RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÉRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
Université Ammar Thélidji Laghouat

Faculté des sciences
Département de mathématique et Informatique

MÉMOIRE DE MASTER

DOMAINE : MATHEMATIQUE ET INFORMATIQUE (MI)
FILIERE : INFORMATIQUE
OPTION : SYSTEME D'INFORMATION DECISIONNEL

Présenté par :
Benzian Khadidja
Cherif Asma

Théme :

ÉTUDE COMPARATIVE ENTRE DEUX FONCTIONS D'AGREGATION DES DONNEES TEXTUELLES

Soutenu publiquement devant le jury composé de :

Melle. A.BELABBASI	Université de Laghouat	Présidente
Mr. A.ZIANI	Université de Laghouat	Examinateur
Mr. Y. GUELLOUMA	Université de Laghouat	Examinateur
Dr. Y.OUINTEN	Université de Laghouat	Encadreur
Mr. M.BOUAAKAZ	Université de Laghouat	Co_Encadreur

ANNÉ UNIVERSITAIRE 2012-2013

DEDICACES

Nous dédions ce mémoire
à
nos parents
nos frères et soeurs
et
nos amis

Remerciements

Nous remercions en premier lieu Dieu qui nous a accordé la force et la patience pour accomplir ce travail.

Nous exprimons toute la reconnaissance que nous éprouvons envers tous ceux qui ont contribué par leurs conseils et leurs encouragements à l'aboutissement de ce travail.

Nous tenons à remercier Monsieur Youcef OUINTEN de nous avoir encadré et de nous dirigé, ainsi que notre co-encadreur Mr Mustapha BOUAKKAZ pour ses orientations ses encouragements nous ont donné la volonté durant notre projet de fin d'étude.

Nos remerciements à toutes les personnes de l'université d'Amar TELIDJI Laghouat enseignants, administrateurs, et plus particulièrement, ceux du dèpartement de Mathématiques et Informatique.

Nous exprimons nos vifs remerciements aux membres du jury qui ont bien voulu accepter de juger ce travail.

A nos familles et nos amis, nous adressons un grand merci.

Résumé

Ce mémoire présente une étude comparative entre les fonctions d'agrégation des données textuelles contenues dans un corpus de documents.

Nous allons d'abord introduire les outils de stockage de ce type de données qui sont les entrepôts de données, ainsi que les outils d'analyse, d'agrégation et de visualisation de données qui sont traduites par les systèmes OLAP.

Nous intéressons dans ce travail aux fonctions d'agrégation des données textuelles, nous présentons les principes de quelques travaux dans ce domaine, parmi ces travaux nous avons choisi les fonctions Top-Keyword et Topic en utilisant deux méthodes pour calculer les poids des termes avec la mesure tf.idf.

Nous avons implémenté ces deux fonctions et testé leurs performances sur un corpus de documents réels dans le but de comparer et interpréter les résultats obtenus.

Mots-clés : entrepôts de données, documents, Top keyword, Topic, la mesure tf.idf, agrégation.

Abstract

This thesis presents a comparative study of aggregation functions of textual data in a corpus of documents. We will first introduce the storage tools for this type of data are data warehouses and analysis, aggregation and visualization tools of data that are OLAP systems.

We are interested in this work to aggregation functions of textual data, we present some principles of work in this topic, and including these works we have chosen the Top-Keyword and Topic functions using two methods to calculate the weights of terms with tf.idf measure.

We have implemented these two functions and tested their performances on a real corpus of documents in order to compare and interpret their results.

Table Des Matiéres

Table des figures

Liste des tableaux

Introduction

Les documents textuels connaissent un avantage grandissant dans la circulation des informations dans les entreprises. Ainsi, ces documents sont multi structurés et Parviennent de différentes sources (multi source). Les décideurs doivent utiliser ces documents dans leur processus de prise de décision, ce qui provoquent l'augmentation de volume de données textuelles de manière considérable dans les systèmes d'information des entreprises. Pour cela les entreprises doivent utiliser des outils et des techniques qui permettent de stocker et d'agréger la grande masse de données pour rendre facile l'exploitation et l'analyse de ces données. L'agrégation consiste à réduire le volume de données à visualiser lors de l'analyse. Les fonctions d'agrégation utilisées prennent donc en entré une grande masse de données et donnent en sortie une vision agrégée et globale de ces données. Puisque les données sont de types différents, les fonctions d'agrégations sont divisées en deux catégories : les fonctions d'agrégation des données numériques et les fonctions d'agrégation des données textuelles.

L'objectif de ce mémoire est de faire une étude comparative entre quelques fonctions d'agrégation des données textuelles, ces foncions font l'objet de nombreux travaux de recherche dont le but est d'analyser le contenu textuel des documents.
Dans le cadre de notre travail, nous avons organisé ce mémoire en quatre chapitres.

Dans le premier chapitre, Nous présentons l'environnement d'un système décisionnel qui est constitué des entrepôts de données textuels, des magasins de données et de la technologie OLAP qui est étendue pour la manipulation des données textuels. Ces outils permettent d'organiser et de représenter la grande masse de données textuelles sous forme agrégées à l'aide des fonctions d'agrégation de données textuelles fournies dans l'environnement OLAP.

Dans le deuxième chapitre, nous nous intéressons à l'état de l'art et aux travaux de recherche sur les fonctions d'agrégation des données textuelles. Nous discutons sur ces travaux et expliquons les principes de quelques fonctions d'agrégations. Ces fonctions opérent sur le contenu textuel des documents et leur objectif est de constituer au sein d'un corpus une vision synthétique des informations issues de documents.

Dans le troisième chapitre, Nous avons expliqué en détail les principes des fonctions que nous avons choisies pour faire notre étude. En utilisant un exemple d'application nous avons essayé d'expliquer leurs fonctionnements.

Dans le quatrième chapitre, nous avons commençons par présenter l'implémentation des fonctions choisies, ensuite nous décrivons le corpus des documents utilisés dans nos expérimentations. À la fin nous avons interprété et comparé les résultats trouvés après l'exécution de chaque fonction et cité quelques inconvénients de ces fonctions.

La conclusion de ce mémoire synthétise les études faites durant la réalisation de ce mémoire. Ainsi que les domaines d'application des fonctions d'agrégation des données textuelle.

Chapitre 1

Les entrepôts de données textuelles

Actuellement, la prise de décision qui est basée sur l'intuition n'est pas suffisamment et ne peut s'appliquer qu'à des problèmes familiers. Il est nécessaire d'utiliser des données textuelles pour la prise de décision, ces données sont sous forme de documents non structurés et ayants des formats différents (HTML, XML) A cet effet, le volume de données qui passent au système d'information des entreprises augmente rapidement et engendre la difficulté de leur exploitation. Pour mieux exploiter ces données dans le processus décisionnel, les entreprises utilisent des systèmes d'aide à la décision, qui répondent au stockage et à la représentation sous forme agrégé des données volumineuses en proposant des outils et des techniques de traitement et de manipulation de données.

1 Architecture d'un systéme décisionnel

Nous commençons par la présentation de l'architecture d'un système décisionnel,dans la Figure 1.1

Au niveau stockage, on trouve deux composants :

- L'entrepôt de données qui constitue la mémoire de l'entreprise, il archive toutes les données (des bases de données relationnelles ou des données externes à l'entreprise) nécessaires à la prise de décision. Le chargement des données dans l'entrepôt implique l'utilisation de processus ETL (Extract, Transform, Load) qui garantie l'homogénéité des données sources afin d'intégrer ces donnés au schéma de l'entrepôt, ce processus est effectué de manière périodique pour garder toutes les versions des données dans l'entrepôt.
- Le magasin de donnée est un fragment de donnés extrait (ETL) de l'entrepôt de données qui concerne un sujet d'analyse pour un groupe d'utilisateur.

Au niveau de l'analyse : on trouve les systèmes OLAP qui fournissent des outils d'analyse, d'agrégation et de la visualisation des données.

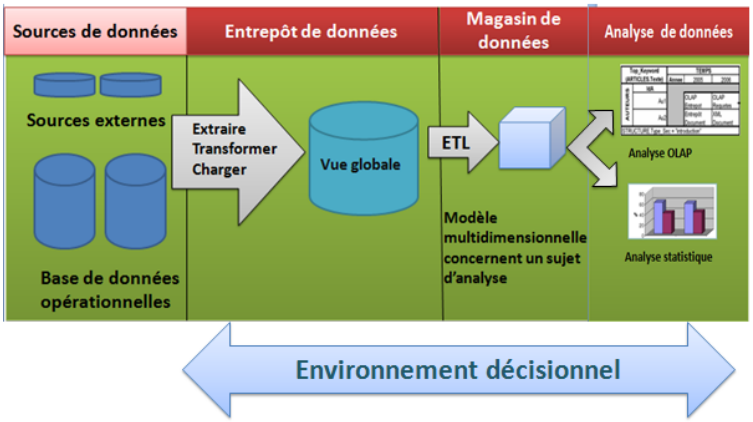

FIGURE 1.1 – Architecture d'un systéme décisionnel

A partir de cette architecture, on peut déduire qu'il y a des outils de stockage (les entrepôts de données, et les magasins de données) et des outils d'interrogations (les systèmes OLAP), qu'on va bien les expliquer ces outils dans les sections suivantes.

2 Les entrepôts de données

« Le concept d'entrepôt de données a été proposé par W. H. Inmon en 1990 pour répondre à des besoins d'analyse pour les décideurs » [Lam06].

Un entrepôt de données stocke les données en provenance de plusieurs sources, souvent réparties et hétérogènes et dans le but de donner une vision globale de l'information aux analystes et aux décideurs. Les données stockées dans l'entrepôt sont issues des applications de l'entreprise elles sont restituées et analysées par les utilisateurs. Il fournit aux décideurs un environnement homogéne et historié [Abd10].

Caractéristiques des entrepôts de données :

« selon Bill Inmon, un entrepôt de données est une collection de données orientées sujet, intégrées, non volatiles, historisées. » [Fav07].

Orientées sujet : l'entrepôt de données est organisé par des thèmes (sujets) de l'entreprise.

Intégrées : Les données proviennent de sources différentes, elles sont regroupées au sein d'une vision homogène.

Historisées : toutes les versions des données sont stockées pour pouvoir faire l'analyse au cours du temps.

Non volatiles : les données stockées au sein de l'entrepôt ne peuvent pas être modifiées ou supprimées pour garder la traçabilité des informations.

Les entrepôts de données offrent une vision uniforme des données qui seront extraites en fonction de besoin d'analyse pour alimenter le magasin de données.

3 Le magasin de données (le datamart)

Un magasin de données est un sous ensemble de données, extrait d'un entrepôt de données pour répondre à un besoin particulier d'utilisateurs concernant un sujet d'analyse spécifique (par exemple un DataMart Marketing, un DataMart Commercial) pour la résolution d'un problème, contenant un volume de données plus petit et rapidement analysées.

En conséquence, les magasins de données et les entrepôts de données effectuent un travail collaboratif, l'organisation des données dans l'entrepôt construite à partir d'intégration des données déjà existantes de telle manière à les réutiliser ultérieurement, alors que la réorganisation des données dans le magasin construite en vue d'assurer une analyse des besoins de l'utilisateur (marketing, finances). En générale, les données de l'entrepôt sont stockées de manière permanente tandis que le magasin stocke les données à court terme.

4 Modélisation multidimensionnelle des données

4.1 Représentation multidimensionnelle (cube de données)

Selon [Ron07], Les magasins de données s'appuient sur une modélisation multidimensionnelle, qui permet de représenter les données d'un magasin sous forme de points dans un espace à plusieurs dimensions par un cube de données afin de faire l'analyse en ligne (OLAP) multidimensionnelle.

Exemple :
La Figure 1.2 présente un exemple de cube de données, qui permet l'analyse des ventes d'un magasin informatique. Il modélise les activités de ventes de ce magasin. Les quantités des ventes représentent les indicateurs d'analyse du sujet Ventes. Cet indicateur peuvent être analysé en fonction de trois dimensions : LES PRODUITS vendus, LE TEMPS correspondant aux dates de ventes et LES REGIONS correspondants au lieu où ont été effectuées les ventes. Les dimensions TEMPS et REGIONS disposent de deux niveaux de détail (mois et année) et (ville, pays) respectivement, la dimension PRODUIT dispose d'un seul niveau de détail, ces dimensions permettent de visualiser les données de façon détaillée au cours des analyses.

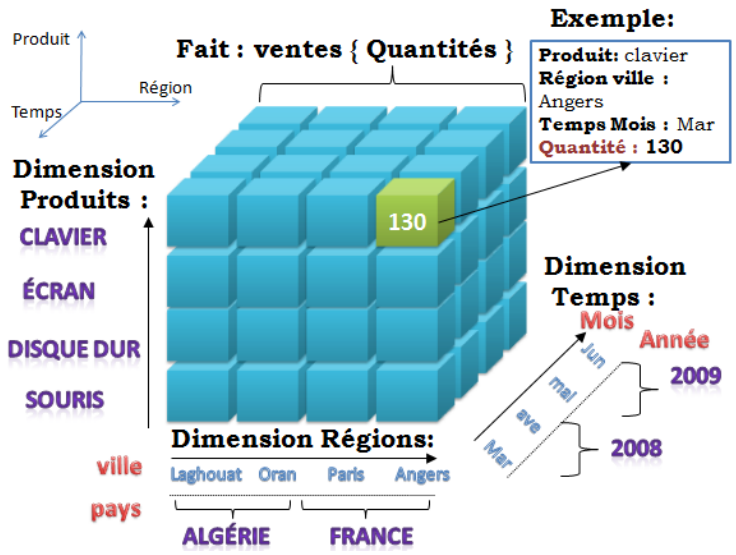

FIGURE 1.2 – Exemple d'un Cube de données.

4.2 Modèle conceptuel de représentation de données

La modélisation multidimensionnelle des données permet de visualiser les données sous forme adaptée à l'analyse, cette modélisation est basée sur deux concepts fait et dimension [FRA07].

Le fait représente le sujet d'analyse. Un fait regroupe un ensemble **de mesures**. Une mesure correspond à l'élément de donnée analysée sur lequel on va appliquer des fonctions d'agrégation. Ces fonctions permettent d'agréger les données en fonction des axes d'analyses, **Une dimension** est un axe d'analyse selon lequel sont visualisées les mesures d'un sujet d'analyse. Une dimension constitue un ou plusieurs attributs appelés **paramètres**, Ces paramètres sont organisés en partant d'un faible niveau de détail vers des données plus

détaillées qui s'appellent **hiérarchie**. Une hiérarchie permet lors des analyses de diminuer ou d'agrandir les niveaux de détail de l'analyse. Donc on peut définir un schéma en étoile comme un fait et ses dimensions regroupantes les paramètres l'analyse.

Exemple :
Dans la Figure 1.3 , les analystes analysent les ventes avec la mesure (*Quantités*) de l'exemple précédent, en fonction de *TEMPS*, des *PRODUITS* et des *REGIONS*, ces dimensions sont précisées avec les différents niveaux de détails (paramètres).

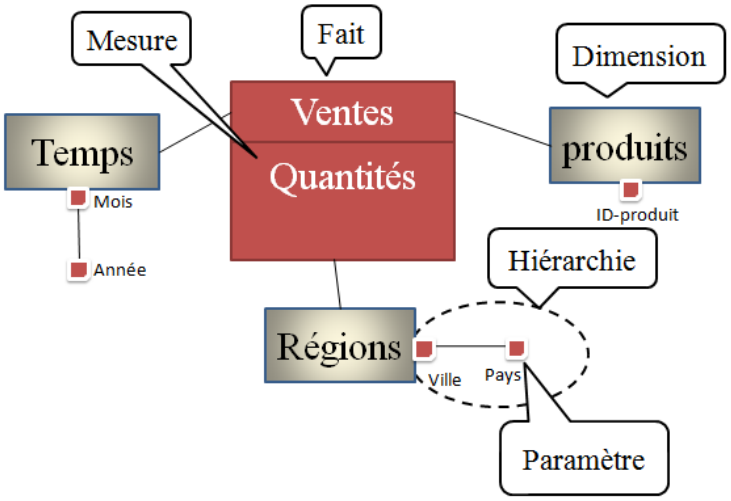

FIGURE 1.3 – Exemple de visualisation multidimensionnelle du précédent cube.

5 L'analyse OLAP (On Line Analytical Processing) de données multidimensionnelles

OLAP est un système qui permet aux décideurs d'analyser une grande masse de données. L'analyse des données dans un environnement OLAP correspond aux agrégations, interrogations et visualisations et des données.

5.1 Les opérateurs de manipulation dans l'environnement OLAP

L'analyse des données multidimensionnelles à divers niveaux de détail s'effectue grâce à des opérateurs classiques fournis dans l'environnement OLAP. Ces opérateurs agissent sur un cube OLAP.

- **La rotation :** permet de changer un axe d'analyse (dimension) par un autre, ou de changer la hiérarchie sur une même dimension ;
- **Les forages :** le forage vers le haut (ROLL-UP) permet de passer d'un niveau de granularité plus détaillé à un autre moins détaillé sur une dimension. Le forage vers le bas (DRILL-DOWN) contrairement au Roll-up, cette opération permet de passer d'un niveau de granularité moins détaillé à un autre plus détaillé sur une dimension ;
- **Nest :** permet d'imbriquer un paramètre d'une dimension sous un autre paramètre d'une autre dimension ;
- **Push :** permet de transformer un paramètre en une mesure ;
- **Pull :** permet de transformer une mesure de fait en un paramètre ;

- **Cube :** c'est l'opération de calculs d'agrégat qui permet d'ajouter dans une table multidimensionnelle des calculs agrégeants les lignes et/ou les colonnes ;
- **unCube :** c'est l'opération inverse.

5.2 Les fonctions d'agrégation dans l'environnement OLAP

Ces fonctions agissent sur les mesures [Ron07]
- **SUM :** calculer la somme numérique de l'agrégat ;
- **AVG :** calculer la moyenne d'un agrégat ;
- **MIN :** donner la valeur minimale d'un agrégat ;
- **MAX :** donner la valeur maximale d'un agrégat ;
- **COUNT :** calculer le nombre d'instances dans un agrégat.

5.3 La visualisation multidimensionnelle des données

La visualisation multidimensionnelle des données est faite à l'aide d'un tableau de deux dimensions qui permet de représenter toutes les combinaisons possibles entre deux dimensions (axes d'analyse). Chaque cellule correspond à un agrégat de la mesure. Pour plus de deux dimensions, il n'est pas possible de visualiser toutes les données simultanément, il faut utiliser des opérations puissantes pour manipuler les données, Par exemple permuter les dimensions pour visualiser l'information suivant les différentes dimensions [FRA07].

Exemple :
Dans la Figure 1.4 , la Quantité des unités vendues de la ville d'Oran de Région Algérie est agrégée par la fonction *SUM* par rapport aux dimensions produit et mois.

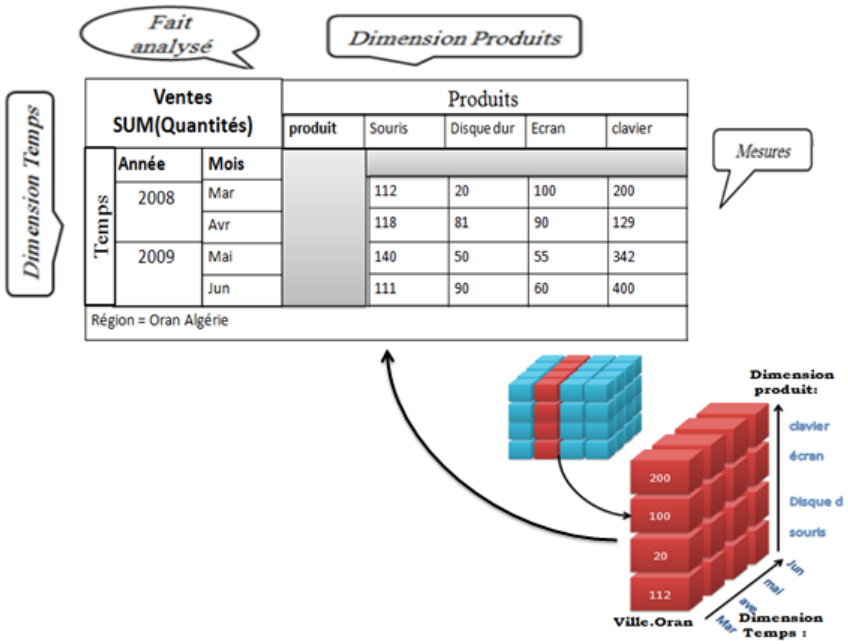

FIGURE 1.4 – La visualisation multidimensionnelle des données.

6 Les entrepôts de données textuelles

6.1 problématique

Les documents textuels ont toujours été considérés comme des documents non structurés (documents libres), provenants de sources hétérogènes et augmentent de manière considérable, utilisés par les décideurs dans leur processus de prise de décision. Ces documents doivent être soumis à des prétraitements pour les rendre exploitables par les systémes décisionnels. Ces systémes font face aux problémes de stockage et d'analyse de données textuelles volumineuses.

6.2 Alimentation des documents dans l'entrepôt de données textuelles

Les documents textuels sont hétérogènes en structure et en contenu. L'hétérogénéité en structure se traduit par des documents semi structurés et non structurés. L'hétérogénéité en contenu se traduit par la diversité des domaines des documents. Un modèle générique d'entrepôts de documents textuels permet de modéliser les structures et le contenu des documents afin de stocker les documents hétérogènes tels que les documents semi-structurés (HTML) ou non structurés (texte) [FRA00].

Un entrepôt de données textuelles doit être alimenté par le contenu des documents textuels de l'entreprise et de n'importe quel type (semi structuré ou non structuré), ces documents proviennent à l'origine de l'entreprise (e-mails) ou proviennent de Web (HTML). Ces documents sont utiles dans le processus de prise de décision. Avant de stocker ces documents dans l'entrepôt, il est nécessaire de les rendre homogènes pour favoriser l'analyse multidimensionnelle de ces documents dans un espace de temps raisonnable.

« Les entrepôts de documents permettent aussi la persistance des documents jugés pertinents. Les informations issues du Web par exemple sont très volatiles, c'est à dire qu'elles évoluent rapidement et peuvent disparaître, être modifiées ou encore changer de localisation fréquemment » [KM07].

6.3 Analyse multidimensionnelle des documents textuels

Les magasins de données (datamart) comme nous l'avons présenté précédemment, permettent de représenter les sujets à analyser selon différents axes d'analyses (dimensions). Etant donné que les entrepôts de données sont étendus au stockage des documents textuels, on peut faire une analyse multidimensionnelle d'un ensemble de documents concernant un domaine spécifique. Les documents à analyser sont généralement transportés de l'entrepôt de données vers le magasin de données, qui permettent de faire des analyses multidimensionnelles sur le contenu de ces documents.

6.3.1 Modélisation des données textuelles

Dans un environnement OLAP d'analyse multidimensionnelle de données textuelles, La structure, la métadonnée et le contenu des documents textuels sont transformés en (faits) et en axes d'analyse (dimensions) avec l'ajout de mesures textuelles. Cet environnement fournit une vision précise d'un ensemble de documents [FO07]. L'environnement OLAP permet aussi d'analyser le contenu d'un entrepôt de documents. Il est dispose de fonctions d'agrégation adaptées à ce type de données qui permettent d'obtenir une représentation agrégée des données textuelles à analyser.

Donc on peut dire que le contenu textuel des documents peut être analysé de la même façon que les données numériques. L'environnement OLAP fournit des fonctions d'agrégation spécifique aux mesures textuelles et visualise le résultat de l'agrégation sous forme des mots clés les plus représentatifs.

6.3.2 Agrégation des données textuelles

Selon le type des données à agréger, il existe deux catégories des fonctions d'agrégations. La premiére catégorie des fonctions s'applique sur une mesure numérique et la deuxiéme catégorie s'applique sur une mesure textuelle [FO08].

– ***Typologie des mesures dans l'environnement OLAP :***

1. **Une mesure numérique :** constituée des données numérique. Ces données sont additives (on peut appliquer toutes les fonctions d'agrégation numérique) ou semi-additives (des quantités de stock).
2. **Une mesure textuelle :** constituée des données textuelles représentées sous forme d'un ensemble de mots, un paragraphe ou un document complet. Deux types de mesures textuelles sont définis :
   - *Une mesure textuelle brute :* le contenu de cette mesure correspond à un document complet ou un fragment de documents (par exemple le contenu d'un article au format XML privé des balises XML qui le structurent).
   - *Une mesure textuelle élaborée :* le contenu de cette mesure est obtenue á partir d'une mesure textuelle brute après l'application d'un certain nombre de prétraitements sur cette mesure textuelle brute. Par exemple après l'application d'un prétraitement concerne le retrait de mots vide de sens sur une mesure textuelle brute il donne un

ensemble de mots clés les plus significatifs, ces mots clés représentent une mesure textuelle élaborée.

– ***Les fonctions d'agrégation :***

1. **Fonctions d'agrégation numériques :** c'est les fonctions classiques que nous avons déjà défini (sum, min, max, avg...).
2. **Fonction d'agrégations textuelles :** c'est l'ensemble des fonctions appliquées sur une mesure textuelle. Toutefois, dans [FO08], la fonction Top-Keyword permet de donner les n mots clés les plus significatifs. La fonction AVG-KW permet de donner un ensemble de mots clés agrégés à partir d'un ensemble de mots clés [Ron07].

Dans ce chapitre nous avons présenté l'importance des systèmes d'aide à la décision pour mieux gérer la grande masse de données qui circule dans les systèmes d'information des entreprises. Le système OLAP est de nos jours étendu pour supporter non seulement des analyses numériques mais aussi des analyses sur le contenu textuel des documents. Il utilise les différentes fonctions d'agrégation adaptées à ce type pour aider les décideurs à la prise de bonne décision.

Dans le chapitre suivant on va présenter les travaux réalisés dans le contexte d'agrégation des données textuelles.

Chapitre 2

Etat de l'art

Nous présentons dans ce chapitre un état de l'art qui porte sur quelques travaux qui ont été proposés pour analyser et agréger les données issues de documents textuelles, nous discutons et expliquons dans ce qui suit ces travaux afin de donner une idée générale sur les principes de quelques fonctions d'agrégation.

• *Etat de l'art sur les fonctions d'agrégation des données textuelles :*

Les fonctions d'agrégation présentées dans la suite sont des fonctions agissantes sur les données contenues dans les documents textuelles. Ces fonctions peuvent être classées en trois catégories :

- Les approches basées sur les connaissances linguistiques.
- Les approches statistiques.
- Les approches basées sur l'utilisation des connaissances externes.

Il y a d'autres techniques hybrides qui combinent entre ces trois approches.

1 Approches basées sur les connaissances linguistiques

Ces approches utilisent les caractéristiques linguistiques des mots ainsi que les travaux classés dans cette catégorie, les auteurs considèrent qu'un corpus est décrit par le vocabulaire contenu dans les documents. L'approche linguistique comprend l'analyse lexicale et l'analyse syntaxique.

1.1 Analyse lexicale

Pour l'aspect lexical les connaissances sont utilisées dans plusieurs travaux afin de bien classer les documents d'un corpus.

Dans [CG06] Poudat et al ont utilisé la classification thématique ou domaniale où les textes dans ce cas sont réduits à l'état de (sacs de mots). Les domaines représentent le plan lexical. Ils ont exposé une classification de données textuelles de discours scientifiques basée sur les noms parce qu'ils sont des parties du discours non vides, davantage susceptibles de pointer sur des concepts scientifiques, que les adverbes, verbes ou adjectifs et ils peuvent aisément être extraits. Après la classification des noms en domaines, ils peuvent agréger chaque classe par un seul nom qui représente tout le domaine.

Dans [AD08] , les auteurs ont effectué une expérience d'extraction de liste de fréquences lexicales spécifiques à la langue arabe à partir d'un corpus journalistique brut de deux millions de mots (quotidien Al-Hayât) au moyen du logiciel AraConc.
AraConc est un logiciel de concordance et de calcul de fréquences pour les mots arabes. Il traite des textes écrits en arabe, et fait l'analyse mot par mot. Le logiciel fait l'extraction des mots et les stocke dans des fichiers spécifiques, en vue de différents regroupements et de l'établissement de concordances, ensuite il propose l'étiquette racine pour agréger le groupe des mots.

1.2 Analyse syntaxique

Dans [BEC09], l'auteur a indiqué que l'analyse syntaxique est basée sur la sélection de syntagmes qui peuvent être définis par « l'extension logique de la sélection de catégories lexicales pouvant être des noms, des verbes, des adverbes, etc En effet un syntagme peut

être défini comme un ensemble de mots formants une unité catégorielle et fonctionnelle et constituant une unité sémantique. Une définition plus détaillée et complète de la notion de syntagme se trouve dans [Bou87].
Plusieurs travaux ont été consacrés aux méthodes d'extraction des mots-clefs basés sur l'aspect syntaxique.

Habert et al [BA96] ont exploité les relations entre les éléments de syntagmes nominaux (les noms et les adjectifs) pour l'extraction des mots-clefs. Leur principe est de réduire récursivement les termes candidats trouvés par un extracteur en mesurant la proximité entre deux mots simples par le nombre de contextes partagés par ces mots. Cette méthode n'est pas efficace, car elle nécessite d'une part l'utilisation de la sémantique pour définir les relations entre les syntagmes nominaux, et l'intervention d'un expert pour interpréter et valider les syntagmes proposés d'autre part.

Dans [DN03] pour l'extraction des mots clefs, l'auteur a proposé une typologie e contenante deux étapes : (1) acquisition de termes, regroupe les outils qui permettent l'extraction de termes candidats à partir du corpus analysé, (2) structuration de termes et regroupement conceptuel, elle nécessite d'une part des outils d'aide à la structuration d'ensembles de termes et d'autre part, des outils de repérage de relation pour effectuer la classification de termes.

2 Les approches statistiques

Ces approches sont simples et ne nécessitent pas de connaissances préalables sur le domaine, elles sont basées sur des informations statistiques pour attribuer un score de qualité aux mots clefs extraits. Les méthodes proposées pour l'approche statistique permettent de regrouper tous les concepts dans des classes dont la corrélation entre les termes est définie par des critères. Plus que le critère de corrélation est important, plus il y a de chance de trouver des liens sémantiques entre les membres de la classe.

Parmi les méthodes qui sont proposées dans cette catégorie, on peut citer la méthode LSA (Latent Semantic Analysis) [TP98]. Dans cette méthode le corpus est représenté par une matrice d'occurrence où les lignes représentent les mots et les colonnes les documents. Chaque cellule de cette matrice contient le nombre d'occurrences des mots dans les documents. Après la décomposition et la réduction de la matrice originale, cette méthode produit des vecteurs qui représentent les mots du corpus. Deux mots sémantiquement proches sont représentés par des vecteurs proches. La mesure de proximité est définie, dans ce cas, par le cosinus de l'angle entre les deux vecteurs.

Christian et al [RC08] proposent une autre méthode qui s'appelle Topic. Les auteurs ont considéré que chaque sujet (topic) est représenté par un groupe de mots-clés. Ils commencent par créer une matrice de corrélations pour définir une mesure des distances entres les mots. Ensuite, les deux éléments qui ont la plus grande distance sont utilisés comme des centres de deux classes. L'assignation des éléments aux centres est basée sur le principe de la classification défini dans k-means, cet algorithme est expliqué en détails dans [FE06]. on reprend le même processus d'éclatement à chacune des deux classes. L'éclatement dépend d'une valeur de seuil spécifiée. L'algorithme trouve donc un arbre binaire des classes et chaque classe représente un topic.

Dans [FO08] les auteurs proposent une nouvelle fonction d'agrégation des mots clefs qui s'appelle Top-Keyword. Cette fonction consiste à extraire les K mots les plus représentatifs d'un corpus de documents. Pour cela, il est nécessaire de calculer la représentativité de chaque terme du corpus. A partir des travaux de recherche dans ce domaine, [FO08] ont utilisé la mesure tf.idf pour l'évaluation de la représentativité d'un terme. Cette mesure est évaluée par le produit entre la représentativité d'un terme dans un document (tf : term frequency) avec l'inverse de sa représentativité dans l'ensemble des autres documents de corpus (idf : inverse document frequency). L'inconvénient majeur de cette approche et que la mesure tf.idf extrait seulement les mots simples. Par exemple : le verbe "fait" n'a pas le même sens avec le mot "fait" comme dans " la table de fait".

Dans [SA10] les auteurs ont définit deux nouvelles fonctions d'agrégation. La première méthode s'appuie sur une hiérarchie adaptée aux données textuelles, le but est de sélectionner les mots clefs pertinents en utilisant une nouvelle mesure de tf.idf adaptative qui tient en compte des hiérarchies associées aux dimensions, dans ce cas les représentativités des termes sont calculées par rapport au nombre de documents dans un niveau de granularité souhaité par l'utilisateur ensuite, conserver les n mots ayant les poids les plus élevés.

La seconde fonction est une agrégation dynamique permettant d'effectuer des groupements sur les mots clefs, le résultat d'agrégation obtenu peut être différent du regroupement fondé sur une hiérarchie statique. Les méthodes de clustering utilisées (k-means dans [FE06]) s'appuient sur des mesures classiques : cosinus, distance euclidienne (représentation fréquentielle) pour calculer les proximités entre les termes. Ensuite, sélectionner les mots ayant le poids le plus élevé, on peut accorder des poids fréquentiels ou de type tf.idf.

3 Approches basées sur l'utilisation des connaissances externes

Ces approches consistent à sélectionner certains mots-clefs qui représentent un domaine. Elles utilisent souvent des ressources supplémentaires afin d'effectuer des tâches spécifiques sur des données textuelles telles que l'agrégation des résultats obtenus ou la classification des documents. Dans la littérature, on distingue deux types de modèles de connaissance utilisés comme ressources externes : les thésaurus et les ontologies.

3.1 Ontologie

[Far09], définie le rôle principal de l'ontologie est d'avoir un consensus sur le vocabulaire à utiliser pour un domaine donné. En effet, une ontologie c'est une conceptualisation dans le sens où elle fournit un vocabulaire formalisé de concepts et de leurs relations Dans le domaine d'extraction d'information les auteurs utilisent des ontologies légères, elles se limitent à la définition des concepts et des relations entre les concepts.

Dans [Ron07], les auteurs proposent la fonction d'agrégation AVG-KW, elle s'intéresse aux mesures textuelles de type mots-clefs. Elle prend en entrée un ensemble de mots-clefs d'un corpus de document et donne en sortie un autre ensemble de mots-clefs agrégés et plus générale. Pour établir le processus d'agrégation, la fonction AVG-KW utilise une ontologie légère qui correspond à un ensemble de termes sous forme arborescente où les noeuds sont les termes et les arcs représentent une relation «est-un» Lors de l'agrégation, l'ontologie de domaine consiste à trouver pour chaque paire de noeuds (termes) :

- le plus petit ancêtre commun (lca) correspondant.

- la distance d qui est le nombre d'arcs entre le noeud représentant le plus petit ancêtre commun (lca) et le noeud le plus bas dans la hiérarchie entre ces deux noeuds.

Lorsque les termes sont très éloignés, dans l'ontologie il est possible de retourner le noeud racine comme un lca, et le résultat de l'agrégation s'accompagne par une perte de sens. Pour éviter ce problème, la fonction utilise une distance maximale (Dmax) autorisée lors de l'agrégation des termes (Cette distance est généralement comprise entre 3 et 5). L'inconvénient majeur de cette approche est que lors de l'agrégation de mots clef en des mots clef plus généraux, une perte de précision, en terme de sémantique, est à prévoir. Afin de limiter la perte de sens liée à l'agrégation, il est donc nécessaire de suivre l'évolution de cette agrégation [Ron07].

3.2 Les thésaurus

[Far09] définie «le thésaurus comme étant un vocabulaire contrôlé, il rassemble un ensemble de termes structurés choisis pour leur capacité à décrire un domaine ». Le thésaurus regroupe dans une classe les termes d'un domaine particulier. Ces termes sont reliés entre eux par des relations sémantiques : liens hiérarchiques (généralisation et spécialisation), synonymie et définition...

Dans cet état de l'art nous avons résumé les principaux travaux réalisées sur les méthodes d'extraction des mots clefs pour agréger les données contenues dans les documents textuels. Nous avons choisi deux fonctions qui sont parmi les travaux les plus connus dans le domaine d'analyse des données textuelles : Top-Keyword [FO08] et Topic [RC08].

Dans le chapitre suivant nous allons expliquer en détail leurs principes pour bien comprendre leurs fonctionnements, ensuite nous allons les implémentées dans le but de comparer leurs résultats.

Chapitre 3

Les fonctions d'agrégation Top-Keyword et Topic

Dans ce chapitre nous allons présenter les fonctions d'agrégation des données textuelles que nous avons choisies pour notre étude, la fonction Top-Keyword et la fonction Topic qui est basée sur la classification. Ces fonctions font l'objet de nombreux travaux de recherche. Les données agrégées apparaissent sous forme de mots clés les plus représentatifs d'un ensemble de documents. Ces fonctions sont faciles à comprendre et simples à implémenter.

1 La fonction d'agrégation Top-Keyword

1.1 principe de la fonction

La fonction d'agrégation Top-Keyword utilise une matrice de fréquences où les colonnes sont les mots et les lignes sont les documents, cette matrice permet d'attribuer à chaque terme j sa fréquence d'apparition dans le document i. Elle commence par le calcul du poids de chaque terme dans le corpus de documents en utilisant la mesure tf.idf, ensuite on va ordonner les termes en fonction de leur poids tf.idf. A la fin, cette fonction extrait les K premier termes et les considérés comme le résultat d'agrégation obtenus par la fonction Top-Keyword [FO08].

1.2 Calcul du poids des termes « la mesure tf.idf »

Pour calculer les poids des termes nous utilisons deux formules de tf.idf :

– **tf.idf complexe :**

Selon [Ron07] la mesure tf.idf est utilisée pour évaluer la représentativité d'un terme. Cette mesure est opérée sur la matrice de fréquences. Le tf.idf est le produit entre la représentativité d'un terme dans un document (tf : term frequency) avec l'inverse de sa représentativité dans l'ensemble des autres documents de corpus (idf : inverse document frequency). elle est évalué comme suit :

$$tf.idf(t) = \frac{n_{ij}(t)}{n_{ij}} \times \log \frac{d_{ij}+1}{d_{ij}(t)}$$

- tf(t) est le nombre de fois où le terme t est présent dans un document $n_{ij}(t)$ par rapport au nombre total de termes contenus dans ce document n_{ij}
- idf(t) est l'inverse du nombre de documents contenant le terme t $d_{ij}(t)$ par rapport au nombre de documents totales (d_{ij}).
  La fonction logarithme est utilisée pour réduire les conséquences de grandes valeurs. En ajoute 1 au nombre totale des documents afin d'éviter d'obtenir une représentativité égale à 0 dans le cas, ou le terme t est contenu dans tous les documents et on trouve $d_{ij} = d_{ij}(t)$.

– **tf.idf simple :**

On peut calculer la valeur de tf.idf pour chaque terme par la formule standard :

$$tf.idf(t) = \frac{n_i(t)}{n}$$

- $n_i(t)$ est la fréquence totale de terme t dans corpus.
- n est la fréquence totale de tous les termes dans le corpus.

Pour bien expliquer le fonctionnement de cette fonction nous utilisons des exemples d'application, indiquent comment calculer les poids des termes avec les deux formules en détail.

Exemple 1 : tf.idf complexe

Dans la Figure 3.1 , nous avons appliqué la mesure tf.idf complexe sur une matrice qui contient trois documents et six termes pour calculer le poids de chaque terme on suit les étapes suivantes :

1. Calculer le nombre de termes total dans chaque document.
2. Calculer le TF de chaque terme pour chaque document.
3. Calculer la somme de TF de chaque terme par rapport à tous les documents.
4. Pour chaque terme calculer le nombre de documents contenant ce terme.
5. Calculer l'IDF de chaque terme.
6. Calculer le produit entre le TF et IDF de chaque terme.

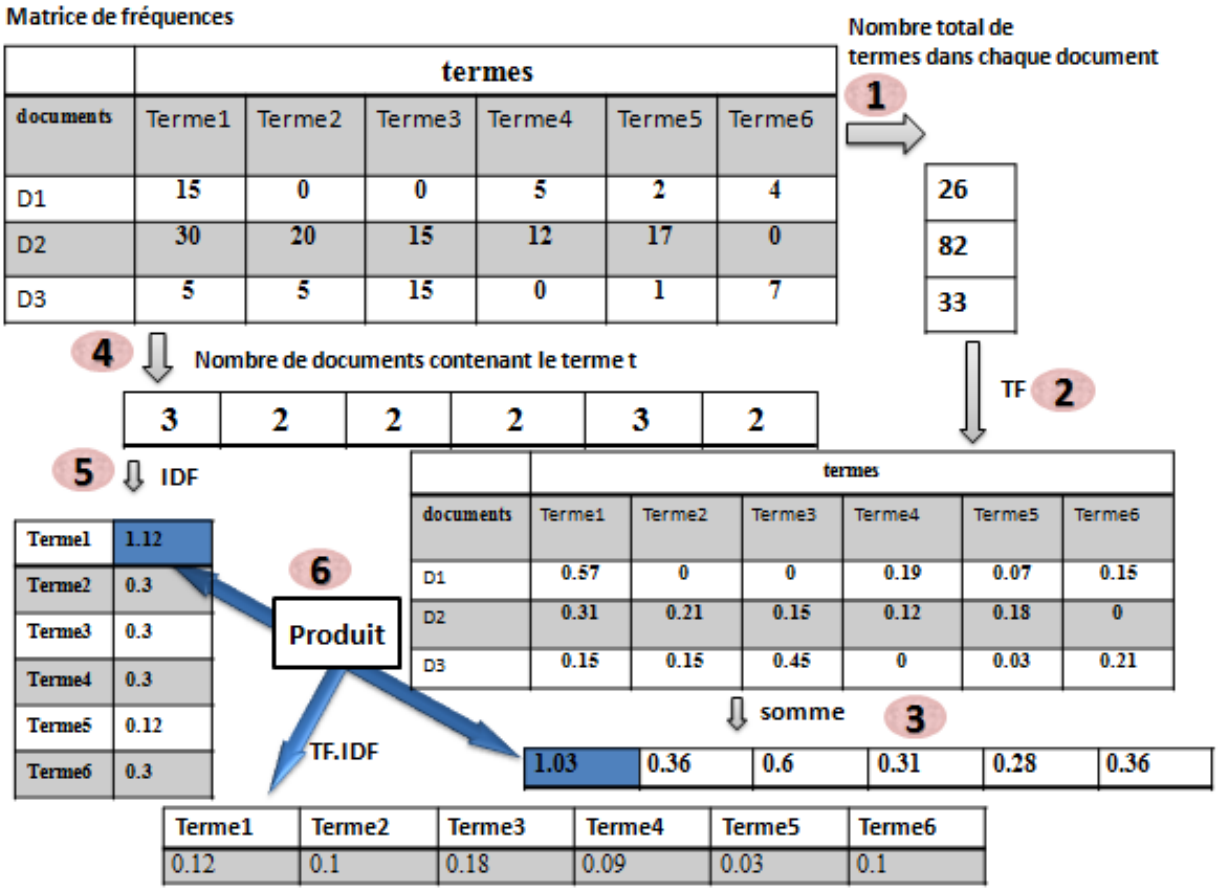

FIGURE 3.1 – Exemple d'application de la fonction Top-keyword f.idf complexe

Dans ce cas, après avoir trié les termes dans l'ordre décroissant de leurs tf.idf et en prenant K=4 alors les 4 Top-keyword sont : terme3, terme1, terme2 et terme6.

Exemple 2 : tf.idf simple

Dans la Figure 3.2 , nous avons appliqué la mesure tf.idf simple sur la même matrice pour calculer le poids de chaque terme. Les étapes à suivre sont :

1. Calculer la somme des fréquences totales des termes.
2. Calculer la somme des fréquences de chaque terme dans le corpus.
3. Calculer le tf.idf de chaque terme par la division de la somme des fréquences de terme dans le corpus par la somme des fréquences totales des termes.

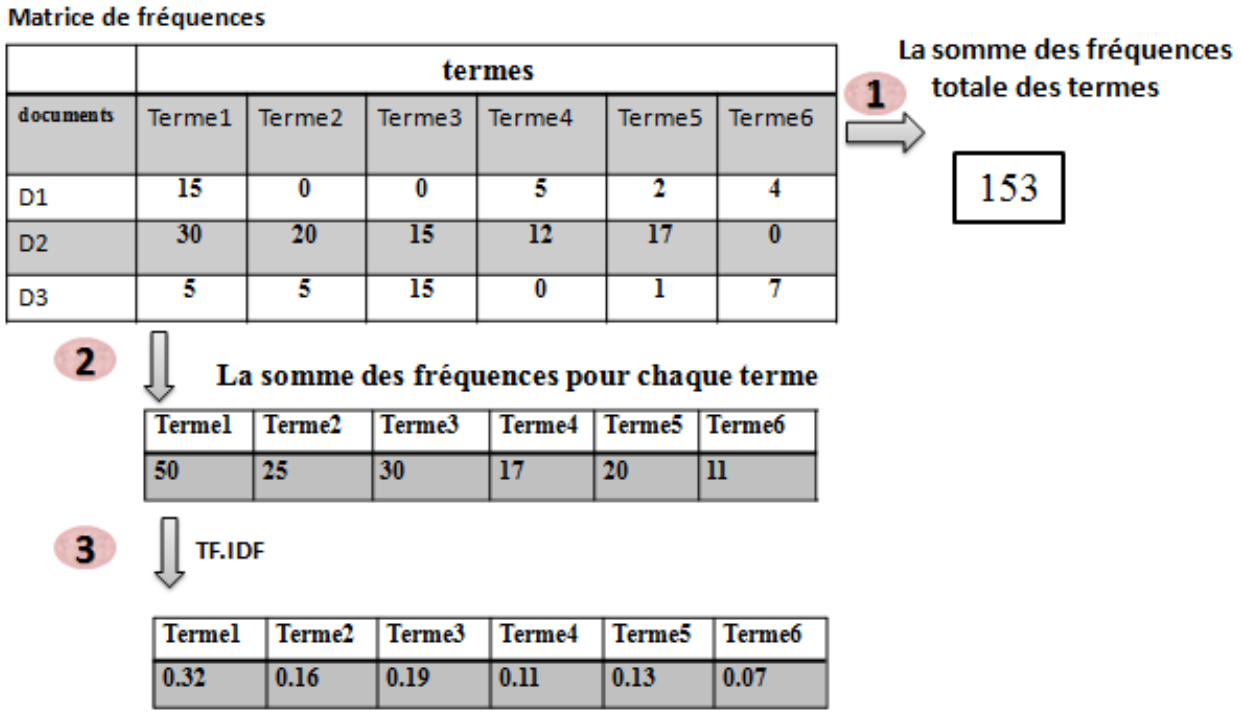

FIGURE 3.2 – Exemple d'application de la fonction Top-keyword tf.idf simple

On ordonne les termes dans l'ordre décroissant de leur poids, puis on extrait les K Top-Keyword. Si k=4 alors les 4 Top-Keyword sont : terme1, terme3, terme2 et terme5.

2 La fonction d'agrégation Topic

2.1 principe de la fonction

La fonction Topic connue par Topic Detection by Clustering Keywords les auteurs ont considéré que chaque sujet (topic) est représenté par un groupe de mots clés. Pour effectuer une agrégation à l'aide de cette fonction, ils ont commencé par la création de la matrice de distances entre les mots pour regrouper les mots similaires dans la même classe.
Une manière efficace pour définir la similarité entre deux mots consiste à calculer une distance d (t,s) entre les termes t, s par les étapes suivantes :

- Entre chaque deux termes (t , s) la mesure de similarité est évaluée par la formule de cosine similarity :

$$\cos sim\,(t,s) = \frac{\sum_{d\in C} Q_t(d)Q_s(d)}{\sqrt{(\sum_{d\in C} Q_t(d)^2)(\sum_{d\in C} Q_s(d)^2)}}$$

Où
$Q_t(d) = n(d,t)/n(t)$.
$n(d,t)$: *le nombre d'occurrence de terme t dans le document d.*
$n(t)$: *le nombre d'occurrence de terme t dans le corpus.*
C : *corpusdedocuments.*

- La fonction de distance = 1- $\cos sim(t,s)$.

On sélectionne deux termes qui ont la plus grande distance (les termes les plus éloignés) à partir de la matrice des distances, pour être utilisés comme des centres pour deux classes initiaux. Ensuite on applique l'algorithme k-means, pour assigner tous les éléments restants à l'un des deux termes centres. Une fois que tous les éléments ont été affectés à une classe, on va recalculer les nouveaux centres jusqu'à ce que les deux centres soient en convergence. Pour les deux groupes trouvés, on va les éclater de la même manière, c'est-à-dire la procédure initiale est répétée pour chaque classe jusqu'à ce que le nombre de classes atteigne un seuil spécifié. Ensuite, pour chaque classe obtenue on doit sélectionner un leader celui ayant la valeur maximale de tf.idf. Les leaders obtenus représentent les agrégats de corpus obtenus par la fonction Topic.

Exemple 3 :

Dans la Figure 3.3 , la fonction Topic prend en entre une matrice de fréquences et passe par quatre étapes pour donner en sortie un ensemble de classes.

(a) Calculer la distance entre chaque paire de termes.

(b) Extraire les deux termes ayants la distance maximale pour initialiser les centres des classes.

(c) En prenant une valeur de seuil (doit être définit par l'utilisateur) égale à quatre.

(d) Eclater chaque classe en deux pour obtenir à la fin quatre classes.

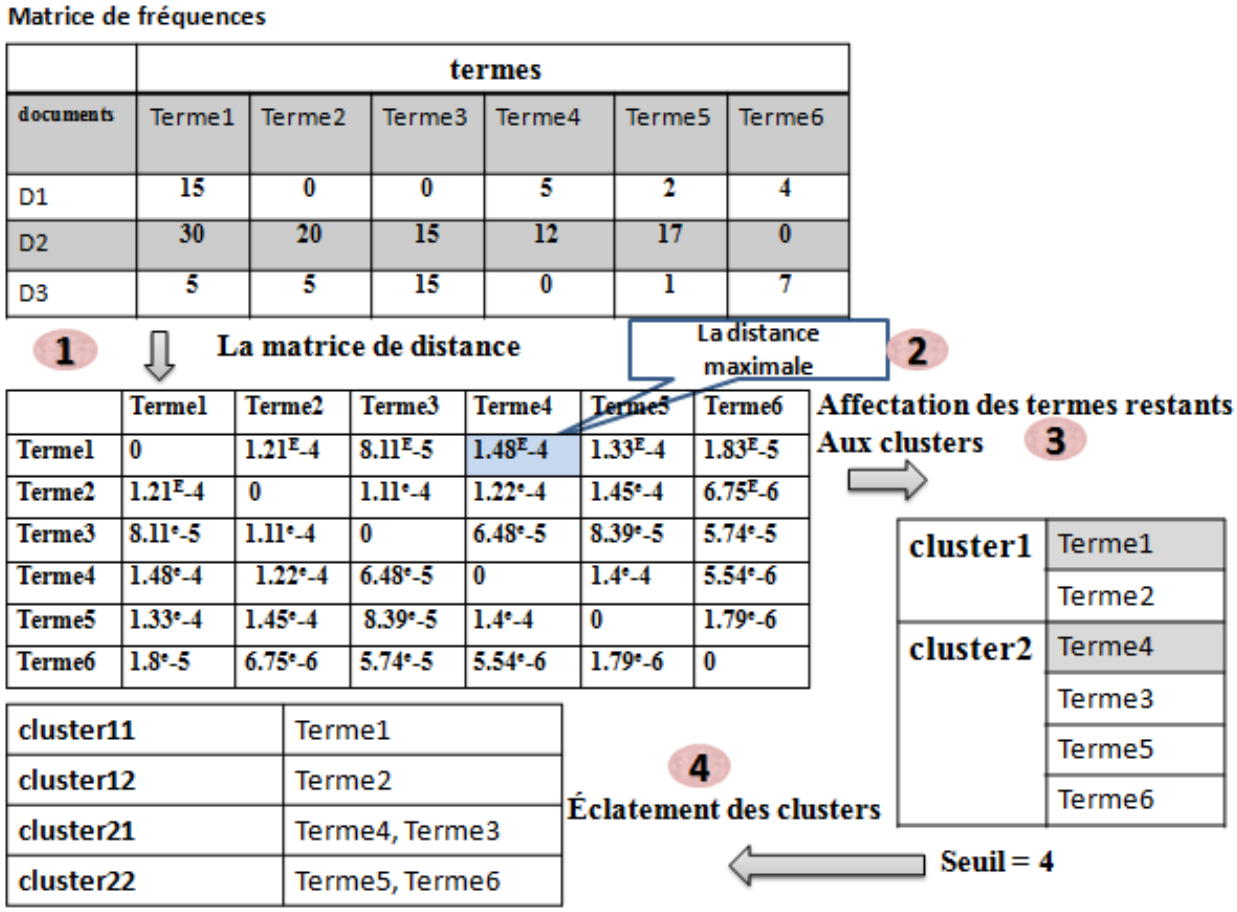

FIGURE 3.3 – Exemple d'application de la fonction Topic.

Pour chaque classe obtenue, on doit extraire le terme ayant la valeur maximale de tf.idf (simple et complexe) qui a déjà été calculé dans les exemples précédents, on trouve quatre termes qui représentent le corpus.
- Avec tf.idf simple, les agrégats sont : terme1, terme2, terme3 et terme5.
- Avec tf.idf complexe, les agrégats sont : terme1, terme2, terme3 et terme6.

Dans ce chapitre nous avons expliqué en détail les fonctions choisies pour bien comprendre leurs principes et leurs fonctionnements, dans le chapitre suivant on va appliquer ces fonctions sur un corpus réel pour analyser les résultats obtenus, mais avant de passer à l'application, nous expliquons d'abord les étapes d'implémentation de ces fonctions.

Chapitre 4

Implémentation et Expérimentations

L'objectif de ce chapitre est de présenter les différentes expérimentations effectuées pour évaluer les résultats des deux fonctions expliquées précédemment. En commençant par citer les étapes d'implémentation.

1 Implémentation des fonctions Top-keyword et Topic

1.1 Environnement de travail

Le programme de ce travail a été écrit en langage java, ce langage intègre les concepts les plus intéressants des technologies informatiques récentes dans une plate-forme de développement riche et homogène. L'approche objet de ce langage, sa portabilité et sa gratuité, le placent parmi les outils de programmation les plus efficaces. Depuis la version1.4.2, Java dispose d'outils modernes d'installation et de mise à jour. Il est maintenant possible de télécharger le JDK (Java Development Kit) ou le JRE (Java Runtime Environment) sur internet.

Nous avons installé la version JDK1.6.0-26 dans un ordinateur ayant un processeur Intel® CORE ™ I5, une RAM de 4GO et disque dur de 500 GO avec le système d'exploitation windows 7 professionnel service pack 1 de type 32 bits.

Le programme est écrit dans l'éditeur de code NetBeans IDE 7.0, C'est un éditeur parmi les IDE (Integrated Development Environment) Java. Il simplifie grandement l'édition et la gestion d'un programme. Ils intègrent les fonctionnalités suivantes [Emm08] :

- Editeur de textes avec mise en couleur des mots clés Java, des commentaires ...
- Complétion automatique (menus contextuels proposant la liste des méthodes d'un objet).
- Génération automatique des dossiers nécessaires à l'organisation d'un programme et des paquetages des classes.
- Intégration des commandes Java et de leurs options dans des menus et des boîtes de dialogue appropriés.
- Débogueur pour corriger les erreurs.

1.2 Algorithmes et explications

La classe Top-KW : c'est la classe main, elle contient le programme principale, il commence par la lecture des données à partir du fichier texte pour créer la matrice des fréquences. Ensuite il fait appel aux fonctions définies dans les classes TF-IDF et k-means pour voir les résultats des fonctions d'agrégation Top-Keyword et Topic. Les étapes d'implémentation de ces fonctions sont montrées dans la figure 4.1.

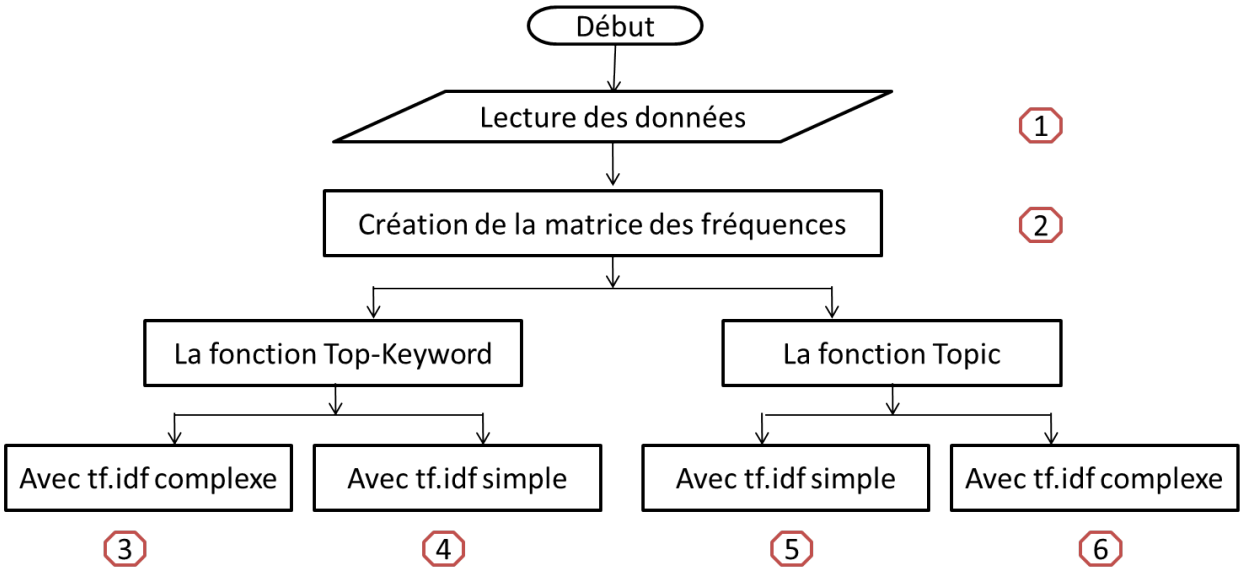

FIGURE 4.1 – Le programme principale.

– **(1) et (2) :** ces étapes sont très importantes, le programme lit les données contenues dans le fichier texte pour créer la matrice.
La matrice des fréquences obtenue est utilisée pour appliquer les fonctions Top-Keyword et Topic.
– **(3) et (4) :** La classe TF-IDF contient les fonctions : *existe, TF, somme, IDF, tfidf, max, nbtt, topkw.*
Les fonctions : *nbtt et topkw* sont appelées pour retourner les top keywords par la formule tf.idf simple (voir la figure 4.2), et les autres sont appelées pour retourner les top Keywords par la formule tf.idf complexe (voir la figure 4.3).

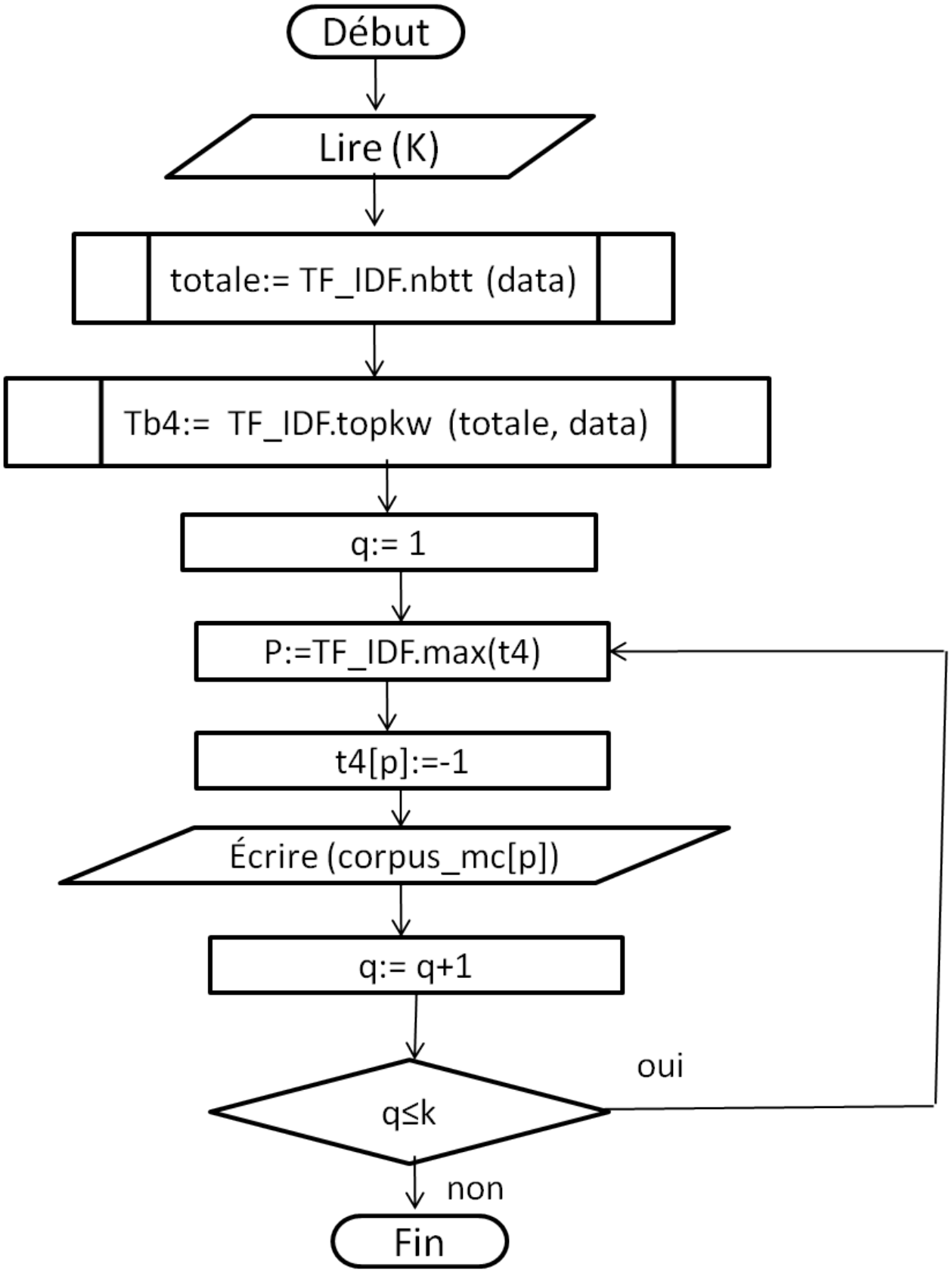

FIGURE 4.2 – la fonction Top-keyword avec tf.idf simple

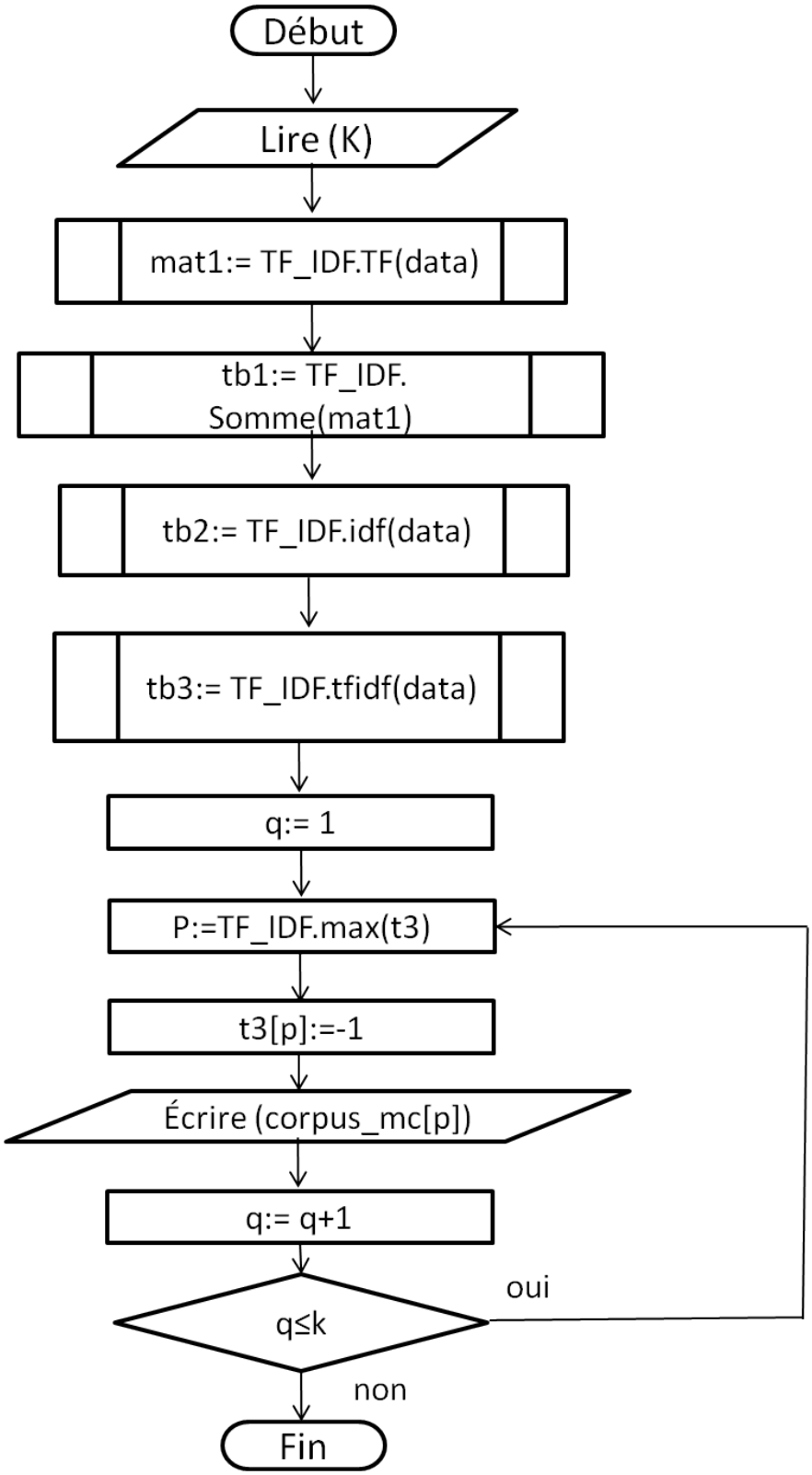

FIGURE 4.3 – la fonction Top-keyword avec tf.idf complexe

– **(5) et (6) :** La classe k-means contient les fonctions : *max, dist-topic, exist*, qui sont appelées pendant l'exécution de la fonction Topic. Après la création des classes, nous utilisons la mesure tf.idf simple et tf.idf complexe pour sélectionner les leaders. Les étapes sont dans la figure 4.4.

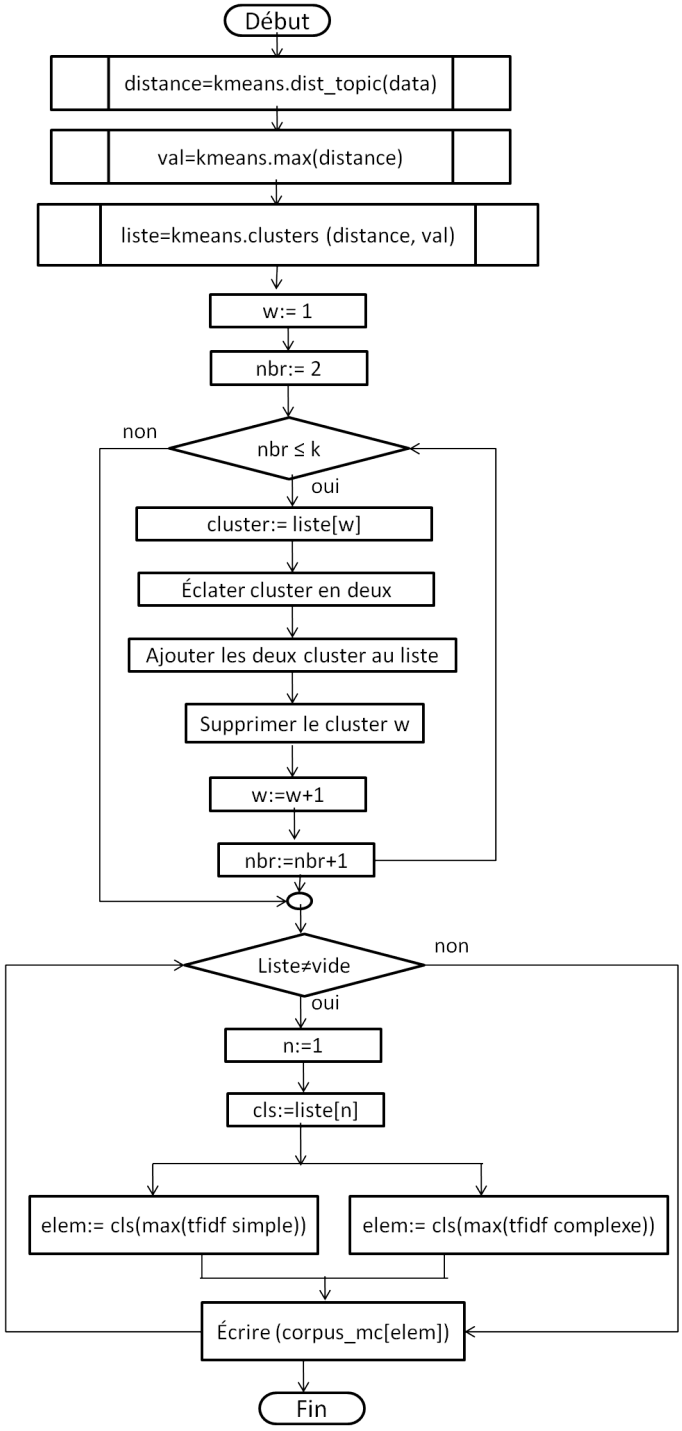

FIGURE 4.4 – la classe K-means pour la fonction Topic.

Nous avons expliqué en détail la phase d'implémentation, nous allons passer à l'exécution de notre programme, mais avant celà il faut décrire le corpus que nous avons utilisé dans nos expérimentations.

2 Le corpus de l'expérimentation

Afin de comparer efficacement les résultats obtenus, Nous avons testé les fonctions sur un ensemble de données textuelles correspondentes à des articles publiés dans la conférence «Innovation 2011» qui s'est tenue à Dubaï. Ces articles exposent les innovations et les recherches sur les technologies de l'information. Dans cette conférence tous les documents sont écrits en langue anglaise et traitent les sujets suivants : «Computing and Embedded systems», «Intelligent and software systems», «Communication and networking», «Computer and information Security», «Internet applications and web services», «Learning and education technologies».

Pour effectuer l'agrégation par les fonctions choisies, nous avons utilisé les 81 articles de cette conférence comme étant un corpus d'expérimentation. Initialement ce corpus contient en totale 11185 mots avec une somme de fréquences 279097 et afin d'identifier les termes qui représentent le corpus, il faut effectuer un prétraitement sur l'ensemble des termes. Le prétraitement consiste à appliquer les actions suivantes :

- Segmenter le texte en phrases puis en mots.
- Transformer les majuscules en minuscules.
- Filtrer le corpus en éliminant les mots vides de sens (prépositions, conjonctions, adverbes, chiffres, ponctuations, etc.), ils sont des mots non significatifs
- Ramener chaque mot à une forme invariable qui peut être sa racine ou sa forme canonique (singulier ou l'infinitif pour les verbes).
- Prendre les mots qui ont un pourcentage d'apparition dans le corpus de plus de 30% (un seuil fixé par les experts du domaine) pour travailler seulement avec les mots les plus importants.

Après le prétraitement le nombre total des mots est réduit à 110 termes.
Avant de lancer l'exécution de notre programme, il faut structurer le corpus prétraité dans un fichier texte. La première ligne dans ce fichier contient le nombre de documents et le nombre total des termes, ensuite les termes qui représentent le corpus avec leurs fréquences d'apparition, ainsi que les termes et leurs fréquences pour chaque document séparés par le caractère « # ».
Description de fichier texte :

81 110
factor 144
result 145
line 147
large 148
main 149
distribution 150
.
.
.
+
acceptable 3

accomplish 1
accurately 1
achieved 1
.
.
.
#
accepted 1
access 6
accessed 6
according 3
accordingly 1
accuracy 1
#
.
.
.
#

3 Les résultats

Nous avons lancé l'exécution de notre programme afin de voir les résultats obtenus par les différentes fonctions.

3.1 La fonction Top-Keyword

Pour cette fonction le nombre des termes obtenus dépendent de la valeur de K saisie par l'utilisateur.
Comme nous avons expliqué précédemment, la fonction Top-keyword utilise la mesure tf.idf pour évaluer la représentativité de chaque mot. Nous avons utilisé deux formules pour l'évaluation des poids des termes.
- Dans le premier cas, les poids des termes sont calculés par la mesure tf.idf simple (avec la formule standard), nous avons choisi les dix premiers termes et ils sont triés et résumés dans la table 4.1

- Pour le deuxième cas, les poids de dix premiers termes obtenus par tf.idf complexe sont introduits dans la table 4.2

Les termes	Les tf-idfs simple
System	0.0436
network	0.02743
data	0.02641
service	0.02453
model	0.02346
information	0.02289
method	0.02053
number	0.02035
web	0.01871
time	0.01868

TABLE 4.1 – Les poids de dix premiers termes obtenus par la mesure tf.idf simple avec la fonction Top-Keyword.

Les termes	Les tf-idfs complexe
Image	0.69
packet	0.68
web	0.53
cell	0.45
agent	0.4
service	0.4
channel	0.39
vector	0.38
Bit	0.35
network	0.33

TABLE 4.2 – Les poids de dix premiers termes obtenus par la mesure tf.idf complexe avec la fonction Top-Keyword.

Dans les deux cas, après le trie descendant des termes en fonction de leurs poids, le programme récupère les K premiers mots qui veut représenter le résultat de l'agrégation effectuée par la fonction Top-Keyword.

3.2 La fonction Topic

L'objectif de cette fonction est de générer à partir des données textuelles rassemblées dans un corpus les classes de mots clés reflétant ses principaux thèmes. Ensuite pour chaque classe, on va sélectionner un leader grâce à la mesure tf.idf.

Pour effectuer l'agrégation avec la fonction Topic, le programme commence à calculer les distances entre les mots selon les formules expliquées précédemment afin de créer la matrice de corrélations entre les termes. Puis, il va trouver la plus grande distance entre les termes : *recommendation* et *item*.
L'algorithme k-means utilise ces deux termes comme des centres initiaux pour créer les classes. La table 4.3 présente les classes initiales crées par k-means.

Classe (1)	Classe (2)
recommendation ; factor ; line large ; main ; space ; bandwidth ; experiment ; present ; decision ; structure ; architecture ; tool ; example ; request ; cluster ; evaluation ; sensor ; security ; research ; layer ; computer ; language ; wireless ; object ; message ; approach ; video ; signal ; mobile ; test ; bit ; location ; communication ; word ; application ; work ; value ; user ; number ; model ; service ; data.	item ; result ; distribution ; software ; study ; accuracy ; source ; threshold ; region ; management ; capacity ; event ; knowledge ; simulation ; recognition ; sequence ; design ; position ; graph ; distance ; component ; database ; class ; access ; semantic ; agent ; cloud ; scheme ; transaction ; query ; generation ; cell ; detection ; environment ; distributed ; analysis ; vector ; function ; size ; protocol ; error ; order ; process ; type ; problem ; case ; performance ; table ; control ; context ; channel ; search ; power ; feature ; technology ; level ; technique ; packet ; provide ; set ; image ; web ; time ; method ; information ; network ; system.

TABLE 4.3 – Les classes initiales construites par l'algorithme de classification k-means et obtenues après l'exécution de la fonction Topic.

Ces deux classes permettent d'effectuer une première détection de thèmes sur le corpus. Cette procédure est répétée tant que le nombre des classes est inférieur à k, A la fin de l'exécution de cette fonction on va obtenir k classes.

Afin de sélectionner le leader de chaque classe nous avons utilisé la mesure tf.idf simple et le tf.idf complexe pour choisir le terme le plus représentatif, c'est-à-dire le terme qui a le poids maximum dans la classe. Pour les deux cas, les résultats d'agrégation obtenus par la fonction Topic sont montrés en détail dans la table 4.4 , en prenant par exemple K=10.

4 Comparaison et interprétation des résultats

Pour effectuer la comparaison entre les différents résultats des fonctions nous avons pris les valeurs de k entre 2 et 10, les agrégats obtenus par chaque fonction sont dans La table 4.5.

Les classes	Le nombre des termes de classe	Leader sélectionné par tf .idf simple	Le tf.idf simple	Leader sélectionné par tf .idf complexe	Le tf.idf complexe
Classe (1)	1	approach	0.01008	approach	0.04
Classe (2)	18	Data	0.02641	service	0.4
Classe (3)	13	System	0.0436	vector	0.38
Classe (4)	20	Information	0.02289	graph	0.23
Classe (5)	9	Web	0.01871	image	0.69
Classe (6)	25	Level	0.01179	cell	0.45
Classe (7)	14	Word	0.01113	recommendation	0.32
Classe (8)	1	Application	0.01185	application	0.03
Classe (9)	1	research	0.00683	Research	0.03
Classe (10)	8	Model	0.02346	Model	0.26

TABLE 4.4 – Les classes finales avec leurs représentants obtenus par la mesure tf .idf simple et tf.idf complexe avec la fonction Topic.

Afin d'évaluer la performance de chaque fonction sur notre corpus, nous avons comparé les résultats résumés dans le tableau 4.5 avec les thèmes définis dans la conférence. La collection des documents dans cette conférence traite 6 sujets, pour celà nous avons choisi le cas où k=6 pour effectuer la comparaison. En utilisant l'arbre dans la figure 4.5, nous avons essayé de découvrir les thèmes détectés par chaque fonction.

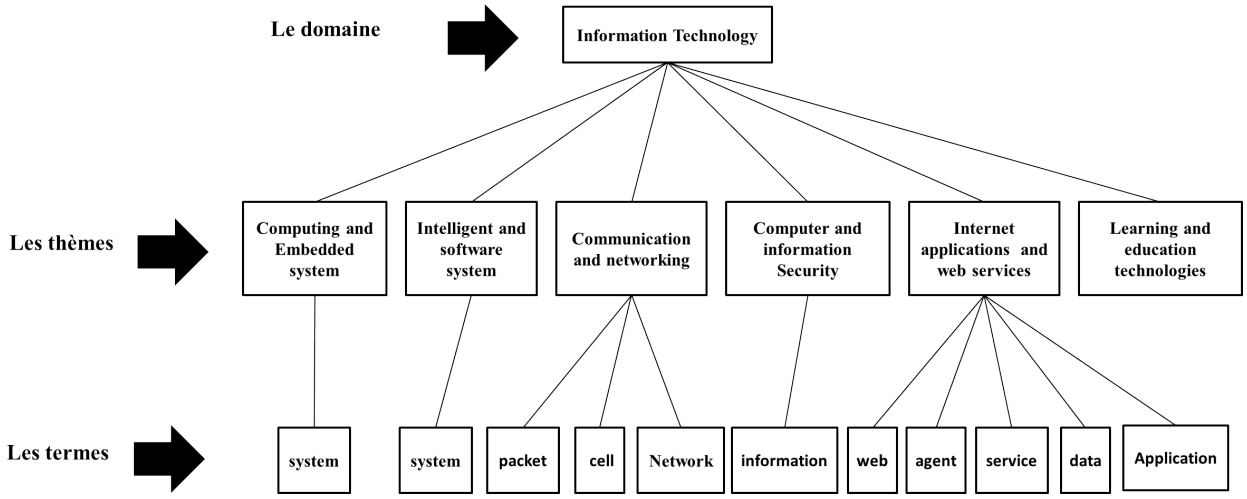

FIGURE 4.5 – représentation des termes du corpus par rapport aux thème de la conférance

Nous obtenons la table 4.6 qui indique pour chaque fonction quels sont les thèmes qui correspondants aux résultats.

	Top-Keyword avec tf.idf complexe	**Top-Keyword avec tf.idf simple**	**Topic avec tf.idf complexe**	**Topic avec tf.idf simple**
K=2	Image, packet	System, network	Service, image	Data, system
K=3	Image, packet, web	System, network, data	Image, recommendation, service	System, model, data
K=4	Image, packet, web, cell	System, network, data, service	Recommendation, service, vector, image	Model, data, system, web
K=5	Image, packet, web, cell, agent	System, network, data, service, model	Service, vector, image, recommendation, model	Data, system, web, application, model
K=6	Image, packet, web, cell, agent, service	System, network, data, service, model, information	Vector, image, recommendation, model, application , service	System, web, application, model, approach, data
K=7	Image, packet, web, cell, agent, service, channel	System, network, data, service, model, information, method	Image, recommendation, model, approach, service, vector, graph	Web, application, model, approach, data, system, information
K=8	Image, packet, web, cell, agent, service, channel, vector	System, network, data, service, model, information, method, number	Recommendation, model, approach, service, vector, graph, image, cell	Application, model, approach, data, system, information, web, level
K=9	Image, packet, web, cell, agent, service, channel, vector, bit	System, network, data, service, model, information, method, number, web	Model, approach, service, vector, graph, image, cell, recommendation, application	Model, approach, data, system, information, web, level, word, application
K=10	Image, packet, web, cell, agent, service, channel, vector, bit, network	System, network, data, service, model, information, method, number, web, time	Approach, service, vector, graph, image, cell, recommendation, application, research, model	Approach, data, system, information, web, level, word, application, research, model

TABLE 4.5 – les résultats des fonctions d'agrégation Top-Keyword et Topic en variant le k entre 2 et 10.

Les fonctions d'agrégation	**Les termes**	**Les thémes détectés**	**Nombre totale des thèmes dètectés**
Top-Keyword avec tf.idf complexe	Image	/	2 thèmes
	Packet , cell	Communication and Networking	
	web, service , agent	Internet applications and web services	
Top-Keyword avec tf.idf simple	System	computing and embedded systems , intelligent and software systems	5 thèmes
	network	Communication and Networking	
	service , data	Internet applications and web services	
	model	/	
	information	computer and information security	
Topic avec tf.idf complexe	vector , image , recommendation , model	/	un thème
	application , service	Internet applications and web services	
Topic avec tf.idf simple	system	computing and embedded systems , intelligent and software systems	3 thèmes
	web , data , application	Internet applications and web services	
	approche , model	/	

TABLE 4.6 – les résultats des fonctions d'agrégation Top-Keyword et Topic en variant le k entre 2 et 10.

D'après la table 4.6, nous avons trouvé que les termes obtenus par la fonction Top-Keyword qui utilise le tf.idf simple représentent le plus grand nombre des thèmes du corpus (5 thèmes), donc cette fonction donne des meilleurs résultats par rapport aux autres. Ensuite nous avons trouvé la fonction Topic avec tf.idf simple (3 thèmes) et la fonction Top-Keyword avec tf.idf complexe (2 thèmes). Et les résultats de ces derniers sont meilleurs que les résultats de la fonction Topic qui utilise tf.idf complexe (un seul thème).

Donc on peut déduire que : la fonction Top-Keyword avec tf.idf simple donne des meilleurs résultats (5/6) et ce qui concerne la fonction Topic les résultats sont acceptables (3/6).

Après avoir comparé les termes obtenus par les fonctions que nous avons choisies, on peut conclure que la fonction Top-Keyword lorsqu'elle utilise la mesure tf.idf simple réalise une meilleure agrégation des données textuelles.

Après l'analyse des résultats des fonctions pour trouver les meilleures agrégations, on s'interesse aussi à etudier la similarité de ces résultats. Pour chaque valeur de k (entre 2 et 10) nous avons calculé le pourcentage des termes en communs entre les fonctions après chaque étape d'agrégation. Nous avons obtenu la table 4.7.

	Top-Keyword avec tf.idf simple / Top-Keyword avec tf.idf complexe (%)	**Top-Keyword avec tf.idf simple / Topic avec tf.idf complexe (%)**	**Top-Keyword avec tf.idf simple / Topic avec tf.idf simple (%)**	**Top-Keyword avec tf.idf complexe / Topic avec tf.idf simple (%)**	**Top-Keyword avec tf.idf complexe / Topic avec tf.idf complexe (%)**	**Topic avec tf.idf simple / Topic avec tf.idf complexe (%)**
K=2	0	0	50	0	50	0
K=3	0	0	66	0	33	0
K=4	0	25	50	25	25	0
K=5	0	40	60	20	20	20
K=6	16	33	50	16	33	16
K=7	14	28	57	14	28	28
K=8	12	25	50	12	50	25
K=9	22	22	55	11	44	33
K=10	30	20	50	10	40	40

TABLE 4.7 – Les pourcentages des termes en communs entre les fonctions selon les dix valeurs de k.

Nous avons utilisé la représentation graphique pour bien montrer la similarité des résultats entre les différentes fonctions (figure 4.6).

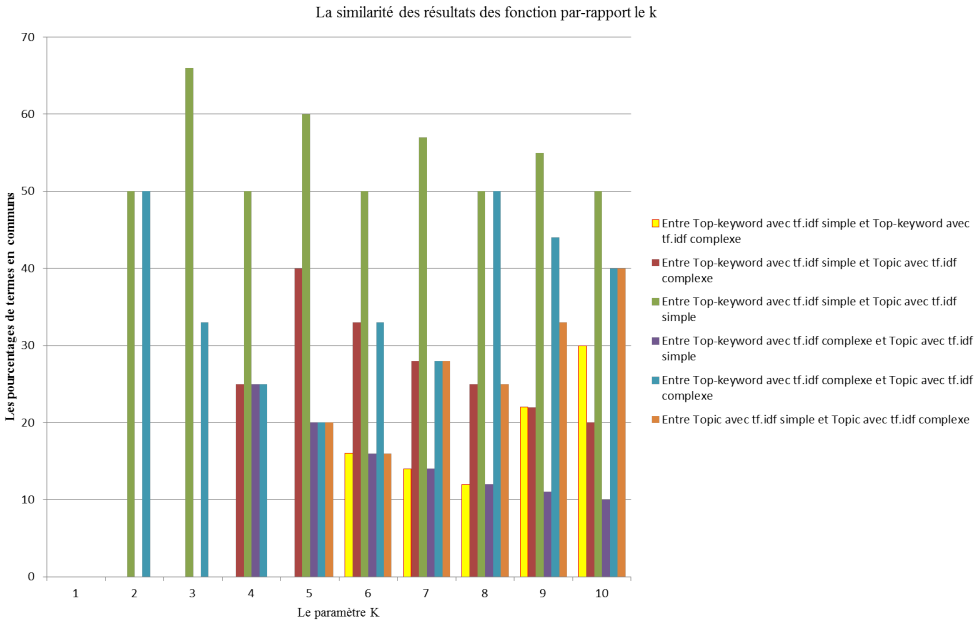

FIGURE 4.6 – La représentation graphique de la similarité des résultats des fonctions Top-Keyword et Topic.

A partir du tableau 4.7 et la figure 4.6 en peut remarquer que : pour n'importe quelle valeur de k, la courbe qui concerne les termes communs entre Top-Keyword avec tf.idf simple et Topic avec tf.idf simple est en dessus, dans ce cas en déduisant qu'il y a une forte similarité (pourcentage supérieur ou égale à 50%) entre ces deux fonctions et à certaine valeur de k les résultats de ces deux fonction sont presque les mêmes, mais toujours en prenant compte que la valeur de k ne doit pas être grande pour que l'agrégation garde son importance.
Les autres courbes déterminent une faible similarité (tous les pourcentages inférieur à 50%) entre les fonctions : Top-Keyword avec tf.idf simple et Topic avec tf.idf complexe, Top-Keyword avec tf.idf complexe et Topic avec tf.idf simple, Top-Keyword avec tf.idf simple et Top-Keyword avec tf.idf complexe , Topic avec tf.idf simple et Topic avec tf.idf complexe et entre Top-Keyword avec tf.idf complexe et Topic avec tf.idf complexe.

5 Remarques

La fonction Top-Keyword avec n'importe quelle formule de tf.idf utilisé, elle connait un inconvénient majeur qui peut influer sur son efficacité. Ce problème apparait lorsqu'on trouve un terme très fréquent dans un seul document même s'il n'apparait dans aucun autre document, son poids avec la mesure tf.idf va le classer parmi les mots les plus représentatifs dans le corpus complet. Ceci implique que ce terme est élu comme étant un Top-Keyword pourtant il représente un seul document dans le corpus.

Les résultats de la fonction Topic sont acceptables mais dans son fonctionnement, elle est basée sur la classification des termes. On peut trouver dans une seule classe plusieurs termes qui sont représentatifs et on va sélectionner un seul terme celui qui a le poids maximum comme étant un leader et négliger les autres. Par contre on peut trouver dans une autre classe des termes qui ne sont pas représentatifs ou qui contient un seul terme qui va aussi être sélectionné comme leader de cette classe. Il va, ainsi représenter tous le corpus et ceci influe sur l'efficacité de cette fonction.

Conclusion

Acause des problèmes d'analyse de la grande masse de données textuelles plusieurs fonctions ont été proposées pour agréger ce type de données, notre objectif est d'effectuer une étude comparative entre ces fonctions.

Dans le cadre de ce mémoire et pour faire notre étude comparative nous avons choisi les fonctions Top-Keyword et Topic en utilisant deux formules pour calculer les tf.idfs des termes, ces deux fonctions sont très simples et elles sont parmi les travaux les plus connues dans ce domaine, nous les avons implémentées à l'aide du langage java.

Afin d'évaluer la performance de chaque fonction nous avons testé notre programme sur un corpus réel constitué des articles de la conférence innovations 2011 qui s'est tenue à Dubaï. Après analyser les résultats nous sommes à la conclusion que la fonction Top-Keyword, utilisée avec la mesure tf.idf simple donne de meilleurs résultats. D'autre part les deux fonctions Top-Keyword et Topic, utilisée avec la mesure tf.idf simple produisent les plus de termes en commun.

Ce travail a été une expérience enrichissante, nous avons découvert plusieurs aspects pour une meilleure programmation avec java, en plus nous avons acquis de nouvelles connaissances dans le domaine de l'analyse des données textuelles.

Bibliographie

[Abd10] ADLA Abdelkader. *Aide à la Facilitation pour une prise de Décision Collective : Proposition dun Modèle et dun Outil.* Thèse de doctorat de l'Université Toulouse III Paul Sabatier, 2010.

[AD08] Ramzi Abbes and Joseph Dichy. *Extraction automatique de fréquences lexicales en arabe et analyse d'un corpus journalistique avec le logiciel AraConc et la base de connaissances DIINAR.1.* Presses Universitaires de Lyon, 2008.

[BA96] Habert B and Nazarenko A. *La syntaxe comme marche pied de l'acquisition des connaissances : Bilan critique d'une expérience.* Actes des septièmes Journées Acquisition des Connaissances, 1996.

[BEC09] N. BECHET. *Extraction et regroupement de descripteurs morpho-syntaxiques pour des processus de Fouille de Textes.* Thése de Doctorat d'Université Montpellier II, 2009.

[Bou87] L. Bouquiaux. *Enquête et description des langues à tradition orale*, 1987.

[CG06] Poudat C and Cleuziou G. *Catégorisation de textes en domaines et genres. Complémentarité des indexations lexicale et morphosyntaxique*, 2006.

[DN03] Bourigault D and Aussenac-Gilles N. *Construction d'ontologies à partir de textes.* Université Paul Sabatier 118, 2003.

[Emm08] Puybaret Emmanuel. *Java 1.4 et 1.5 (3ème édition).* EYROLLES, 2008.

[Far09] HARRATHI Farah. *Extraction de concepts et de relations entre concepts à partir des documents multilingues : Approche statistique et ontologique.* Thèse de Doctorat, L'Institut Nationale des Sciences Appliquées de Lyon, 2009.

[Fav07] Cecile Favre. *Évolution de schémas dans les entrepôts de données :mise à jour de hiérarchies de dimension pour la personnalisation des analyses.* Thèse de doctorat de Laboratoire ERIC, 2007.

[FE06] Archetti F and Fersini E. *A hierarchical document clustering environment based on the induced k-means*, 2006.

[FO07] Ravat Franck and Teste Olivier. *Analyse multidimensionnelle de documents via des dimensions OLAP.* Institut de recherche en informatique Université Toulouse 3, 2007.

[FO08] Ravat Franck and Teste Olivier. *Top-Keyword : agrégation de mots-clefs dans un environnement d'analyse en ligne (OLAP).* Toulouse CEDEX 9 (France), 2008.

[FRA00] ravat FRANCK. *Modélisation et manipulation d'entrepôts de données complexes et historisées.* Thèse de doctorat de l'UNIVERSITE PAUL SABATIER DE TOULOUSE (SCIENCES), 2000.

[FRA07] RAVAT FRANCK. *Modéles et outils pour la conception et la manipulation de systéme d'aide à la décision.* mémoire en vu d'obtention du diplôme d'habilitation de l'UNIVERSITE des sciences sociales (Toulouse I), 2007.

[KM07] KHROUF Kais and MBARKI Mohamed. *Les entrepôts de documents : gestion des versions*. Université Toulouse, 2007.

[Lam06] NAOUM Lamiaa. *Un modèle multidimensionnel pour un processus d'analyse en ligne de résumés flous*. Thèse de doctorat d'Université de Nantes, 2006.

[RC08] Brussee R and Wartena C. *Topic Detection by Clustering Keywords*, 2008.

[Ron07] Tournier Ronan. *Analyse en ligne (OLAP) de documents*. Thèse de doctorat d'Université Toulouse III- Paul Sabatier (France), 2007.

[SA10] Bringay S and Laurent A. *Bien cube, les données textuelles peuvent s'agréger*. France, 2010.

[TP98] Landauer T and Folltz P. *An introduction to Latent Semantic Analysis*, 1998.

Annexe A

Tableau de poids des termes

TABLE A.1: Les poids des termes obtenus par tf.idf simple et tf.idf complexe

Les termes	**Les tf-idfs complexe**	**Les tf-idfs simple**
factor	0.1	0.00405
result	0.02	0.01575
line	0.12	0.00408
large	0.04	0.00417
main	0.02	0.00417
distribution	0.09	0.00394
software	0.06	0.0043
study	0.06	0.00579
accuracy	0.11	0.00428
source	0.07	0.00512
threshold	0.07	0.00458
Space	0.08	0.00598
region	0.19	0.00439
management	0.08	0.00447
bandwidth	0.12	0.00458
capacity	0.15	0.00489
experiment	0.09	0.00458
event	0.12	0.00464
knowledge	0.07	0.00467
simulation	0.11	0.00554
recognition	0.28	0.00472
sequence	0.16	0.00472
design	0.09	0.00512
position	0.14	0.00551
graph	0.22	0.00598
present	0.02	0.00321
distance	0.09	0.00484
component	0.06	0.00503
decision	0.13	0.00548
database	0.1	0.00514

Suite à la page suivante

TABLE A.1 – *Suite de la page précédente*

Les termes	**Les tf-idfs complexe**	**Les tf-idfs simple**
class	0.24	0.00834
access	0.11	0.00548
semantic	0.19	0.00548
structure	0.09	0.00635
agent	0.37	0.00528
architecture	0.09	0.00618
cloud	0.29	0.00551
recommendation	0.28	0.00528
scheme	0.18	0.00808
transaction	0.08	0.00542
query	0.24	0.00811
generation	0.23	0.00803
cell	0.45	0.00559
detection	0.17	0.00582
environment	0.08	0.00562
tool	0.17	0.00573
example	0.03	0.00705
request	0.18	0.00584
cluster	0.24	0.00587
evaluation	0.08	0.00587
sensor	0.25	0.00828
distributed	0.09	0.0059
security	0.22	0.00601
analysis	0.04	0.00601
research	0.03	0.0064
vector	0.34	0.00629
function	0.05	0.0064
size	0.07	0.00764
layer	0.21	0.00677
computer	0.02	0.00738
language	0.22	0.00694
wireless	0.17	0.00694
item	0.17	0.00408
object	0.2	0.00699
message	0.22	0.00727
protocol	0.16	0.00649
error	0.2	0.00736
approach	0.04	0.00945
order	0.02	0.00758
process	0.05	0.00859
type	0.05	0.00769
problem	0.06	0.00792
case	0.01	0.0080
video	0.23	0.00845
signal	0.21	0.0085

Suite à la page suivante

TABLE A.1 – *Suite de la page précédente*

Les termes	**Les tf-idfs complexe**	**Les tf-idfs simple**
mobile	0.22	0.00836
performance	0.04	0.00923
test	0.14	0.00643
bit	0.32	0.00652
table	0.06	0.00979
control	0.15	0.00954
location	0.13	0.00926
context	0.21	0.00985
channel	0.36	0.00691
communication	0.08	0.00987
search	0.06	0.00092
power	0.29	0.01063
feature	0.16	0.01035
word	0.3	0.01043
technology	0.08	0.01083
application	0.03	0.01111
level	0.09	0.01105
work	0.04	0.01186
technique	0.1	0.01217
value	0.09	0.0134
packet	0.63	0.01203
provide	0.02	0.00551
set	0.11	0.01626
image	0.64	0.01654
web	0.49	0.01754
time	0.05	0.01751
user	0.18	0.00901
number	0.06	0.01908
method	0.17	0.01925
information	0.08	0.02146
model	0.24	0.02199
service	0.38	0.023
data	0.08	0.02476
network	0.31	0.02571
system	0.06	0.04088

Annexe B

Les fonctions appelées dans le programme

Algorithme de la fonction TF

Données : $mat[1..n][1..m]$ matrice de fréquences.
Résultat : $m[1..n][1..m]$ matrice de TF.
début
  pour tous les i *de* 1 *á* n **faire**
    $S \leftarrow 0$;
    pour tous les j *de* 1 *á* m **faire**
      $S \leftarrow S + mat[i, j]$;
    pour tous les j *de* 1 *á* m **faire**
      $m[i, j] \leftarrow mat[i, j]/s$;

Algorithme 1 : TF(**d** $mat[1..n][1..m]$: matrice de fréquences. **r** $m[1..n][1..m]$ matrice de TF.)

Algorithme de la fonction som

Données : $m[1..n][1..m]$ matrice de TF.
Résultat : $t[1..m]$ tableau des som de TF des termes.
début
  pour tous les j *de* 1 *á* m **faire**
    $Som \leftarrow 0$;
    pour tous les i *de* 1 *á* n **faire**
      $Som \leftarrow Som + m[i, j]$;
    $t[j] \leftarrow som$;

Algorithme 2 : som (**d** $m[1..n][1..m]$: matrice de TF. **r** $t[1..m]$ tableau des som des termes.)

Algorithme de la fonction IDF

```
Données : m[1..n][1..m] matrice de fréquence.
Résultat : t[1..m] tableau des IDF des termes.
début
    nbr_doc ← n + 1;
    /* +1 pour éviter log zéro */
    pour tous les j de 1 á m faire
        nbr_dtr ← 0;
        pour tous les i de 1 á n faire
            si m[i, j]] ≠ 0 alors nb_dtr ← nb_dtr + 1 ;;
        tab[j] ← log(nb_doc/nb − dtr);
```

Algorithme 3 : idf (**d** $m[1..n][1..m]$: matrice de fréquences. **r** $t[1..m]$tableau des IDF des termes.)

Algorithme de la fonction somme de toutes les fréquences

```
Données : m[1..n][1..m] tableau de fréquences.
Résultat : s : entier /* la somme de fréquences*/
début
    pour tous les i de 1 á n faire
        pour tous les j de 1 á m faire
            s ← s + m[i, j];
```

Algorithme 4 : nbtt (**d** $m[1..n][1..m]$ tableau de fréquences. **r** s : entier /* la somme de fréquences*/

Algorithme de la fonction tf.idf complexe

```
Données : tf[1..m] tableau de TF ; idf[1..m] tableau de idf.
Résultat : tf.idf[1..m] tableau de tf.idf de chaque terme.
début
    pour tous les i de 1 á m faire
        tf.idf[i] ← tf[i] × idf[i];
```

Algorithme 5 : tf.idf (**d** $tf[1..m]$ tableau de TF ; $idf[1..m]$ tableau de idf ; **r** $tf.idf[1..m]$ tableau de tf.idf de chaque terme.)

Algorithme de la fonction max

Données : $tfidf[1..m]$ tableau de tf.idf.
Résultat : ind : entier, indice de la valeur maximale.
début
  $x \leftarrow tfidf[1]$;
  $ind \leftarrow 1$;
  pour tous les i *de* 2 *á* m **faire**
    si $(tfidf[i] > x)$ **alors**
    $x \leftarrow tfidf[i]$;
    $ind \leftarrow i$;

Algorithme 6 : max (**d** $tfidf[1..m]$ tableau de tf.idf ; **r** ind : entier indice de la valeur maximale.)

Algorithme de la fonction tf.idf simple

Données : $m[1..n][1..m]$ matrice de fréquences ; som : entier
Résultat : $tfidf[1..m]$ tableau de tf.idf
début
  pour tous les j *de* 1 *á* m **faire**
    /* som_fr : la somme de fréquences d'un seule terme */
    $som\_fr \leftarrow 0$;
    pour tous les i *de* 1 *á* n **faire**
      $som\_fr \leftarrow som\_fr$+m[i,j];
    $tfidf[j]$ ←som_fr/som;

Algorithme 7 : topkw (**d** $m[1..n][1..m]$ tableau de fréquences ; som : entier. **r** $tfidf[1..m]$ tableau de tf.idf.)

Algorithme de la fonction qui calcule la distance entre les termes

Données : $m[1..n][1..m]$ tableau de fréquences ;
Résultat : $dist[1..n][1..m]$ matrice de distances
début
  pour tous les j *de* 1 *á* m **faire**
    $s \leftarrow 0$
    pour tous les i *de* 1 *á* n **faire**
      $s \leftarrow$s+m[i,j];
    $corpus_fmc[j] \leftarrow$s;
  pour tous les i *de* 1 *á* m **faire**
    pour tous les j *de* i *á* m **faire**
      $som \leftarrow 0$;
      $som1 \leftarrow 0$;
      $som2 \leftarrow 0$;
      si $(i = j)$ **alors** $dist[i, j] \leftarrow 0$;
      sinon pour tous les p *de* 1 *á* n **faire**
        $som \leftarrow$som+((m[p,i]/corpus$_f mc[i]$) $\times$(m[p,j]/corpus$_f mc[j]$) ;
        $som1 \leftarrow$som1+$(m[p, i]/corpus\_fmc[i])^2$;
        $som2 \leftarrow$som2+$(m[p, j]/corpus\_fmc[j])^2$;
      $x \leftarrow \sqrt{som1 \times som2}$;
      $sim \leftarrow$som/x;
      $dist[i, j] \leftarrow (1 - \cos(sim \times PI)/180)$;
      $dist[j, i] \leftarrow (1 - \cos(sim \times PI)/180)$;

Algorithme 8 : dist-topic (**d** $m[1..n][1..m]$ matrice de fréquences ; **r** $dist[1..n][1..m]$ matrice de distances.)

Algorithme de la fonction max de la classe kmeans

```
Données : m[1..n][1..m] matrice de distance ;
Résultat : tab[1..3] tableau de deux termes et leurs distance maximale
début
    tab[1] ← m[1, 1];
    tab[2] ← 1;
    tab[3] ← 1;
    pour tous les i de 1 á m faire
        pour tous les j de 1 á m faire
            si tab[1] < m[i, j] alors
            tab[1] ← m[i, j];
            tab[2] ← i;
            tab[3] ← j ;
```

Algorithme 9 : max (**d** $m[1..n][1..m]$ matrice de fréquences. **r** $tab[1..3]$ tableau de deux termes et leurs valeur maximale)

Algorithme de la fonction qui éclater une classe en deux

Données : $mat[1..m]$ matrice de distances ; $tab[1..3]$ tableau de deux termes et leurs distance maximale
Résultat : $liste[1..2][1..m]$ matrice de deux tableaux /* cluster1[], cluster2[] */
début
  pour tous les i *de* 1 *á* m **faire**
    $tab\_ind[i] \leftarrow i$;
  $tab\_ind[tab[2]] \leftarrow$ -1;
  $tab\_ind[tab[3]] \leftarrow$ -1;
  pour tous les i *de* 1 *á* m **faire**
    si $(mat[tab[2], i] < mat[tab[3], i])$ **alors**
      si *(tab_ind[i] ≠-1)***alors**
        cluster1.ajouter(i) ;
        $tab\_ind[i] \leftarrow$*-1*;
        $tab[2] \leftarrow i$;
    ;
      finsi
    sinon si *(tab_ind[i] ≠-1)***alors**
        cluster2.ajouter(i) ;
        $tab\_ind[i] \leftarrow$*-1*;
        $tab[3] \leftarrow i$;
    ;
      finsi
    finsi

Algorithme 10 : cluster (**d** $mat[1..3]$ matrice de distances ; $tab[1..3]$ tableau de deux termes et leurs distance maximale. **r** $liste[1..2][1..m]$ matrice de deux tableaux.)

Annexe C

Les classes finaux de la fonction Topic

Les classes	**Les termes**
classe (1)	approach
classe (2)	Work, object, main, decision, structure, architecture, tool, example, message, video, signal, mobile, bit, location, user, number, service, data.
classe (3)	Software, accuracy, source, capacity, event, recognition, sequence, design, vector, function, problem, network, system.
classe (4)	Management, database, distribution, study, threshold, position, graph, distance, component, distributed, analysis, protocol, error, process, type, case, technique, set, method, information.
classe (5)	Generation, transaction, item, result, packet, provide, image, web, time.
classe (6)	Size, region, knowledge, simulation, class, access, semantic, agent, cloud, scheme, query, cell, detection, environment, order, performance, table, control, context, channel, search, power, feature, technology, level.
classe (7)	Sensor, recommendation, factor, bandwidth, experiment, present, request, cluster, evaluation, layer, computer, language, communication, word.
classe (8)	application
classe (9)	Research
classe (10)	Model, wireless, space, line, large, security, test, value.

TABLE C.1 – Les termes composants chaque classe