant donné un ensemble de documents $S^n = \{d_1, d_2, \dots, d_n\}$, nous cherchons tout d'abord à extraire tous les multi-mots dans les documents à agréger.

[20] Dans le cas de l'OLAP, la collection est définie par l'ensemble de documents à agréger.

Définition 5. La fonction *Keyphrases* est définie par

$$Keyphrases : \ S^n \longrightarrow P^m \ \text{avec,} \ m \ et \ n \in \mathbb{N}$$
$$\{d_1, d_2, \dots, d_n\} \longmapsto \{p_1, p_2, \dots, p_m\}$$

- $S^n = \{d_1, d_2, \dots, d_n\}$ est l'ensemble des n documents à agréger.
- $P^m : \langle p_1, p_2, \dots, p_m \rangle$ est un ensemble de multi-mots, apparaissant dans au moins σ documents. Où $\sigma \in \mathbb{N}$ est un paramètre à définir.

Une fois tous les multi-mots déterminés, la fonction d'agrégation $Top\_Keyphrases$ prend en entrée ces multi-mots pour fournir un ensemble d'agrégats, les *k* multi-mots les mieux classés. La définition formelle de la fonction $Top\_Keyphrases$ est donnée ci-dessous.

Définition 6. La fonction *Top_Keyphrases* est définie par

$$Top\_Keyphrases : \ P^m \longrightarrow T^k \ \text{avec,} \ m \ et \ n \in \mathbb{N}$$
$$\{p_1, p_2, \dots, p_m\} \longmapsto \langle t_1, t_2, \dots, t_k \rangle$$

- $P^m : \langle p_1, p_2, \dots, p_m \rangle$ est l'ensemble de multi-mots fournis par la fonction *Key-Phrases.*
- $T^k : \langle t_1, t_2, \dots, t_k \rangle$ est l'ensemble des multi-mots les mieux classés de l'ensemble $P^m, k \leq m.$
- Dans l'ensemble T^k, les k multi-mots choisis constituent les thèmes les plus représentatifs de l'ensemble des documents à agréger.

5.4.3 Cadre théorique

Nous avons fondé la tâche d'identification des *keyphrases* sur une approche statistique. L'avantage des méthodes statistiques est qu'elles sont indépendantes du domaine et de la langue. Ce qui permet de les adapter plus facilement à différents domaines, différentes langues.

Notre choix s'est porté, sur la méthode des séquences de mots fréquentes (*Frequent word Sequences*, FS) [3, 95, 96]. La méthode de découverte des FSs a été d'abord proposée par Ahonen [99]. Celle-ci a aussi été appliquée dans la détection de thèmes [95], la catégorisation de texte [98], le *clustering* hiérarchique [97] ainsi que le résumé automatique de texte [96].

Pour considérer qu'une FS est représentative d'un thème dans la collection de documents, trois étapes sont nécessaires : étant donné un ensemble de documents et après une phase de prétraitement, (1) tous les FSs sont découvertes en utilisant l'algorithme de Ahonen [99, 3]. Celles-ci représentent l'ensemble $P^m : \langle p_1, p_2, \dots, p_m \rangle$ de la fonction *Keyphrases* (c.f définition 5). (2) Les FSs trouvées sont ensuite regroupées dans des classes d'équivalence, où chaque classe d'équivalence est considérée comme représentative d'un thème. (3) Les classes d'équivalence sont ensuite ordonnées selon le score moyen des FSs qu'elles contiennent. Les k classes d'équivalence les mieux classées représentent l'ensemble $T^k : \langle t_1, t_2, \dots, t_k \rangle$ de la fonction de *summarization* $Top\_Keyphrases$ (c.f définition 6). Les sections suivantes sont consacrées à chacune de ces trois étapes.

5.4.3.1 Découverte des FSs

Une séquence de mots est dite fréquente si elle apparaît dans au moins σ documents, où σ est un seuil de support pré-spécifié. Le but de l'algorithme est de trouver toutes les séquences de mots fréquentes dans l'ensemble de données textuelles. L'algorithme de la figure 5.2 décrit les étapes du processus de découverte des FSs.

L'algorithme de découverte des FSs prend en entrée un ensemble de documents prétraités[21] ainsi que le paramètre de fréquence σ. Il fournit en sortie l'ensemble de toutes les séquences de mots fréquentes, FSs. Le processus est principalement composé de deux phases. Une première qui consiste à collecter, tous les bi-grams fréquents, c.-à-d. les FSs de longueur = 2. Dans la deuxième phase, appelée « phase de découverte », les FSs de la première phase sont étendues avec un mot de plus pour former des séquences de longueur = 3. Toutes les FSs de longueur = 3 sont ensuite étendues. Ce processus est itératif jusqu'à ce qu'aucune FS ne soit candidate à l'expansion. Le paramètre de fréquence σ est choisi selon la taille de la collection de documents.

[21] Les mêmes prétraitements que pour la catégorisation et le *clustering* sont retenus.

```
Input:      D: a set of pre-processed documents, σ: a frequency threshold
Output:     Fs: a set of frequent word sequences
        // Initial phase: collecting all frequent pairs
1.      For all the documents  d ∈ D
2.          Collect all the ordered pairs and occurrence information within d
3.      End For
4.      Seq2 = all the ordered word pairs that appear in at least σ documents in D
        // Discovery phase: building longer word sequences
5.      k := 2
6.      Fs := Seq2
7.      While Seqk is not void
8.          For all phrases  s ∈ Seqk
9.              Let l be the length of the sequence s
10.             Find all the sequences  s' such that s is a subsequence of  s' …
                and the length of  s' is l+1
11.             For all  s'
12.                 If  s'  appears in at least σ documents in D
13.                     S := S ∪ {s'}
14.             End For
15.             Fs := Fs ∪ S
16.             Seqk+1 :=  Seqk+1 ∪ S
17.         End For
18.         k := k + 1
19.     End While
20.     Return Fs
```

Figure 5.3 : Algorithme de découverte de toutes les FSs dans une collection de documents, extrait de [96]

5.4.3.2 Classes d'équivalence

Après extraction de toutes les FSs, ces dernières seront groupées dans des classes d'équivalence [3, 95] en se basant sur leurs co-occurrences, les unes avec les autres. Le but est de réduire la redondance ainsi que le nombre de FSs découvertes. La génération des classes d'équivalence requiert en entrée :

- L'ensemble des FSs retournées par l'algorithme de découverte (figure 5.3);
- Un paramètre de confidence qui permet d'évaluer la co-occurrence entre deux FSs.

Soient A et B, 2 FSs de l'ensemble des FSs découvertes. La classe d'équivalence de A, Eq_A, contient l'ensemble des FSs qui co-occurrent avec A dans les documents où ils apparaissent, selon un paramètre de confidence. Pour 2 FSs A et B, si :

$$\frac{fréquence(A,\ B\ co{-}occur)}{fréquence(A)} \geq confidence \qquad (5.9)$$

Où :

- $fréquence(A)$ est le nombre de documents dans lesquels la FS A apparaît;

- et $fréquence(A,\ B\ co-occur)$ est le nombre de documents dans lesquels A et B co-occurrent.

Alors B est ajouté à un ensemble appelé Det_A, et qui représente l'ensemble des FSs déterminées par A. Les autres FSs sont testées de la même manière et sont rajoutées à l'ensemble Det_A, s'ils satisfont la règle (5.9). Eq_A, la classe d'équivalence de A, est ainsi formée de toutes les FSs X tel que $Det_X = Det_A$.

5.4.3.3 Obtention des Top *k* thèmes

Chaque classe d'équivalence est considérée comme représentative d'un thème. Les classes d'équivalences sont ordonnées selon le score moyen de leurs FSs. Les expérimentations menées par Zhan [96] montrent que le schéma de pondération tf donne de meilleurs résultats que le schéma *tf-idf*. Zhan propose aussi de combiner au schéma de pondération tf d'autres paramètres tel que la longueur de la FS [96]. En se basant sur ces deux paramètres (tf et, longueur de la FS), le score d'une FS devient :

$$score = tf.\log_2 l \tag{5.10}$$

où :

- tf est la fréquence de la FS dans l'ensemble des documents de la classe d'équivalence.
- l la longueur de la FS.

Ainsi, les *k* classes d'équivalence les mieux classées sont représentatives des top *k* thèmes de l'ensemble de la collection des documents.

5.4.4 Processus de *Summarization*

Le processus de *summarization*, en utilisant les *k top keyphrases* (thèmes), peut être vu principalement selon quatre étapes. Étant donné un ensemble de documents (ou granules de documents) des cellules d'un cube de textes correspondant à une requête multidimensionnelle. Étant donné *k* le nombre de thèmes désiré. (1) Une étape de prétraitements permet d'appliquer les même filtres qu'en section 5.2.3.1. (2) Suit une étape de découverte de toutes les séquences de mots fréquentes, FSs, en utilisant l'algorithme de la figure 5.3. (3) une étape assurera l'identification des classes d'équivalence. (4) Enfin, une dernière étape qui effectue le ranking des thèmes représentés par les classes d'équivalence. En sor-

tie nous retenons les k thèmes les mieux classés représentés par l'ensemble T^k : $\langle t_1, t_2, \dots, t_k \rangle$. Ce processus est donné par l'algorithme 5.3.

Entrées : S = un ensemble de n documents (ou sections de documents) des cellules d'un cube de textes *TC* correspondant à une requête *R*, multidimensionnelle. k le nombre de thèmes désiré.
Sorties : Les top *k* thèmes $\langle t_1, t_2, \dots, t_k \rangle$.

$S' := \emptyset$ // Ensemble des documents d_i après prétraitements.
$T := \emptyset$ // Liste ordonnée de tous les thèmes

Étape 1 : Prétraitements

Pour chaque document $d_i \in S$, $i := 1, .., n$
Faire
- Appliquer le filtre de $tokensisation$ au document d_i ;
- Appliquer le filtre de $normalisation$ au document d_i;
- Appliquer au document d_i le filtre qui supprime les mots vides en utilisant une liste de mots vides ;
- Appliquer au document d_i le filtre qui supprime les accents ;
- Appliquer au document d_i le filtre qui réalise l'opération de *stemming* ;
- $S' := S' \cup \{d_i\}$.

Fin_Faire

Étape 2 : Découverte des FSs dans l'ensemble S'

$Keyphrases(S') = \{p_1, p_2, \dots, p_m\}$

//la fonction $Keyphrases$ utilise l'algorithme de la figure 5.3 pour fournir l'ensemble P^m de toutes les FSs.

Étape 3 : identification des classes d'équivalence

$EQ := \emptyset$ //l'ensemble des classes d'équivalence

Pour chaque FS $p_i \in P^m$, $i := 1, .., m$
Faire
- Calculer Det_{p_i} qui est l'ensemble des FSs déterminées par p_i, en utilisant la formule 5.9.
- $Eq_{p_i} := \{p_i\}$

Fin_faire

Pour chaque FS $p_i \in P^m$, $i := 1, .., m$
Faire
- Pour chaque FS p_j $i \neq j$
  Faire
  - Si $Det_{p_i} = Det_{p_j}$ Alors $Eq_{p_i} = Eq_{p_i} \cup \{p_j\}$
  Fin_faire
- Si $|Eq_{p_i}| > 1$ alors $EQ := EQ \cup Eq_{p_i}$

Fin_faire

Étape 4 : Ranking des classes d'équivalence

Pour chaque $Eq_l \in EQ$
Faire
- Calculer score moyen de toutes les FSs $\in Eq_l$
- $T := T \cup \{\langle score, t_l \rangle\}$.

//le score de chaque FS est calculé par la formule 5.10

Fin_faire

$T^k := \langle t_1, t_2, \dots, t_k \rangle$
Retourner T^k

//l'ensemble T^k est formé des *k* premiers thèmes de l'ensemble T.

Algorithme 5.3 : calcul des agrégats sous forme des *k* top thèmes.

5.5 Fonction *Summary*

La fonction *Summary* permet l'agrégation d'un ensemble de documents en une représentation condensée de leurs sources. Il s'agit des phrases les plus pertinentes extraites des textes des documents à agréger. Nous utiliserons à cet effet une approche du domaine du résumé automatique que nous avons présenté dans le chapitre 2 (section 2.6). Dans la section suivante, nous présentons un scénario de l'étude de cas présenté dans le chapitre 4, pour ce type de *summarization*.

5.5.1 Exemple

Notre exemple pour cette fonction se base sur le scénario 4 présenté en section 4.3.2. Dans celui-ci, le décideur voudrait analyser l'expérience professionnelle des candidats dans une région donnée par rapport à un niveau d'étude précis. Il voudrait, par exemple, savoir quels secteurs d'activités sont prédominants dans cette région. La requête soumise par le décideur peut être (Localité = "Centre", niveau d'étude = "Bac+5", section = "Expériences professionnelles"). Dans le modèle multidimensionnel que nous avons proposé pour l'analyse des CVs où l'indicateur à observer est le CV, un ensemble de granules (sections "Expériences professionnelles") des documents CVs, est renvoyé. La *summarization* de cet ensemble de granules se présente sous forme de segments textuels les plus pertinents. Un résumé de texte indicatif est ainsi fourni au décideur.

5.5.2 Cadre formel

Afin d'obtenir le résumé de texte comme agrégat, nous utilisons une méthode de résumé automatique. Soit $S^n = \{d_1, d_2, \dots, d_n\}$ l'ensemble des n documents à agréger. Après prétraitements, les documents sont segmentés pour obtenir un ensemble de segments textuels. Ces derniers (généralement des phrases), candidats pour construire le résumé, sont sélectionnés en se basant sur des critères tels que : la fréquence des mots, la position des phrases, la présence des mots des titres dans ces segments, etc. (voir chapitre 2, §2.6.3.1). La valeur de pertinence d'une phrase est exprimée en combinant les valeurs des critères choisis sous forme d'une fonction linéaire. Les phrases qui ont les scores de pertinence les plus élevés sont celles qui seront extraites en premier pour construire le résumé. La définition formelle de la fonction *Summary* est donnée ci-dessous.

Définition 7. La fonction *Summary* est définie par

$$Summary : \ S^n \ \longrightarrow \ St^m \ \text{ avec, } m \, et \, n \in \mathbb{N}$$
$$\{d_1, \ d_2, \dots, d_n\} \ \longmapsto \langle s_1, s_2, \dots, s_m \rangle$$

- $S^n : \{d_1, \ d_2, \dots, d_n\}$ est l'ensemble des n documents à agréger.
- $St^m : \langle s_1, s_2, \dots, s_m \rangle$ est un ensemble de segments textuels les plus pertinents.

5.5.3 Cadre théorique

L'une des méthodes de résumé automatique multi-documents les plus couramment utilisée, est le résumé par partitionnement (*clustering-summarization*). Il s'agit d'une approche par extraction dont l'avantage est d'être indépendante du domaine d'application. Cette méthode consiste à regrouper les documents sources dans des clusters où chacun représente un thème pour la collection de documents. Un résumé est généré pour chaque cluster (chaque thème) et est fourni à l'utilisateur qui pourra décider, quel thème présente un intérêt pour lui. Cette approche peut se résumer selon les étapes suivantes :

Étape 1 : prétraitements;
Étape 2 : *clustering* de documents;
Étape 3 : identification des thèmes pour chaque cluster;
Étapes 4 : extraction de segments textuels les plus pertinents avec réduction de la redondance.

Cette approche suppose que les thèmes des documents sont distribués sans de façon non recouvrante à travers les clusters. Cette assertion n'est pas toujours vraie, ce qui est le cas pour notre collection de CVs. En effet, un même document peut traiter plusieurs thèmes différents (un thème peut être une compétence candidat) et donc peut faire partie de plusieurs clusters. Pour cette raison, nous avons opté pour la méthode de Zhan [96] qui se base sur l'utilisation des classes d'équivalence [99]. Les étapes précédentes deviennent alors :

Étape 1 : prétraitements : prétraitement des documents en appliquant les mêmes filtres (c.f section 5.2.3.1).

Étape 2 : découverte des FSs et des classes d'équivalence : en utilisant la même démarche (c.f sections 5.4.3.1 et 5.4.3.2).

Étape 3 : Ranking des classes d'équivalence : comme pour la fonction *Top_KeyPhrases* (section 5.4.3.3), les classes d'équivalences sont ordonnées selon le

score moyen de leurs FSs. La valeur de pertinence d'une FS est estimée sur la base de deux paramètres, la fréquence du terme (tf) et la longueur (l) de la FS, en utilisant la formule 5.10.

Étapes 4 : Génération d'un résumé pour chaque classe d'équivalence : chaque classe d'équivalence est considérée comme représentative d'un thème. La figure 5.4 montre un exemple de structure thématique d'une collection de documents où chaque classe d'équivalence garde la trace des documents dans lesquels les thèmes apparaissent. L'avantage de la méthode des classes d'équivalence est qu'un document peut être associé à plusieurs thèmes comme le cas des documents, CV1, CV7, CV50.

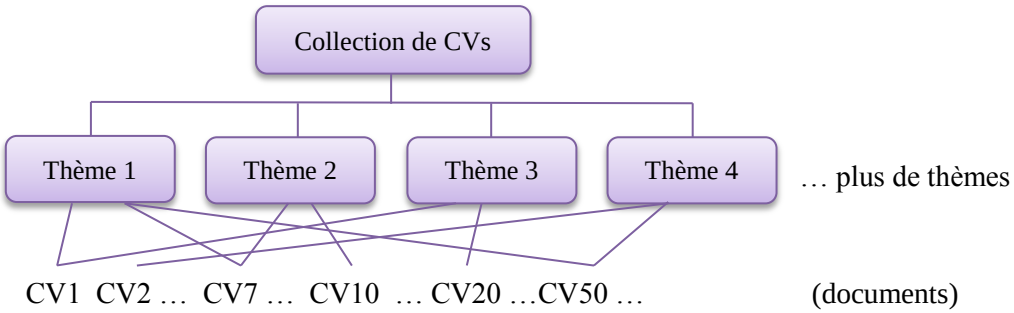


Figure 5.4 : Exemple de la structure thématique d'une collection de documents en utilisant les classes d'équivalence

Un résumé est généré pour chaque classe d'équivalence (chaque thème) en prenant de chaque document de la classe, les phrases contenant le thème représenté par la classe. Ainsi, nous retenons un seul critère dans le choix des phrases, celui de la présence du thème dans la phrase. Pour identifier la redondance dans les phrases choisies, nous utilisons la méthode MMR (*Maximale Marginale Relevance*) [14]. C'est une méthode efficace pour réduire la redondance tout en maximisant la diversité des passages sélectionnés.

5.5.4 Processus de *Summarization*

Le processus de *summarization*, utilisant le résumé de texte, peut être vu principalement selon quatre étapes. Étant donné un ensemble de documents (ou granules de documents) des cellules d'un cube de textes correspondant à une requête multidimensionnelle. Étant donné m le taux de compression désiré. (1) Une étape de prétraitements est entamée en appliquant les même filtres qu'en section 5.2.3.1. (2) Elle est suivie par une étape de découverte de toutes les FSs, en utilisant la fonction $Keyphrases$ (3) une étape d'identification des classes d'équivalence et de ranking de ces classes est déployée, en uti-

lisant la fonction $Top\_Keyphrases$. (4) Enfin, une étape de génération d'un résumé pour chaque thème clos cette tâche. Ces étapes sont données par l'algorithme 5.4.

Entrées : S = un ensemble de n documents (ou sections de documents) des cellules d'un cube de textes *TC* correspondant à une requête *R*, multidimensionnelle. m le taux de compression (nombre de phrases) désiré.
Sorties : un résumé pour chaque thème t_i (des k premiers thèmes), composé des m phrases $\langle s_{i1}, s_{i2}, \dots, s_{im} \rangle$ les plus pertinentes.

$S' := \emptyset$ // Ensemble des documents d_i après prétraitements.
$St := \emptyset$ //Ensemble des résumés pour les *k* premiers thèmes

Étape 1 : Prétraitements

Pour chaque document $d_i \in S$, $i := 1, .., n$
Faire
- Appliquer le filtre de $tokensisation$ au document d_i ;
- Appliquer le filtre de $normalisation$ au document d_i;
- Appliquer au document d_i le filtre qui supprime les mots vides en utilisant une liste de mots vides ;
- Appliquer au document d_i le filtre qui supprime les accents ;
- Appliquer au document d_i le filtre qui réalise l'opération de *stemming* ;
- $S' := S' \cup \{d_i\}$

Fin_Faire

Étape 2 : Découverte des FSs dans l'ensemble S'

$Keyphrases(S') = \{p_1, p_2, \dots, p_m\}$

Étape 3 : Identification des classes d'équivalence et ranking des classes identifiées

$Top\_Keyphrases(\{p_1, p_2, \dots, p_m\}) = \langle t_1, t_2, \dots, t_k \rangle$

Étape 4 : Génération du résumé pour chaque thème

Pour chaque thème t_i, $i := 1, .., k$
Faire
- $St_i := \emptyset$ //l'ensemble des *m* phrases formant le résumé du thème t_i
- $S := \emptyset$ // l'ensemble des phrases contenant le thème t_i
- Pour chaque document du thème t_i
  Faire
  - Segmenter le document en phrases
  - Pour chaque phrase s_j ; Si phrase contient le thème t_i alors $S \cup \{s_j\}$

  Fin_faire
- Calculer la valeur MMR pour toutes les phrases dans S. //formule 2.29
- $St_i := \langle s_{i1}, s_{i2}, \dots, s_{im} \rangle$ les *m* phrases ayant la valeur MMR la plus élevée
- $St := St \cup St_i$.

Fin_faire

Retourner St

Algorithme 5.4 : calcul des agrégats sous forme de résumé de texte.

5.6 Conclusion

Nous avons présenté dans ce chapitre quatre fonctions d'agrégation pour les documents textuels. Ces fonctions se basent sur une mesure textuelle pour synthétiser ce type de documents sous une nouvelle forme permettant ainsi d'extraire les informations pertinentes. Pour cela, nous avons adapté au contexte OLAP des techniques de fouille de texte.

L'agrégation des documents en utilisant la fonction *Classes* permet d'obtenir un ensemble d'agrégats, (classes), suite à l'application de la catégorisation de texte. La fonction *Clusters* offre des agrégats, (clusters), comme résultats de la tâche de *clustering*. La fonction *Top_Keyphrases* fournit un ensemble de multi-mots, (keyphrases), il s'agit des thèmes partagés par les documents à agréger. La fonction *Summary* quand à elle, offre un résumé, sous forme de phrases les plus pertinentes, des textes contenus dans les documents à agréger en utilisant la technique de résumé automatique de texte.

Ces fonctions permettent des analyses qualitatives et sémantiques sur le contenu textuel des documents. Des analyses peuvent même être faites sur des sections du document, comme nous l'avons vu pour les CVs. Ces quatres fonctions d'agrégation ont l'avantage de couvrir une panoplie de scénarios d'applications. En effet, la catégorisation, le *clustering*, l'extraction de keyphrases (descripteurs) et même le résumé automatique, sont des techniques utilisées dans de très nombreux domaines. Reste à voir comment intégrer cet ensemble de fonctions dans un même environnement OLAP pour les données textuelles.

Un autre aspect important à citer, est que ces fonctions peuvent être utilisées de façon imbriquée pour obtenir des agrégations plus fines. On peut imaginer après application de la fonction *Clusters*, l'application de la fonction *Top_Keyphrases* ou la fonction *Summary*. Il s'agit d'agréger le contenu des clusters en thèmes partagés dans les clusters, ou en un résumé texte par cluster, ce qui permet de mieux analyser les résultats obtenus.

Notons enfin, que des algorithmes simples ont été utilisés pour valider ces fonctions d'agrégation. Les résultats expérimentaux ainsi que les problèmes d'optimisation sont discutés dans le chapitre suivant.

CHAPITRE 6
EXPÉRIMENTATIONS ET VALIDATION

6.1 Introduction

Pour valider nos propositions, une étape d'évaluation est plus que nécessaire. En général, les mesures d'évaluation peuvent être classées en deux catégories [75] : les méthodes intrinsèques et les méthodes extrinsèques. Dans ces dernières, les sorties du système à évaluer sont jugées en se basant sur leur aptitude à accélérer la complétion d'autres tâches. Les méthodes intrinsèques quant à elles, réalisent un jugement direct des résultats obtenus. Ce jugement est réalisé, le plus souvent, de manière automatique en calculant des mesures de similarité vis-à-vis de références produites par des humains.

Ce chapitre présente les expérimentations menées pour deux objectifs : (1) valider les démarches de *summarization* proposées. Nous avons pris, à cet effet comme exemple, les fonctions *Classes* et *Clusters*. (2) valider la convenance de modéliser la structure du document textuel. Pour ces expérimentations, nous utilisons diverses mesures[22] (intrinsèques et extrinsèques) sur un corpus de documents CVs. Après une présentation du corpus en section 6.2, les sections 6.3 et 6.4 sont consacrées respectivement aux expérimentations menées sur les fonctions *Classes* et *Clusters*. Les résultats des algorithmes choisis sont discutés, les performances en temps de calcul sont aussi présentées. Nous discutons en section 6.5 les problèmes d'optimisation dans les cubes de textes et nous concluons ce chapitre dans la section 6.6.

6.2 Présentation du corpus

Pour faire nos expérimentations, nous avons constitué un corpus de taille moyenne de CVs pour la catégorie « Ingénieur en Informatique ». Nous avons utilisé pour cela le site Emploi Algérie[23] donnant un accès gratuit à leur CVthèque. 310 CVs au format html

[22] Rappelons que ces mesures sont présentées dans le chapitre 3, et cela pour chaque technique utilisée.
[23] http://www.emploiAlgerie.com

ont été téléchargés puis nettoyés en préparation du corpus. Nous avons utilisé apache Tika[24] pour extraire le texte des fichiers html.

La traduction de la catégorie « Ingénieur en Informatique » en une requête multidimensionnelle est la suivante : (Localité = "*", secteur d'activité = "Informatique", niveau d'étude = "Bac + 5 et plus", structure = "*"), où "*" dénote quelque soit. Nous simulons ainsi, une requête multidimensionnelle en procédant les expérimentations sur un corpus de documents textuels qui lui simule l'ensemble de documents retourné par un cube de textes, suite à la requête précédemment formulée.

6.3 Fonction *Classes*

Cette section vise à présenter les résultats de l'expérimentation menée pour valider la fonction *Classes*. Rappelons que cette fonction vise à présenter au décideur un résumé sous forme de *n* classes. Pour le cas des documents CVs, trois classes sont fournies au décideur (le recruteur). Pour cela, nous avons choisi une approche de classification par apprentissage supervisé. Il s'agit de faire de la *summarization* des documents CVs par catégorisation dans les cubes de textes [91].

Rappel de la démarche suivie : après une étape de prétraitements, les documents CVs ont été transformés en des vecteurs de poids des termes, dans un espace vectoriel, en utilisant le schéma de pondération *tf-idf*. Un apprentissage a été effectué en utilisant l'algorithme « basé centroide » dont le principe est de construire, pour chaque ensemble de documents de la même classe, un vecteur prototype, le centroide. Un document CV est assigné à la classe dont le vecteur centroide est le plus similaire au vecteur de ce document. Cette similarité est calculée en utilisant la mesure du cosinus.

Nous avons effectué nos expérimentations sur un sous-ensemble de 141 CVs du corpus des 310 CVs. Ceci est dû au fait que par rapport à une offre d'emploi, prendre la totalité des CVs va engendrer des classes très déséquilibrées car le nombre de candidatures non pertinentes est beaucoup plus grand que celles qui sont pertinentes.

Pour les 141 CVs, chaque candidature est identifiée comme pertinente, pouvant être pertinente ou non pertinente. Une évaluation pertinente est attribuée à une candidature potentiellement intéressante pour un emploi donné, une valeur «peut être» correspond à une

[24] http://www.tika.apache.org. Tika est un toolkit Java facilement intégrable, destiné à la détection, l'extraction et l'analyse de métadonnées et de données texte structurées à partir d'une très large variété de formats de fichiers. Parmi ces formats, on retrouve : HTML, XML, Microsoft Office, OpenOffice, PDF, images, ebooks, Rich Text, divers formats de compression et de packaging, audio/image/vidéo, email/mbox, etc.

bonne candidature mais pas la meilleure et une évaluation non pertinente est donnée à un candidat rejeté, selon l'avis de l'expert en recrutement. Le tableau 6.1 résume les informations sur le corpus utilisé et son étiquetage.

Nombre de CVs	Pertinence de la candidature		
	Oui	Peut être	Non
141	35	33	73

Tableau 6.1: Statistiques du corpus utilisé en fonction de l'étiquetage.

6.3.1 Protocole expérimental

Afin d'éviter de faire l'expérimentation sur un seul ensemble d'apprentissage et un seul ensemble de test[25], nous avons choisi la stratégie de la validation croisée ou « *k-fold cross validation* » [18, 80]. Dans cette approche, les données sont réparties aléatoirement (mais de façon équilibrée) en *k* ensembles. k exécutions sont réalisées, où pour chaque exécution, un des k ensembles est utilisé pour la validation et (k-1) ensembles restants sont utilisés pour l'apprentissage. Les valeurs de k les plus utilisées sont celles comprises entre 5 et 10.

Nous avons choisi *5-fold cross validation*; les 141 CVs ont donc été scindés en cinq sous-ensembles approximativement de même taille $F_i; i = 1, \dots, 5;$ avec une répartition aléatoire mais équilibrée des candidatures dans chaque sous-ensemble. Cinq exécutions ont été réalisées $E_i; i = 1, \dots, 5$ où pour chaque exécution E_i, quatre des cinq sous-ensembles sont concaténés pour constituer l'ensemble d'apprentissage et le cinquième est utilisé pour la validation. Le tableau 6.2, résume les différentes expérimentations menées.

	E1		E2		E3		E4		E5	
	Ensemble	taille	Ensemble	taille	Ensemble	taille	Ensemble	taille	Ensemble	taille
Jeux d'apprentissage	F1, F2, F3, F4	81	F1, F2, F3, F5	80	F1, F2, F4, F5	81	F1, F3, F4, F5	80	F2, F3, F4, F5	80
jeux de validation	F5	22	F4	23	F3	22	F2	23	F1	23

Tableau 6.2 : *validation croisée*, répartition des différents sous-ensembles sur les cinq expérimentations.

Les mesures utilisées ont été présentées dans le chapitre 2 (voir section 2.5.2.3) que nous rappelons ci-dessous. Nous avons utilisé la mesure $F - score$ (formule 6.1) des documents bien classés, moyennée sur toutes les classes.

[25] Ce qui pourrait conduire au phénomène de sur-apprentissage ou apprentissage par cœur.

$$F_1 - score = \frac{2 * \langle Précision \rangle * \langle Rappel \rangle}{\langle Précision \rangle + \langle Rappel \rangle} \quad (6.1)$$

La précision moyenne et le rappel moyen sont calculés par les formules (6.2) et (6.3) ci-dessous, avec n le nombre de classes :

$$\langle Précision \rangle = \frac{\sum_{i=1}^{n} Précision_i}{n} \quad (6.2)$$

$$\langle Rappel \rangle = \frac{\sum_{i=1}^{n} Rappel_i}{n} \quad (6.3)$$

Où, pour chaque classe i :

$$Précision_i = \frac{Nombres\ de\ documents\ correctement\ attribués\ à\ la\ classe\ i}{nombre\ de\ documents\ attribués\ à\ la\ classe\ i} \quad (6.4)$$

$$Rappel_i = \frac{Nombres\ de\ documents\ correctement\ attribués\ à\ la\ classe\ i}{nombre\ de\ documents\ appartenant\ à\ la\ classe\ i} \quad (6.5)$$

6.3.2 Résultats et discussion

Pour voir l'impact de la structure du CV sur les résultats de la catégorisation et identifier ainsi les parties du CV contenant les informations les plus pertinentes, nous avons mené trois expérimentations différentes. Dans la première expérimentation les parties *formations*, *expériences professionnelles* et *compétences* (*CV_FEC*) sont utilisées. Dans la deuxième expérimentation la partie *expériences professionnelles* (*CV_E*) est utilisée. Dans la troisième, les parties *expériences professionnelles* et *compétences* (*CV_EC*) sont utilisées. Chacune de ces expérimentations a été faite avec le protocole décrit dans la section précédente. Les résultats obtenus, par l'algorithme «basé centroide» sur l'ensemble des 141 CVs pour les trois expérimentations, sont résumés dans le tableau 6.3, ci-dessous :

	Précision			Rappel			F-score		
	CV_FEC	CV_E	CV_EC	CV_FEC	CV_E	CV_EC	CV_FEC	CV_E	CV_EC
Classe A	0,634	0,418	0,500	0,743	0,685	0,629	**0,684**	0,519	0,557
Classe B	0,392	0,297	0,422	0,267	0,276	0,395	0,318	0,286	**0,408**
Classe C	0,703	0,688	0,719	0,726	0,469	0,631	**0,714**	0,558	0,672
Toutes classes	0,576	0,468	0,547	0,579	0,477	0,552	**0,577**	0,472	0,549

Tableau 6.3 : Précision, Rappel et F-score obtenus pour les trois expérimentations.

Les résultats obtenus dépendent, d'une part de la qualité des données et de l'expertise humaine. D'autre part, d'un point de vue ressources humaines, une candidature appartenant à la classe B est une bonne candidature mais ne correspond pas forcément au profil recherché ; ou une bonne candidature mais pas la meilleure. Cette dernière affirmation rend difficile l'initialisation de cette classe. Néanmoins, la démarche adoptée dans cette section, montre qu'il est possible de récupérer les documents d'un cube de textes, correspondant à une requête multidimensionnelle pour appliquer la catégorisation de texte. Les classes obtenues représentent les valeurs de la mesure textuelle OLAP, le CV dans notre cas. Ces classes représentent aussi, pour le décideur, des agrégats en rapport avec la nature des données textuelles et la spécificité du domaine.

Par ailleurs, ces résultats montrent aussi que les meilleurs scores ont été obtenus en utilisant les parties *formations*, *expériences professionnelles* et *compétences* (l'expérimentation CV_FEC). Nous pouvons donc conclure que les informations pertinentes permettant de déterminer la candidature adaptée à une offre d'emploi sont contenues dans ces parties. La figure 6.1 donne les résultats graphiques du tableau 6.3.

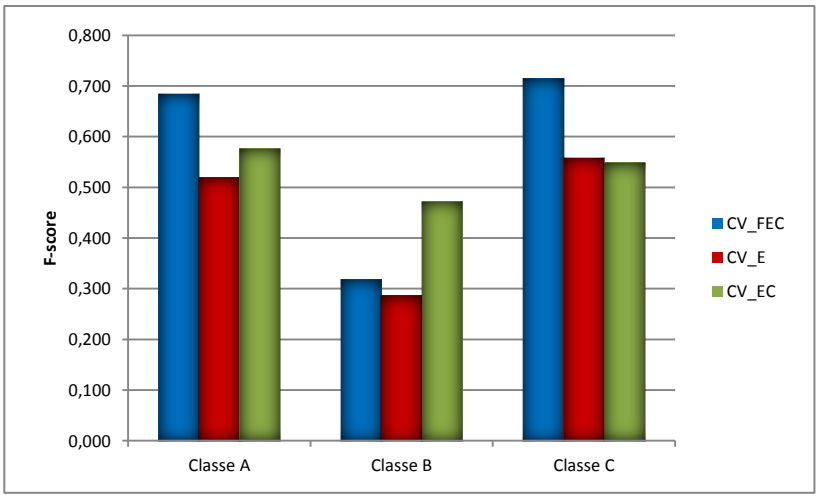


Figure 6.1 : F-score obtenus pour toutes les classes, en utilisant différentes parties du CV.

6.3.3 Performances en temps d'exécution

En plus de sa simplicité, l'algorithme « basé centroide » a une complexité linéaire. En effet, le temps nécessaire pour classer un nouveau document x est au plus $O(km)$, où m est le nombre de termes présents dans x et k le nombre de classes. Les expérimentations menées par Shankar et Karypis [73], montrent que l'agorithme «basé centroide» est de 10 à 20 plus rapide que le classifieur SVM. Cette efficacité computationnelle est très

importante pour les systèmes OLAP qui nécessitent des temps de réponse très courts. Le tableau 6.4 donne les temps de calcul pour les trois expérimentations menées.

Expérimentations	Taille de l'espace des caractéristiques	Temps de calcul (en secondes)
CV_FEC	2747	6,79
CV_E	1761	3,26
CV_EC	2461	5,50

Tableau 6.4 : Temps de calcul obtenus pour les trois expérimentations.

Ces résultats ont été obtenus sur un PC portable, Intel Core™2 Duo 2.0 GHz et 3,00 Go de RAM sous Windows seven[26]. Ils incluent :

- Le temps de filtrage, celui-ci représente 14,73%.
- Le temps de vectorisation, il représente 84,53%.
- Le temps de catégorisation, qui représente 0,74%.

Nous pouvons remarquer que le temps de catégorisation est vraiment négligeable (de l'ordre des millisecondes); par contre le temps de vectorisation est le plus élevé dû au fait que l'estimation du poids du terme nécessite une double passe sur l'ensemble des documents de la collection.

6.4 Fonction *Clusters*

Dans cette section nous présentons les résultats de l'évaluation menée pour valider la démarche de *summarization* en utilisant la fonction *Clusters*. Rappelons que le but de cette démarche est de présenter au décideur un résumé sous forme de *k* clusters. Pour cela, nous avons choisi une approche de classification par apprentissage non supervisé. Il s'agit d'une variante de la méthode des K-means, le K-means sphérique, où le paramètre *k* est déterminé par l'utilisateur. L'inconvénient majeur de ce type de méthode est la détermination du paramètre *k* quand aucune connaissance de la structure des données n'est connue a priori, ce qui est le cas pour les CVs. Nous proposons dans cette section, d'aider le décideur à déterminer le paramètre *k* en faisant varier ce paramètre ce qui permet d'obtenir plusieurs schémas de partitionnement. Nous évaluons ensuite ces schémas en utilisant plusieurs indices de qualité. Ces derniers, présentés dans le chapitre 2, permettent d'évaluer la compacité ainsi que la séparabilité des clusters. Avant de voir les résultats obtenus en section 6.4.3, nous commençons dans la section 6.4.1 par rappeler la démarche

[26] Pour l'implémentation nous avons utilisé le langage de programmation java sous l'environnement de développement Eclipse Indigo, http://www.eclipse.org/indigo/

de *summarization* par *clustering*. En section 6.4.2 nous présentons la démarche expérimentale adoptée. Les performances en temps de calcul sont discutées en section 6.4.4.

6.4.1 Rappel de la démarche de *summarization*

La démarche proposée consiste en une étape de prétraitements, les documents CVs ont été transformés en des vecteurs de poids des termes, dans un espace vectoriel, en utilisant le schéma de pondération *tf-idf*. Nous avons ensuite appliqué l'algorithme K-means dont le principe est de chercher un partitionnement des données en se basant sur les centroides des clusters. Dans une première étape, ces centroides sont tout d'abord initialisés aléatoirement en prenant des vecteurs documents aléatoirement dans l'ensemble de leurs représentations vectorielles, X. Dans une deuxième étape, l'algorithme consiste en la répétition de deux étapes : (1) affecter un document x au cluster dont le centre μ est le plus proche au vecteur document, selon une distance donnée, dans notre cas la mesure cosinus. (2) Redéfinir les centres des clusters. Ces deux étapes sont répétées jusqu'à ce que la fonction objective soit localement optimisée, ou jusqu'à un nombre τ d'itérations.

6.4.2 Démarche expérimentale

Nous avons effectué nos expérimentations sur l'ensemble des 310 CVs. Comme aucune connaissance sur le partitionnement des données du corpus n'est connue, nous avons choisi de valider le *clustering* en utilisant des critères internes. Notre démarche expérimentale peut être vue en deux phases, tout d'abord (1) nous aidons le décideur à définir le nombre optimal de partitions, ce qui permet d'obtenir un partitionnement de qualité. Dans un deuxième temps, (2) nous refaisons les expérimentations, en ne considérant que la section « Formation » des documents. Comme pour la *summarization* par catégorisation, nous cherchons à valider la dimension structure. Ces deux étapes sont explicitées dans les sections qui suivent.

6.4.2.1 Trouver K optimal (qualité des agrégats)

Comme la qualité d'une partition est étroitement liée au choix du nombre de classes, définir le nombre de clusters *k* est un des problèmes les plus difficiles en *clustering*. En effet, il est souvent nécessaire de fournir le nombre de clusters souhaité comme paramètre de l'algorithme. Pour aider le décideur à trouver le nombre de clusters optimal, nous proposons comme démarche de procéder à plusieurs exécutions du même algorithme,

en faisant varier à chaque fois le nombre de clusters, *k*. Les résultats obtenus sont ensuite comparés en se basant sur un ou plusieurs indices de qualité. La valeur de *k*, qui a permis d'obtenir le meilleur schéma de partitionnement, est alors retenue.

Zhao et Karypis [92, 93] proposent 7 critères pour évaluer la qualité d'un schéma de partitionnement, pour le *clustering* de documents textuels. Les performances de ces fonctions ont été évaluées en utilisant des connaissances externes. Nous proposons d'utiliser pour nos expérimentations 3 de ces critères qui ont donné les meilleures performances et qui sont donnés dans les formules suivantes :

$$\varphi_1(\mathrm{C}) \qquad Maximiser \sum_{j=1}^{k} \sum_{x_i \in C_j} \cos(x_i, \mu_j) \tag{6.6}$$

$$\varphi_2(\mathrm{C}) \qquad Minimiser \sum_{j=1}^{k} n_j \cos(C_j, C) \tag{6.7}$$

$$\varphi_3(\mathrm{C}) \qquad Maximiser \frac{\varphi_1(\mathrm{C})}{\varphi_2(\mathrm{C})} \tag{6.8}$$

Ces indices peuvent être utilisés pour guider le processus de *clustering* en optimisant la fonction correspondante, ou pour évaluer le résultat obtenu. Comme notre objectif est de comparer plusieurs schémas de partitionnement, nous avons choisi de calculer ces indices à l'issue de l'exécution de l'algorithme.

La fonction φ_1 cherche à maximiser la similarité entre chaque document et le vecteur centroide du cluster auquel il est assigné. Cette fonction prend en considération le critère de compacité des clusters. La fonction φ_2 par contre cherche à minimiser le cosinus entre le vecteur centroide de chaque cluster C_i et le vecteur centroide de la collection C. Cette fonction prend en considération le critère de séparabilité des clusters. L'indice φ_3 prend en compte les deux critères de compacité et de séparabilité des clusters.

6.4.2.2 Résultats et discussion

Dans cette première expérimentation, nous avons utilisé toutes les parties du CV. Les résultats du tableau 6.5 montrent que la meilleure partition est obtenue pour la valeur de k=9. En effet, l'indice φ_1 diminue, φ_2 augmente et par conséquent φ_3 diminue. Les résultats graphiques sont donnés par la figure 6.2.

	$\varphi_1(C)$	$\varphi_2(C)$	$\varphi_3(C)$
K=2	55.315	279.805	0.198
K=3	60.197	249.602	0.241
K=4	62.3	241.206	0.258
K=5	66.608	232.532	0.286
K=6	69.848	213.626	0.327
K=7	73.036	206.599	0.354
K=8	74.202	201.267	0.369
K=9	**79.116**	**187.407**	**0.422**
K=10	78.610	195.382	0.402

Tableau 6.5 : valeurs des indices de qualité pour le *clustering* de CVs.

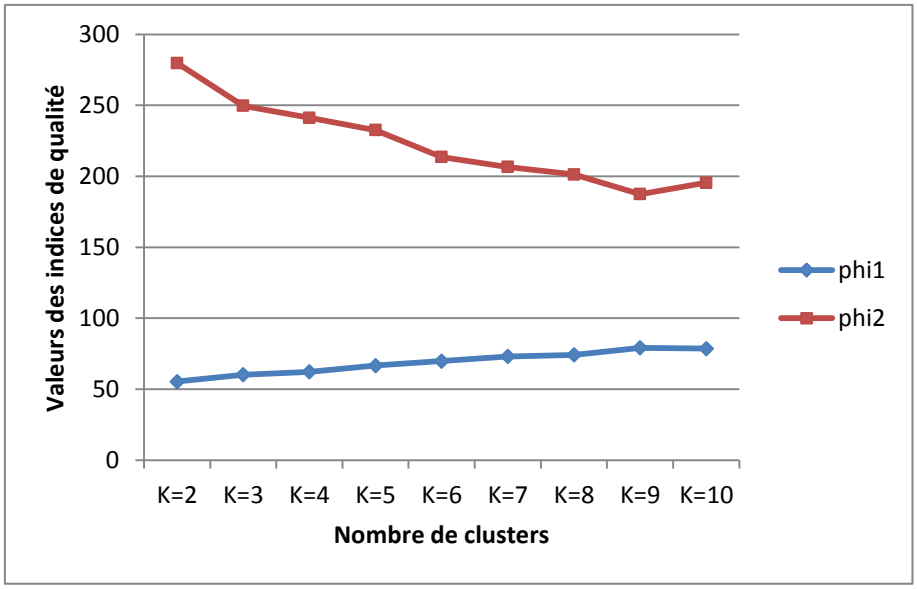


Figure 6.2 : qualité du *clustering* en considérant toutes les parties du CV et en faisant varier le paramètre *k*.

6.4.2.3 Influence de la structure sur les résultats de *clustering*

Par ailleurs et pour étudier l'impact de la structure sur les résultats obtenus, nous avons refait les expérimentations en ne considérant que la section « Formation » des documents. Les résultats sont donnés dans le tableau 6.6; ils montrent un partitionnement des données différent quand seule la section « Formation » est prise en considération. Une valeur de k=14 montre un schéma de partitionnement de bonne qualité. La figure 6.3 illustre les résultats graphiques.

	$\varphi_1(C)$	$\varphi_2(C)$	$\varphi_3(C)$
K=2	57.534	264.211	0.218
K=3	63.249	246.164	0.257
K=4	67.157	223.717	0.300
K=5	72.421	209.760	0.345
K=6	75.465	201.776	0.374
K=7	79.876	196.270	0.407
K=8	83.347	189.052	0.441
K=9	86.635	179.434	0.483
K=10	91.625	171.999	0.533
K=11	92.068	172.758	0.533
K=12	92.521	166.800	0.555
K=13	96.03	161.883	0.593
K=14	**103.594**	**150.470**	**0.688**
K=15	102.165	154.916	0.659
K=16	99.599	157.934	0.631

Tableau 6.6 : Qualité de *clustering* en ne considérant que la partie « Formation » du CVs.

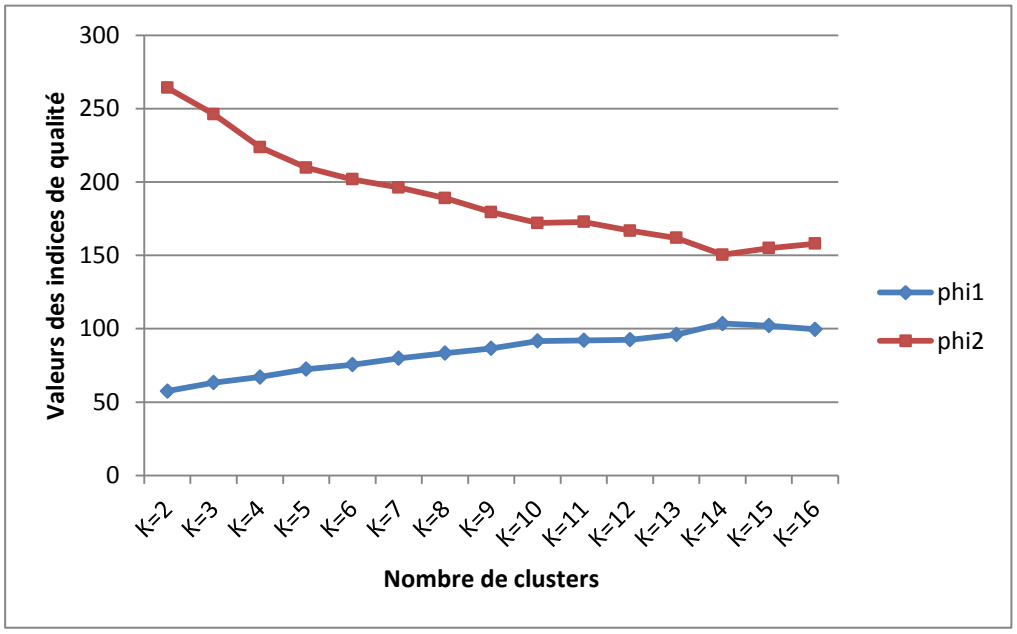


Figure 6.3 : qualité du *clustering* en ne considérant que la partie « Formation » du CV.

Pour mieux visualiser l'impact de la structure sur la qualité du *clustering*, nous avons superposé les courbes φ_3 pour le cas où toutes les parties du CV (courbe CV_FECA) sont pris en considération et le cas où seule la section « Formation » du CV est considérée (courbe CV_F). Rappelons que l'indice φ_3 prend en compte les deux aspects de compacité et de séparabilité des clusters, contrairement φ_1 et φ_2 qui ne prennent en compte qu'un seul des deux aspects. La figure 6.4 illustre cette comparaison.

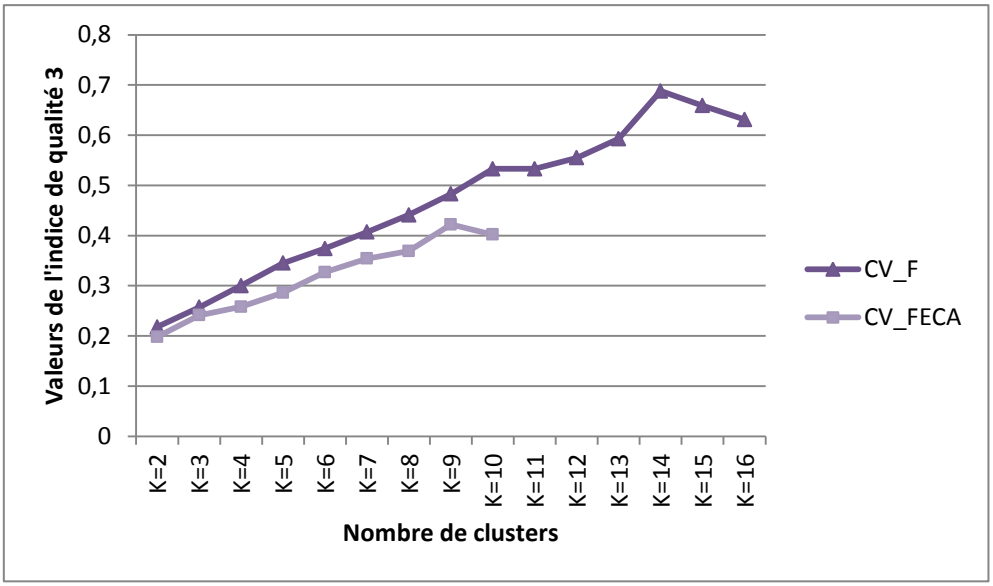


Figure 6.4 : Comparaison des indices φ_3 en ne considérant que la partie « Formation » du CV.

Il est clair à travers cette figure que la qualité du partitionnement obtenue en ne considérant que les sections « Formation » des CVs, est nettement meilleure qu'en prenant le CV tout entier. Ceci peut être expliqué par le fait que le corpus de CVs (les corpus de textes de façon générale) a des caractéristiques denses, ce qui génère du bruit qui affecte négativement la performance des algorithmes de *clustering*. En effet, les CVs, sont écrits de façon libre, malgré la structure conventionnelle. Les candidats citent toutes sortes de compétences (et expériences) même celles qui ne sont pas directement liées au domaine de spécialité recherché. Ceci conduit à un espace de caractéristiques trop bruité.

6.4.3 Performances en temps d'exécution

La grande partie du temps pris par l'algorithme K-means est consacrée au calcul de distance. Le calcul de la distance entre deux documents est O(*M*) où *M* est la dimensionnalité de l'espace des caractéristiques. L'étape de ré-assignement des clusters calcule *KN* distances et donc sa complexité est O(*KNM*). L'étape de calcul des centroides est O(*NM*). Pour un nombre d'itérations *I*, la complexité totale est O(*IKNM*). Ainsi, l'algorithme K-means a une complexité linéaire. Les temps de calcul, pour les deux expérimentations, sont donnés dans le tableau 6.7.

Expérimentations	Taille de l'espace des caractéristiques	Temps de calcul (en secondes)
CV_FECA	4600	18,370
CV_F	1597	5,460

Tableau 6.7 : Temps de *clustering* obtenus pour les deux expérimentations.

Ces résultats ont été obtenus sur un PC portable, Intel Core™2 Duo 2.0 GHz et 3,00 Go de RAM sous Windows seven. Comme pour la catégorisation, les temps de vectorisation sont les plus élevés et représentent plus de 86% du temps total. Rappelons que la raison est due au fait que l'estimation du poids du terme nécessite une double passe sur l'ensemble des documents de la collection.

6.5 Problèmes de performances dans les cubes de textes

Dans cette section nous abordons les problèmes d'optimisation possibles, liés à l'agrégation des données textuelles. De plus, et comme nos expérimentations ont été faites sur des corpus de documents textes, nous discutons les possibilités de matérialisation du cube de textes.

6.5.1 Optimisation des Fonctions proposées

Basées sur la difficulté d'agrégation et les possibilités d'optimisation, les fonctions d'agrégation peuvent être classées en trois catégories [27] : distributives telle que «*count*», algébriques telle que «*average*» et holistiques telles que «*median*» ou «*K smallest value*». Ces dernières sont les plus difficiles à optimiser [27, 66, 86] car le processus d'agrégation ne dispose pas d'un résultat intermédiaire sur lequel il se base pour obtenir le résultat final.

Les fonctions que nous proposons dans ce mémoire sont des fonctions holistiques. La raison commune à toutes ces fonctions qui fait qu'elles soient holistiques, est la fonction de pondération *tf-idf*. En effet, la fonction de pondération globale *idf* dépend de la collection toute entière. Adapté au contexte OLAP, le nombre de documents dans la formule *idf* n'est plus par rapport à la collection de documents mais par rapport au nombre de documents à agréger. Ainsi, si nous reprenons la formule de la fonction de pondération globale, *idf* (voir formule 6.9 ci-dessous), N est le nombre de documents (ou granules de documents) à agréger par la fonction; et n_i le nombre de documents (ou granules de documents) à agréger contenant le terme t_i.

$$idf = log\frac{N}{n_i} \quad (6.9)$$

Cette fonction est donc étroite ment dépendante de la requête décideur, du nombre de documents à agréger. De plus, nous avons vu à travers les expérimentations qu'une partie considérable (plus de 80% pour les deux fonctions expérimentées) du temps de calcul est consacré au calcul de *tf-idf* à cause du fait que ce traitement nécessite une double passe sur l'ensemble des documents à agréger. Pour optimiser la fonction *tf-idf*, nous proposons deux solutions :

- Dans la première solution, nous proposons de pré-calculer la fonction de pondération locale *tf* à utiliser comme résultat intermédiaire et optimiser ainsi les temps d'agrégation.
- Comme nous sommes dans l'ère des données massives, nous pensons que cette solution peut s'avérer insuffisante et donc nous proposons comme deuxième solution de distribuer les traitements. En effet, en utilisant l'implémentation Hadoop[27] pour les traitements distribués, les expérimentations dans [94] montrent une amélioration significative des temps de calcul de *tf-idf*. La figure 6.5, montre les résultats obtenus.

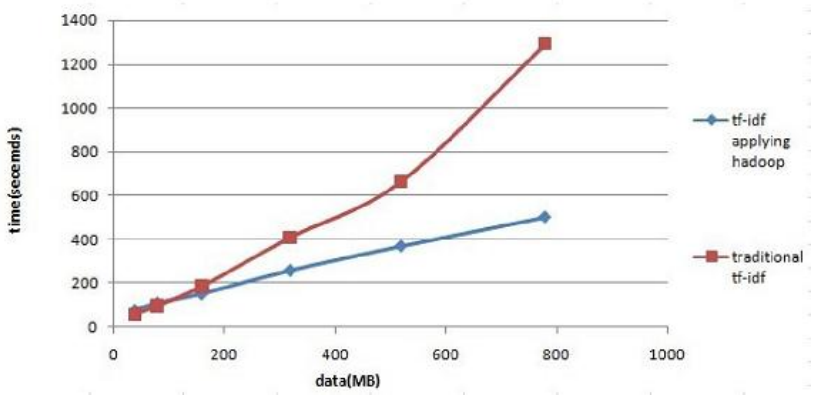


Figure 6.5 : Comparaison des résultats du calcul de tf-idf avec et sans utilisation de Hadoop [94].

6.5.2 Matérialisation du cube de texte

Il faut noter en premier lieu que la matérialisation d'un cube de textes est une tâche complexe. Les raisons ont été citées tout au long de ce mémoire. Nous sommes en présence de données dont le volume augmente sans cesse. La spécificité des données tex-

[27] Hadoop est un framework open source pour le traitement distribué des données. Il permet la programmation et l'exécution d'applications distribuées pour le traitement des données massives (big data). Hadoop est basé sur le modèle de programmation MapReduce.

tuelles par rapport au domaine et la langue est une autre difficulté. D'autres raisons nous ont conduit à la conclusion que même si le processus d'entreposage de documents est le même que pour celui des données (ce qui a déjà été cité en section 1.5, chapitre 1), cependant les mécanismes à utiliser doivent être repensés. Par exemple, comme il a été bien signalé dans [82] la prise en charge des documents dans les entrepôts de données nécessite d'étendre la notion de requête multidimensionnelle et d'indexation et même l'agrégation.

En prenant en compte toute cette complexité relative aux données textuelles, nous avons alimenté une base de données NoSQL orientée documents, en l'occurrence MongoDB[28]. La taille de cette base de données est de 1000 CVs dont les 310 CVs utilisés pour les expérimentations. MongoDB utilise le format de données BJON (binary-JSON[29]) pour la représentation et le stockage des données au sein des documents. La figure 6.6 illustre la description d'un document CV et ces métadonnées au format JSON.


```
{
	"_id" : ObjectId("500c19ed5025d1c573c3e1b7"),
	"cv_name" : "cv1.txt",
	"cv_path" : "D:\\CVs\\clustering\\cv1.txt",
	"Niveau_etude" : "Bac+5",
	"Secteur" : "Informatique",
	"Lieu" : {
		"Ville" : "Adrar",
		"Region" : "Sud"
	},
	"texte_cv" : {
		"Formation" : "FORMATION\t\t\t\t \t 2003 baccaluriat en science natural et de vie 2008 diplome d'ingénieur d'état en informatique spécialité système d'information avancée université mustapha stanbuli MASACARA2009 stage de formation spécialité transmission au niveau de l'école supérieur de transmission KOLEA 1RM\t \t\t",
		"Exp_prof" : "EXPERIENCES PROFESSIONNELLES\t\t\t\t \t\t\t\t\t\t\t2010-2011 chef section informatique service soutien technique au niveau de la direction régionelle de transmission CONSTANTINE 5RM\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t ",
		"Compétences" : "COMPETENCES\t\t\t\t \t les langages : C C++ C++ builder, java, matlab,.....installation des réseaux, traitement d'image, internet ....\t \t\t ",
		"Informations_comp" : "INFORMATIONS COMPLEMENTAIRES\t\t\t\t \t"
	}
}
```


Figure 6.6 : un document CV et ces métadonnées au format JSON.

Nous avons choisi d'utiliser MapReduce comme paradigme de programmation pour l'agrégation. Nous avons opté pour cela, pour le connecteur Hadoop pour MongoDB. Ce choix a été fait après avoir vu que le « *aggregation framework* » offert par MongoDB est inadapté à notre projet. MongoDB et Hadoop sont une combinaison puissante et peuvent

[28] L'annexe A est consacrée à la présentation du SGBD NoSQL de type document, MongoDB proposé par société 10gen.
[29] JSON est standard de description et d'échange de données comme XML,

être utilisés ensemble pour fournir des analyses complexes sur les données stockées dans MongoDB. Les travaux sont à leur début vu la complexité de la tâche, les résultats seront discutés dans nos futurs travaux.

6.6 Conclusion

Nous avons présenté dans ce chapitre les résultats des expérimentations menées sur un corpus de CVs constitué pour le secteur d'activité « Informatique » et le niveau d'étude « Ingénieur ». Ce dernier simule une collection de documents retournée par un cube de textes, correspondant à une requête décideur. Les expérimentations réalisées pour les deux fonctions *Classes* et *Clusters* valident les démarches proposées pour la *summarization* des documents textuels. De plus, ces expérimentations ont montré la pertinence de prendre en considération la structure conventionnelle du CV (la structure de façon générale d'un document textuel). En effet, les résultats sont différents selon les parties sur lesquelles porte la *summarization*. Reste à voir comment matérialiser de façon effective le cube de textes en vue d'optimiser les réponses aux requêtes décideur, ce que nous laissons comme travaux futurs.

CONCLUSION

Bilan et contributions

Le travail présenté dans ce mémoire s'inscrit dans le cadre des systèmes décisionnels. Ces derniers reposent sur deux technologies émergentes, les entrepôts de données et l'analyse en ligne OLAP. Le but de cette dernière, est de permettre la navigation interactive à travers les données et l'exploration de la structure multidimensionnelle de celles-ci, pour en extraire de la connaissance. Cependant, et d'après Inmon [36], 99% des décisions sont prises sur la base des données structurées, alors que le volume des données non structurées est de quatre à cinq fois plus grand que le volume de données structurées. Il est admis que les textes constituent l'essentiel des données non structurées, soit 80%. La valeur ajoutée des données textuelles est inestimable pour le processus d'aide à la décision. Cependant, ces données sont hors de portée des systèmes OLAP, par l'absence de moyens adaptés à leur intégration, leur traitement et leur analyse.

Dans la vaste problématique liée à la prise en charge des données textuelles par les systèmes décisionnels, nous nous sommes intéressés à l'analyse en ligne des documents textuels, en particulier, leur agrégation. Avec les outils OLAP classiques, il est impossible d'agréger des données textuelles selon des fonctions arithmétiques. L'environnement OLAP est inadapté pour ce genre de données, il doit être doté de nouvelles techniques d'agrégation pour prendre en charge la spécificité de ces données. C'est ainsi que nous avons proposé quatre fonctions d'agrégation, en adaptant au contexte OLAP des techniques du domaine de la fouille de texte. OLAP et fouille de textes ont en commun de vouloir synthétiser de gros volumes d'informations pour en extraire les informations les plus pertinentes, permettant à un décideur de prendre des décisions efféctives.

Les fonctions que nous avons proposées ont pour objectifs, la *summarization* des documents dans les cubes de données où la mesure utilisée, est une mesure textuelle modélisant le contenu (ou une partie du contenu) du document à analyser. Nous utilisons le terme de *summarization* par lequel nous entendons, un processus qui permet de résumer leurs contenus sous une nouvelle forme permettant de les analyser afin d'en extraire de la connaissance. Nous voulons aller au-delà des analyses quantitatives sur les documents

textuels, et permettre des analyses qualitatives et sémantiques sur le contenu textuel des documents.

Pour permettre cela, nous avons tout d'abord proposé un modèle multidimentionnel prenant en compte le contenu et la structure des documents textuels. Cette modélisation a ciblé un problème d'analyse précis, celui de l'analyse de CVs. Elle permet aussi d'instancier un cube de textes pour lequel des fonctions d'agrégation adaptées sont proposées. Ces fonctions sont *Classes*, *Clusters*, *Top_Keyphrases* et *Summary*.

L'agrégation des documents en utilisant la fonction *Classes* permet d'obtenir un ensemble d'agrégats, (classes) suite à l'application de la catégorisation de texte. La fonction *Clusters* offre des agrégats (clusters) comme résultats de la tâche de *clustering*. La fonction *Top_Keyphrases* fournit un ensemble de groupes de mots (keyphrases), il s'agit des thèmes partagés par les documents à agréger. La fonction *Summary* quand à elle, offre un résumé sous forme de phrases les plus pertinentes des textes contenus dans les documents à agréger, en utilisant la technique de résumé automatique de texte.

La validation de notre démarche de *summarization* a été menée à travers des expérimentations, sur un corpus de documents textuels, en l'occurrence des CVs. Les résultats sont prometteurs, néanmoins ce travail n'est qu'un début et de nombreuses perspectives sont envisageables.

Perspectives

Les problèmes liés à l'analyse en ligne des données textuelles, notamment l'agrégation de ces données, ne sont que partiellement résolus. Diverses perspectives de recherche sont ouvertes.

Couplage entre OLAP et fouille de texte. Nous pensons que l'OLAP et la fouille de texte sont deux domaines qui se complémentent et leur couplage serait une solution envisageable pour une analyse sémantiquement riche, des données textuelles. Il est donc nécessaire d'envisager ce couplage en un même processus d'aide à la décision, ce qui permet d'étendre les capacités des systèmes OLAP pour prendre en charge les données textes.

Redéfinition de la notion de forage. La notion de forage est différente quand il s'agit des données textuelles. Pour obtenir différents niveaux d'agrégation, les fonctions proposées peuvent être utilisées de façon combinée. Par exemple, l'application de la fonction *Top_Keyphrases* ou la fonction *Summary* après application de la fonction *Clusters*, permet d'agréger le contenu des clusters en thèmes partagés dans le cluster, ou en un résumé texte

par cluster, ce qui permet de passer vers un niveau de *summarization* plus fin. La notion de forage doit être redéfinie pour ce type de données, en créant des fonctions par combinaison des fonctions proposées.

Environnement Text OLAP collaboratif. Les fonctions que nous proposons ont l'avantage de couvrir une panoplie de scénarios d'applications. En effet, la catégorisation, le *clustering*, l'extraction de keyphrases et même le résumé automatique, sont des techniques utilisées dans de très nombreux domaines. Cependant, intégrer cet ensemble de fonctions dans un même environnement OLAP pour les données textuelles n'est pas trivial. La spécificité des documents à résumer implique l'utilisation de techniques différentes de fouille de texte. Le processus de *summarization* est dépendant à la fois du type de texte à agréger ainsi que de l'utilisation qui en sera faite. Pour une même technique, par exemple la catégorisation ou le *clustering*, différents algorithmes sont envisageables et le choix n'est pas évident car il n'y a pas de méthode ayant donné la preuve de sa supériorité. C'est ainsi, que sont apparues les solutions collaboratives. Ces dernières consistent à faire collaborer plusieurs classifieurs ou plusieurs algorithmes de *clustering* [80]. Ces solutions ont donné de meilleurs résultats que les algorithmes pris séparément. Ce qui pourrait être une solution pour un environnement générique, ou par domaine, supportant plusieurs applications.

Exploiter la structure des documents textuels. Nos expérimentations ont montré que les résultats sont meilleurs quand on applique une technique sur des sections de CVs, que sur le document complet. Exploiter la structure des documents textuels est un plus pour l'analyse de ceux-ci. Cependant, ce ne sont pas tous les documents qui possèdent une structure visible. Les données textuelles ont la spécificité d'être peu ou pas structurées. Une solution possible serait de se pencher vers des approches de représentation des structures du texte telle que la RST[30] : *Rhetorical Structure Theory* ou théorie de la structure rhétorique du texte, du domaine du traitement automatique de la langue naturelle (TALN).

[30] http://www.sfu.ca/rst/07french/introduction.html

APPENDICE A
Le SGBD NoSQL, MongoDB

A.1 Introduction

Nous assistons ces dernières années, à l'émergence de nouvelles applications sur le web (telles que Facebook, Twitter, LinkedIn, etc) connaissant un grand succès et par conséquent des charges énormes. Ces applications font face à de gros défis, notamment le volume de données (généralement complexes) qu'elles doivent gérer ainsi que la croissance exponentielle de ce volume. Les bases de données de ces systèmes doivent donc être capables de gérer ces données massives mais aussi de faire face à la montée en charge sans devoir interrompre leur service. Ces nouvelles applications n'ont pas trouvé dans les systèmes de bases de données traditionnelles de solutions à leur problème de gestion de données massives. C'est ainsi, qu'est apparu le mouvement NoSQL (not only SQL) qui regroupe de nombreuses bases de données, récentes pour la plupart, qui se caractérisent par une logique de représentation des données non relationnelle avec un schéma de données non fixe et qui n'offrent donc pas une interface de requêtes en SQL. NoSQL, représente un paradigme appliquant de nouvelles méthodes pour surpasser les limites des SGBDR, particulièrement quand il s'agit du passage à l'échelle.

Les bases de données NoSQL utilisent le modèle de données clé-valeur qui peut être vu comme une large table de hachage persistante. Il est, par conséquent, adapté aux caches et offre donc de hautes performances en termes d'accès aux informations. Cette modélisation très simple est justifiée par le constat qu'un bon nombre d'accès aux bases de données se résume à de simples lectures à partir d'un index. Plusieurs extensions du modèle clé-valeur ont été adoptées par le mouvement NoSQL, nous distinguons les bases de données orientées document, les bases de données orientées colonnes et les bases de données orientées graphe. Nous présentons dans cette annexe MongoDB, une base de données orientée document et qui a été choisie comme solution de persistance dans le cadre de la problématique traitée dans ce mémoire.

A.2 Présentation de MongoDB

MongoDB (dérivé du mot anglais *hu**mongo**us*, et qui veut dire gigantesque) est un SGBD NoSQL orienté document, open source, proposé par la société 10gen[31]. Dans ce qui suit nous présentons les caractéristiques clés de MongoDB.

A.2.1 Modèle de données

Le modèle de données dans MongoDB est orienté document. Un *document*, l'unité de base de MongoDB, est un ensemble ordonné de paires clé-valeur. Il est identifié par son nom. Dans MongoDB les documents sont stockés dans une *collection*. Un ensemble de *collections* forme une *base de données*. Une collection peut être assimilée à une table relationnelle et un document à une ligne dans celle-ci. Chaque document dispose d'une clé unique permettant de l'identifier dans la collection.

A.2.2 Le format de données

La notion de document dans le mouvement NoSQL fait référence à une valeur (telle que dans le modèle clé-valeur) composée de champs. Ces derniers peuvent être imbriqués les uns dans les autres, au format JSON (*JavaScript Object Notation*). Comme XML, JSON[32] est un format de données textuel, ouvert pour la description et la représentation de structures de données. Dans l'exemple ci-dessous, un document stockant un blog au format JSON [101] :

```
{       author: {
                first_name: "Barry",
                last_name: "Devlin"  },
        title: "Business Intelligence - NoSQL… No Problem",
        text: "Business intelligence (BI) originated in the mid-1980s... ",
        keywords: [ "nosql", "database", "data warehousing" ] }
```

MongoDB utilise une variante binaire plus compacte de JSON nommée BSON (*binary-encoded JSON*) pour son stockage interne. À la différence avec les bases de données relationnelles, le nombre de champs d'un document ainsi que sa structure ne sont pas définis au préalable. On dit que MongoDB est « *schema-less* », ce qui signifie qu'une base de données MongoDB, n'est pas limitée par des colonnes ou types de données prédéfinis comme dans les bases de données relationnelles.

---

[31] http://www.mongodb.org/

[32] Des questions se posent au sein de la communauté scientifique sur l'éventualité d'adopter le standard JSON en remplacement du format XML. Bon nombre de pages web existent présentant une comparaison des avantages de ces deux standards, dont voici quelques-unes :
http://www.readwriteweb.com/hack/2010/11/json-vs-xml.php
http://think2loud.com/680-json-xml/
http://www.json.org/xml.html

L'avantage de cette implémentation est que la gestion des données est extrêmement flexible car aucune structure prédéfinie pour le document n'est requise. Ce qui permet d'avoir des documents avec des structures différentes (hétérogénéité structurelle) dans une même collection. L'exemple ci-dessous montre comment que deux types de documents différents peuvent coexister dans une même collection appelée Media [100] :

```
{
    "Type": "CD",
    "Artist": "Nirvana",
    "Title": "Nevermind",
    "Genre": "Grunge",
    "Releasedate": "1991.09.24",
    "Tracklist": [
        {
        "Track" : "1",
        "Title" : "Smells like teen spirit",
        "Length" : "5:02"
        },
        {
        "Track" : "2",
        "Title" : "In Bloom",
        "Length" : "4:15"
        }
    ]
}
{
    "type": "Book",
    "Title": "Definite Guide to MongoDB: The NoSQL
    Database for Cloud and Desktop Computing, the",
    "ISBN": "987-1-4302-3051-9",
    "Publisher": "Apress",
    "Author": [
        "Plugge, Eelco",
        "Membrey, Peter",
        "Hawkins, Tim"
    ]
}
```

Par ailleurs, le fait qu'aucun espace, n'est alloué aux champs qui n'existe pas dans certains documents, ceci signifie que pour les ensembles de données très épars une économie significative de l'espace disque est réalisée par rapport aux bases de données relationnelles où l'espace doit être réservé à chaque champ même s'il est nul.

A.2.3 Architecture de MongoDB

MongoDB a été conçu pour fonctionner selon plusieurs modes différents :

- Serveur seul.
- Réplication maître/esclave.
- Réplication sur un ensemble de serveurs traitants les mêmes données : via Replica set.
- Partitionnement des données sur plusieurs serveurs : via sharding.

A.2.3.1 Serveur seul

Dans ce mode mettant en jeu un seul serveur, un seul processus nommé mongod est utilisé et traite directement les données issues des requêtes du client. Comme sur la figure ci-dessous :

Figure A.1 : mode de fonctionnement serveur seul

A.2.3.2 Réplication maître/esclave

Ce mode est le plus répandu, il peut être utilisé comme outil de sauvegarde, de basculement et peut servir à la montée en charge horizontale (pour les opérations de lectures). Dans ce mode, deux serveurs mongod sont lancés nommés « maître » et « esclave ». Les données sont, dans un premier temps, écrites sur le serveur maître, l'information est ensuite transmise de manière asynchrone aux serveurs esclaves. Si un serveur esclave subit une défaillance, il suffit de relancer l'instance et il synchronisera ses données avec le serveur maître via un fichier log. Si un serveur maître subit une défaillance, le système passe en mode lecture unique jusqu'à ce que le serveur maître soit relancé. Ce mode de réplication offre un service de basculement en cas de défaillance notamment pour les applications à lectures fréquentes.

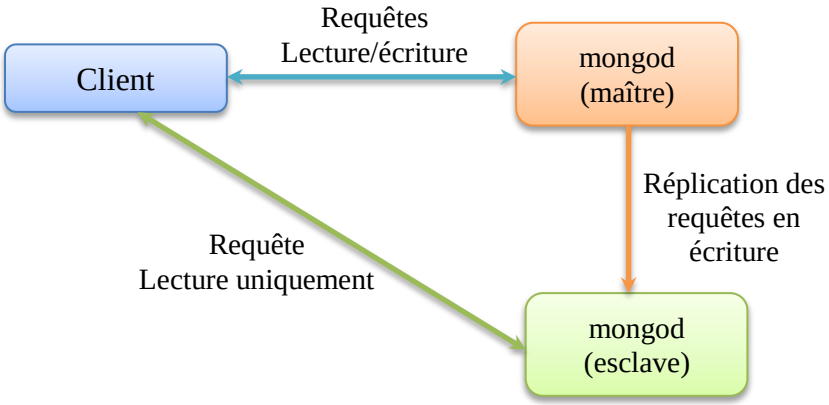


Figure A.2 : réplication maître/esclave

A.2.3.3 Réplication par Replica Set

Il s'agit d'une version améliorée du mode maître/esclave. La réplication par replica Set offre un balancement automatique. Dans ce mode, chaque serveur est identique et a un rôle *primaire* ou *secondaire*. Un seul serveur primaire existe par replica set. En cas de défaillance de ce dernier, les membres du replica set procèdent à l'élection d'un nouveau serveur primaire. Il s'agit de celui qui détient les informations les plus récentes.

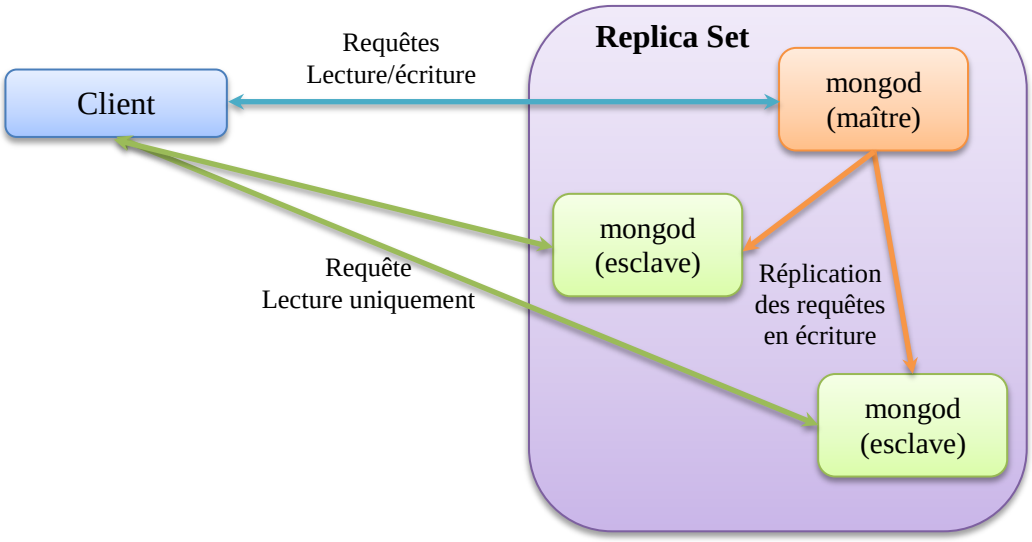


Figure A.3 : fonctionnement via replica set

A.2.3.4 Partitionnement des données via Sharding

Les mécanismes de réplication permettent de faire évoluer les performances en lecture sur la base de données. cependant avec ce système, l'écriture de données repose sur un seul serveur. La solution à ce problème est l'utilisation de la notion de sharding. Un shard se compose d'un ou de plusieurs serveurs : un serveur seul, un couple maître/esclave ou un replica set. Pour partitionner une collection sur plusieurs shards on définit sur celle-ci une shard key. Il s'agit d'une liste de champs du document qui permettront au système de définir sur quel shard un document doit être stocké. Les données sont alors réparties automatiquement de manière la plus homogène entre les différents shards.

Pour configurer le partitionnement de données, deux éléments d'architectures sont ajoutés :

- Un ou plusieurs serveurs mongos dont le rôle est de router les requêtes du client vers le shard approprié.

– Les serveurs de configurations qui stockent la configuration du sharding et sont utilisés par les autres instances afin d'être en mesure de déterminer sur quel shard les opérations doivent être effectuées. Une telle architecture est illustrée par la figure ci-dessous :

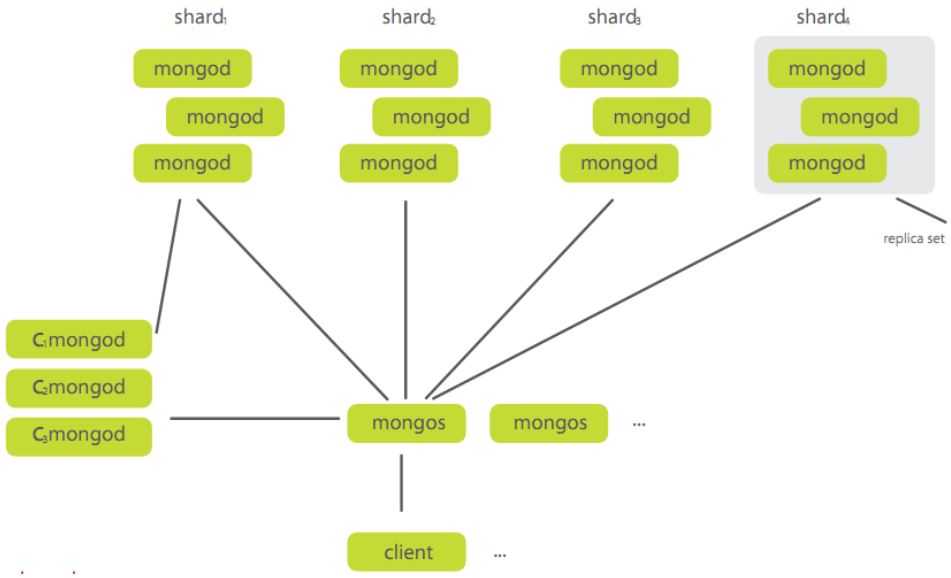


Figure A.4 : fonctionnement via sharding [102]

A.2.4 Modèle de requête

Le modèle de requête comprend le langage de communication et les requêtes qu'il est possible de faire sur la base de données. Le langage de requête du mouvement NoSQL est fortement différent d'une base de données à une autre. Dans le cas de MongoDB, la littérature parle d'un langage de requêtes basé document assez riche, pour une prise en main très rapide. À titre d'exemple, la méthode *sort()* permet de trier les documents retournés, *limit()* permet de limiter le nombre de documents retournés et *skip()* permet de ne retourner que les n premiers documents. MongoDB offre aussi une liste de méthodes pour l'agrégation des données sous son « *aggregation framework* ». Ce dernier comprend les méthodes suivantes :

– *count()* qui retourne le nombre de documents de la collection ou correspondant aux critères de recherche.
– *distinct()* qui retourne une liste de valeurs distinctes pour une clé donnée à travers une collection.
– *group()* similaire à la fonction SQL *GROUP BY* cette méthode permet d'obtenir un tableau d'éléments groupés.

De plus, MongoDB offre des possibilités d'interrogation en Map/Reduce. Pour effectuer un certain nombre de calculs efficacement, MongoDB permet l'écriture de fonctions map() et reduce() en javascript. Les résultats peuvent être stockés dans des collections temporaires pour conserver un cache ceux-ci.

RÉFÉRENCES

1. Aas K., Eikvil L., "Text Categorisation : A Survey", Rapport technique, Norwegian Computing Center, 1999.

2. Abellò A., Romero O., "On-Line Analytical Processing (OLAP)". In Encyclopedia of Database Systems, Springer, 2009, pp. 1949-1954.

3. Ahonen H., "Mining All Maximal Frequent Word Sequences in a Set of Sentences", CIKM, Proceedings of the 14th ACM international conference on Information and knowledge management, 2005, pp. 255-256.

4. Baid A., Balmin A., Hwang H., Nijkamp E., Rao J., Reinwald B., Simitsis A., Sismanis Y., Ham F.V., "DBPubs: Multidimensional Exploration of Database Publications", Proceedings of the VLDB Endowment, 2008.

5. Barzilay R., McKeown K., Elhadad M., "Information fusion in the context of multi-document summarization". In Proceedings of ACL '99.

6. Baziz M., Boughanem M., Aussenac-Gilles N., "Conceptual Indexing Based on Document Content Representation", LNCS 3507, 2005, pp. 171–186.

7. Baziz M., "Indexation Conceptuelle Guidée Par Ontologie Pour La Recherche D'information", thèse de doctorat de l'Université Paul Sabatier, (France), 2005.

8. Blumberg R., Atre S., "The problem with unstructured data", DM Review, 2003, pp. 42-46.

9. Boros E., Kantor P.B., Neu D.J., "A Clustering Based Approach to Creating Multi-Document Summaries". In Proceedings of the 24th ACM SIGIR Conference, 2001.

10. Boussaid O., Bentayeb F., Darmont J., Rabaseda S., "Vers l'entreposage des données complexes : structuration, intégration et analyse", Ingénierie des Sytèmes d'Information, 2003, Volume 8 - n◦ 5.

11. Boussaid O., Ben Messaoud R., Choquet R., Anthoard S., "X-Warehousing: An XML-Based Approach for Warehousing Complex Data", In Proceedings of the 10th East-European Conference on Advances in Databases and Information Systems (ADBIS'06), LNCS 4152, 2006, pp. 39–54.

12. Boussaïd O., Darmont J., Bentayeb F., Rabaseda S. L., "Warehousing complex data from the Web", Int. J. Web Engineering and Technology, 2008.

13. Boussaid O., "Évolution de l'entreposage des données complexes", habilitation à diriger les recherches (HDR), Université Lumière - Lyon 2 (France), 2006.

14. Carbonell J. G., Goldstein J., “The use of MMR, diversity-based reranking for reordering documents and producing summaries”. In A. Moffat, & J. Zobel (Eds.), Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval, Melbourne, Australia, 1998, pp. 335–336.

15. Chaudhuri S., Dayal U., “An Overview of Data Warehousing and OLAP Technology”, ACM SIGMOD RecordACM Press, 1997, vol.26(1), p. 65–74.

16. Codd E.F, “Providing OLAP (On-Line Analytical Processing) to user-analysts : an it mandate”. Technical report, E.F. Codd and Associates, 1993.

17. Cody W.F., Kreulen J.T., Krishna V., Spangler W.S., “The integration of business intelligence and knowledge management”, IBM systems journal, 2002, Vol 41, No 4, pp. 697-713.

18. Cornuéjols A., Miclet L., “Apprentissage Artificiel, Concepts et algorithmes ”, édition Eyrolles, 2ème tirage 2003.

19. Darmont J., Boussaid O., “Processing and managing complex data for decision support”, Idea Group Publishing, 2006.

20. Dempster A. P., Laird N. M., Rubin D. B., “Maximum Likelihood From Incomplete Data Via The EM Algorithm”, Journal Of The Royal Statistical Society, Series B, 1977, 39(1), pp 1–38.

21. Edmundson H. P., “New Methods in Automatic Extracting”, Journal of the ACM (JACM), 1969, 16(2), 264–285.

22. Fayyad U., Uthurusamy R., “Evolving data mining into solutions for insights”. Communications of the ACM, 2002, 45(8):28–31.

23. Feldman R., Sanger J., “The Text Mining Handbook Advanced Approaches in Analyzing Unstructured Data”, Cambridge University Press, 2007.

24. Forest D., “Vers une nouvelle génération d’outils d’analyse et de recherche d’information”, Documentation et Bibliothèques, 2009, Vol. 55, Nº. 2, pp. 77-89.

25. Ghribi M., Cuxac P., Lamirel J.C., Lelu A., “Mesures de qualité de clustering de documents : Prise en compte de la distribution des mots clés”, 10ième Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances, 2010.

26. Goldstein J., Mittal V., Carbonell J., Kantrowitz M., “Multi-document summarization by sentence extraction”. In NAACL-ANLP 2000 Workshop on Automatic summarization, 2000, pages 40–48.

27. Gray J., Bosworth A., Layman A., Pirahesh H., “Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total”, 12th International Conference on Data Engineering (ICDE’96), IEEE Computer Society, 1996, pp.152-159.

28. Hahn U., Mani I., “The Challenges of Automatic Summarization”, In IEEE computer journal, 2000, 33(11), pp. 29 – 36.

29. Halkidi M., Batistakis Y., Vazirgiannis M., “Clustering Validity Checking Methods : Part II”. ACM SIGMOD, 2002, vol.31, No 3, pp. 19-27.

30. Han J., Kamber M., “Data Mining: Concepts and Techniques”, second edition, The Morgan Kaufmann Publishers, 2006.

31. Han E.H., Karypis G., “Centroid-based Document Classification: Analysis and Experimental Results”. Principles of Data Mining and Knowledge Discovery, 2000, p. 424–431.

32. Harbison A., Ryan P., “The problem of analysing unstructured data”, Grant Thornton, 2009.

33. Hofmann T., “Probabilistic latent semantic analysis”, In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, 1999, pp. 50–57.

34. Hovy E. H., “Text Summarization”. The Oxford Handbook of Computational Linguistics, édité par Ruslan Mitkov, chapitre 32. Oxford University Press, 2005, pp. 583-598.

35. Inmon W.H., Strauss D., Neushloss G., “DW 2.0: The Architecture for the Next Generation of Data Warehousing”, edition Morgan Kaufmann, 2008.

36. Inmon W.H, “The Business value of textual integration”, white paper, textual ETL, Forest Rim Technology, 2007.

37. Jain A. K., Murty M. N., Flynn P. J., “Data clustering : a review”, ACM Computing Surveys, 1999, 31(3), pp. 264–323.

38. Jain A. K., “Data clustering : 50 years beyond k-means”. Pattern Recognition Letters, 2009, 31(8) : pp.651–666.

39. Joachims T., “Text categorization with support vector machines : learning with many relevant features”. 10th European Conference on Machine Learning, 1998, pp. 137–142.

40. Keikha M., Razavian N. S., Oroumchian F., Seyed Razi H., “Document Representation and Quality of Text: An Analysis”, dans “Survey of Text Mining: Clustering, Classification, and Retrieval”, Second Edition, Michael W. Berry, Malu Castellanos, Springer, 2007.

41. Khrouf K., Soulé-Dupuy C., “Conception d'entrepôts de documents décisionnels”, INFORSID, 2001, pp. 387-401.

42. Khrouf K., Soulé-Dupuy C., “DocWare: Vers l'entreposage et l'analyse multidimensionnelle de documents”, Conférence en Recherche d'Information et Applications : CORIA, 2005, pp. 405-420.

43. Kim S.N., Medelyan O., Kan M.-Y., Baldwin T., “SemEval-2010 Task 5 : Automatic keyphrase extraction from scientific articles”, In Proceedings of SemEval 2010, Task 5 : Keyword extraction from Scientific Articles, 2010.

44. Loudcher S., “Vers l’OLAP sémantique pour l’analyse en ligne de données complexes”, habilitation à diriger les recherches (HDR), Université Lumière - Lyon 2 (France), 2011.

45. Luhn H. P., “The Automatic Creation of Literature Abstracts”, IBM Journal, 1958.

46. Macqueen J. B., “Some methods of classification and analysis of multivariate observations”, In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.

47. Mani I., Maybury M.T., “Advances in Automatic Text Summarization”, The MIT Press, 1999.

48. Manning C.D., Raghavan P., Schütze H., “An Introduction to Information Retrieval”, Cambridge University Press, 2008.

49. Mbarki M., Soulé-Dupuy C., Vallès-Parlangeau N., “Vers une exploitation flexible de documents multimédia”. INFORSID, 2005, pp. 95-112.

50. McKeown K., Klavans J., Hatzivassiloglou V., Barzilay R., Eskin E., “Towards multidocument summarization by reformulation: Progress and prospects”. In AAAI/IAAI, 1999, pp. 453-460.

51. McKeown K., Radev D., “Generating Summaries of Multiple News Articles”, dans Proceedings, 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995, pp.74-82.

52. Medelyan O., Witten I.H., “Thesaurus based automatic key-phrase indexing”. In Proceedings of ACM/IEED-CS JCDL, 2006, pp. 296–297.

53. Nguyen T.D., Kan M-Y., “Key phrase Extraction in Scientific Publications”, In Proceedings of ICADL, 2007, pp. 317–326.

54. Park B-K, Han H., Song I-Y, “XML-OLAP : A Multidimensional Analysis Framework for XML Warehouses”, 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), LNCS 3589, Springer, 2005, p. 32–42.

55. Park B-K, Song I-Y, “Toward Total Business Intelligence Incorporating Structured and Unstructured Data”, BEWEB'11 Proceedings of the 2nd International Workshop on Business intelligencE and the WEB, 2011.

56. Pérez J.M., R. Berlanga, M.J Aramburu, T.B Pedersen, “R-cubes : OLAP cubes Contextualized with documents”. ICDE 2007, IEEE 23rd International Conference, 2007, pp.1477- 1478.

57. Pérez-Martínez J.M., Berlanga-Llavori R., Aramburu-Cabo M.J., Pedersen T.B., “Contextualizing data warehouses with documents”, Decision Support Systems (DSS), Elsevier, 2007.

58. Ponte J.M., Croft W.B., “A Language Modeling Approach to Information Retrieval”. Research and Development in Information Retrieval, Proc. ACM-SIGIR, 1998, pp. 275-281.

59. Porter M.F., “An algorithm for suffix stripping”, Program: electronic library and information systems, 1980, Vol. 14 (3), pp.130 – 137.

60. Quinlan J.R., “Induction of decision trees”, Machine Learning, 1986, 1(1) pp. 81–106.

61. Radev D.R., Hovy E., McKeown K., “Introduction to the special issue on summarization”. Computational Linguistics, Volume 28, Number 4, 2002.

62. Radev D.R., Hongyan J., Małgorzata S., Tam D., “Centroid-based summarization of multiple documents”, Information Processing and Management, 2004, (40) pp. 919–938.

63. Ravat F., Teste O., Tournier R., Zurfluh G., “A Conceptual Model for Multidimensional Analysis of Documents”. 26th International Conference on Conceptual Modeling, Springer-Verlag, 2007, LNCS 4801, p.550-565.

64. Ravat F., Teste O., Tournier R., “OLAP Aggregation Function for Textual Data Warehouse”, In 9th International Conference on Entreprise Information System (ICEIS) , 2007, pp. 151-156.

65. Ravat F., Teste O., Tournier R., “Analyse multidimensionnelle de documents via des dimensions OLAP”, Document numérique, Lavoisier, 2007, 2 Vol. 10, p. 85-104.

66. Ravat F., Teste O., Tournier R., Zurfluh G., “Top_Keyword: an Aggregation Function for Textual Document OLAP”, 10th International Conference on Data Warehousing and Knowledge Discovery, 2008, pp.55-64.

67. Ravat F., “Modèles et Outils pour la conception et la manipulation de systèmes d’aide à la décision”, habilitation à diriger les recherches (HDR), Université de Toulouse 1 (France), 2007.

68. Salton G., Wong A., Yang C.S., “A Vector Space Model for automatic indexing”, CACM, 1975, 18(11), pp. 613-620.

69. Salton G., Fox E.A., Wu H., “Extended Boolean information retrieval system”, CACM, 1983, 26(11), pp. 1022-1036.

70. Salton G., Buckley C., “Term-weighting approaches in automatic text retrieval”, information processing and management, volume 24, N°5, 1988.

71. Schmid H., “Probabilistic Part-of-Speech Tagging Using Decision Trees”. Proceedings of International Conference on New Methods in Language Processing. 1994.

72. Sebastiani F., “Machine Learning in Automated Text Categorization”, ACM computing surveys, Vol. 34, No 1, 2002.

73. Shankar S., Karypis G., “Weight adjustment schemes for a centroid based classifier”, Principles of Data Mining and Knowledge Discovery, 2000.

74. Simitsis A., Baid A., Sismanis Y., Reinwald B., “Multidimensional Content eXploration”, Proceedings of the VLDB Endowment, 2008, 1(1), pp. 660–671.

75. Spärk-Jones K., Galliers J. R., “Evaluating Natural Language Processing Systems”, Technical Report 291, 1993.

76. Sullivan D., “Document warehousing and text mining : Techniques for Improving Business Operations, Marketing, and Sales”. John Wiley & Sons, 2001.

77. Tan S., Cheng X., “An Effective Approach to Enhance Centroid Classifier for Text Categorization”, PKDD, 2007, LNAI 4702, pp. 581–588.

78. Teste O., “Modélisation et manipulation des systèmes OLAP : de l’intégration des documents à l’usager”, habilitation à diriger les recherches (HDR), Université de Toulouse 1 (France), 2009.

79. Teufel S., Moens M., “Sentence extraction as a classification task”, Proceedings of the ACL Workshop on Intelligent Text Summarization (Madrid), 1997.

80. Torres-Moreno J. M., “Du textuel au numérique : analyse et classification Automatiques”, HDR, université d’Avignon, 2007.

81. Tournier R., “Analyse en ligne (OLAP) de documents”, thèse de doctorat de l’Université Toulouse III – Paul Sabatier, 2007.

82. Tseng F. S.C., Chou A. Y.H., “The concept of document warehousing for multi-dimensional modeling of textual-based business intelligence”, Science Direct, Decision Support Systems, 2006, (42) pp. 727–744.

83. Van Rijsbergen C.J., “Information retrieval”, Butterworth-Heinemann, 1979.

84. Witten I.H., Frank E., “Data Mining practical machine learning tools and techniques”, second edition, Morgan Kofmann Publishers, 2005.

85. Zhang D., Zhai C., Han J., “MiTexCluster: Micro-Text-Cluster Cube for Online Summarization of Text Cells in OLAP”, conference on intelligent data understanding (CIDU), 2009.

86. Zhang D., Zhai C., Han J., Srivastavaz A., Ozaz N., “Topic Modeling for OLAP on Multidimensional Text Databases: Topic Cube and its Applications”, Statistical Analysis and Data Mining (SDM’09), 2009.

87. Zhang D., Zhai C., Han J., “MiTexCube: MicroTextCluster Cube for Online analysis of Text Cells”. Conference on Intelligent Data Understanding (CIDU), 2011, pp. 204-218.

88. Zhang Q., Sun S., “A Centroid k-Nearest Neighbor Method”, LNCS 6440, 2010, pp. 278–285.

89. Dhillon I.S., Fan J., Guan Y., “Efficient clustering of very large document collections”, in Data Mining for Scientific and Engineering Applications, Grossman R. L., Kamath C., Kegelmeyer P., Kumar V., and Namburu R., Eds. Kluwer Academic publishers, 2001, pp. 357–381.

90. Zhong S., “Efficient Online Spherical K-means Clustering”, In Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN), 2005, pp. 3180-3185.

91. Lababou A., Benblidia N., “Summarization des Documents par Catégorisation dans les Text Cubes”, 6ème édition Atelier des Systèmes Décisionnels, 2012.

92. Zhao Y., Karypis G., “Criterion Functions for Document Clustering : Experiments and Analysis”, Technical Report #01-40, University of Minnesota, 2002.

93. Zhao Y., Karypis G., “Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering”, Machine Learning, 2004, 55, pp. 311–331.

94. Li B., Guoyong Y., “Improvement of TF-IDF Algorithm Based on Hadoop Framework”, The 2nd International Conference on Computer Application and System Modeling, 2012, pp. 391-393.

95. Yap I., Loh H.T., Shen L., Liu Y., “Topic detection using MFSs”. In Proceedings of the 19th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, Annecy, France, 2006.

96. Zhan J., “Exploiting Textual Structures Of Technical Papers For Automatic Multi-Document Summarization”, thèse de doctorat, Université de Science et de Technologie, Chine, 2007.

97. Liu Y., Loh H.T., Lu W.F., “Deriving Taxonomy from Documents at Sentence Level”, dans Emerging Technologies of Text Mining: Techniques and Applications, Hércules Antonio do Prado, Edilson Ferneda, Information science reference, Hershey-New York, 2008, pp. 99-119.

98. Liu Y., “On Document Representation and Term Weights in Text Classification”, dans Handbook of Research on Text and Web Mining Technologies, Information science reference, Hershey-New York, 2009, pp. 1-22.

99. Ahonen H., “Finding all maximal frequent sequences in text”, in: Proceedings of ICML-99 Workshop, Machine Learning in Text Data Analysis, 1999.

100. Plugge E., Membrey P., Hawkins T., “The Definitive Guide to MongoDB : The NoSQL Database for Cloud and Desktop Computing”, Apress edition, 2010.

101. Devlin B., “Business Intelligence-NoSQL… No Problem”, white paper, 9sight Consulting, mai 2012.

102. 10gen, “MongoDB a brief introduction”, white paper, 2011.