Une approche simplifiée de l'annotation sémantique d'un corpus spécialisé à l'aide de ressources externes

Victor Ballu

Une approche simplifiée de l'annotation sémantique d'un corpus spécialisé à l'aide de ressources externes

Mémoire pour la validation du master *MEga Données et Analyses Sociales (MEDAS)*

Victor Ballu

Effectué au sein de : BNP PARIBAS - ANALYTICS CONSULTING
Années : 2016-2018
Tuteur pédagogique : Michel Bera
Maître d'apprentissage : Cédric Biron

Le 7 septembre 2018

Abstract

This master thesis aims to provide an efficient method to annotate documents semantically from a specialized corpus. It tries to address the complexity of ontology by using ad-hoc datas. Futhermore, it gives an account of the state of the art of document representation and focus on two main features : the Graph of Words algorithms and ontology. Its results are presented through an application of an Information Retrieval exercise.

Résumé

Ce mémoire de master a pour objectif de fournir une méthode efficace d'annotation sémantique des documents d'un corpus spécialisé. Il tente de réduire la complexité liée à la création d'ontologie en passant par le support de ressources ad-hoc. De plus, il fournit un état de l'art de la représentation de documents et détaille particulièrement l'algorithme du Graph of Words et les ontologies. Les résultats sont présentés à travers un application d'un exercice de recherche de document.

Keywords - Mots clefs

Annotation sémantique - concepts - Graph of Word - Ontologie - Représentation - Information Retrieval - NLP

Introduction

La profusion des ressources, qu'elles viennent du Web ou bien de documents internes aux entreprises est telle qu'il est devenu fondamental de pouvoir les organiser et les manipuler. Être capable de retrouver l'information recherchée est important, mais plus encore apparaît le besoin d'en fournir le contexte à d'autres machines. L'enjeu est donc la capacité à situer les informations, les documents les uns relativement aux autres pour qu'un humain et une machine puissent en comprendre le sens. Pour y répondre, de nombreux travaux ont vu le jour, notamment depuis l'avènement du Web sémantique qui a permis à ces recherches de s'accélérer. Le concept d'ontologie est désormais bien établi, et les méthodes de construction sont de plus en plus simples. En effet, l'existence de nombreuses ressources externes a permis de faciliter la contextualisation des données. C'est autant d'aides à l'automatisation des processus. Cependant, la mise en place d'une base de connaissance à travers des ontologies demande du temps et de lourds calculs. Cela réclame généralement un fort investissement et une grande rigueur des acteurs. C'est pourquoi nous allons présenter dans ce mémoire une méthode s'inspirant des nombreux concepts de représentation existants pour donner du sens aux documents avec un traitement limité et simplifié, mais gardant une efficacité suffisante pour la majorité des usages. Nous l'illustrerons à travers un exercice de recherche de document.

Nous allons donc rappeler les différents enjeux de la création de données sémantiques en détaillant un peu plus particulièrement la recherche d'information. Puis dans un second chapitre, nous présenterons les manières classiques de représenter sémantiquement des documents, en nous attardant un peu plus sur les **ontologies** ainsi que les **graphes de mots**. En effet, nous reprendrons une grande partie de ces concepts dans la méthode que nous proposerons. Nous détaillerons notre méthodologie et présenterons les résultats de notre expérimentation dans une seconde partie de ce document.

Ce que nous ne prétendons pas réaliser

Ce que nous proposons a pour objectif de simplifier l'intégration de documents dans un champ sémantique **spécifique**. Nous travaillons donc sur des **corpus spécialisés**, c'est-à-dire dont les applications sont spécifiques. Il ne s'agit pas pour nous de prétendre simplifier la création d'ontologies généralistes avec cette méthode.

Remerciements

La rédaction de ce document doit grandement à mes collègues du service *Analytics Consulting* du pôle *CIB* de la *BNP PARIBAS*. Une équipe patiente et ambitieuse contribuant beaucoup à la recherche en science de la donnée. Je tiens également à remercier ma collègue de master Imène Agar-Faidi qui m'a *challengé* tout le long de notre parcours commun. Je remercie également mes amis qui m'ont soutenu et conseillé, particulèrement Gabriel Renault qui a eu la patience de relire un grand nombre de mes travaux, ainsi que Charline Fraioli pour sa relecture et son soutien.
Bien évidemment, je remercie Cédric Biron, qui a cru en moi en tant que maitre d'alternance chez *Analytics Consulting* et m'a beaucoup appris sur le fonctionnement des grandes entreprises et l'aboutissement des projets, ainsi que Michel Bera qui m'a amicalement conseillé pour la rédaction pas toujours évidente d'un mémoire dans le cadre d'une alternance.
Enfin je remercie toute l'équipe pédagogique et administrative du master MEDAS pour sa grande patience et son écoute.

Table des matières

Première partie

Revue de l'état de l'art

Cette partie est consacrée à une revue des différentes méthodes de *représentation sémantique* des documents. Notre objectif étant de fournir une approche simplifiée de traitement des documents dans le cadre de **corpus spécialisés**, nous rappellerons brièvement l'histoire de la structuration sémantique des documents, portée par la *recherche d'information* et l'*extraction d'informations* (1.1) d'un côté, et l'émergence du *Web sémantique* (1.2) de l'autre. Nous présenterons ensuite les méthodes les plus répandues pour répondre à ces problèmes. Nous exposerons donc le concept d'*ontologie* (2.3), la *représentation vectorielle* des mots (2.4), l'usage des *réseaux de neurones* (2.5) à travers le *Word2vect* et enfin l'utilité des *graphes* pour extraire les mots clefs d'un document (2.6). Nous détaillerons particulièrement le *Graph of Word* (2.6) ainsi que la notion *d'ontologie* (2.3), la méthode que nous présentons s'appuyant grandement sur ces deux idées. Nous finirons cette partie par le détail méthodologique de la construction d'ontologie (3), de la présentation des contraintes à l'usage possible de ressources externes. Cette partie servira de socle théorique à notre proposition méthodologique qui en reprendra les grandes lignes dans la seconde partie (II).

Chapitre 1

La donnée sémantique

Historiquement, intégrer des documents dans un champ conceptuel était surtout nécessaire pour l'indexation, l'archivage et la gestion de documents. Le développement des approches basées sur la donnée et l'émergence du *Web sémantique* ont mis cette activité au premier plan. Les techniques d'*information retrieval*[1] et d'*information extraction*[2] se sont alors développées pour répondre aux besoins de ces activités.

1.1 L'*information retrieval* par rapport à l'*information extraction*

Que ce soit sur le Web, ou dans des fonds documentaires, la recherche du document adéquat, voire la réponse à une question, est un problème difficile. D'autant plus difficile qu'est importante la quantité de documents et leur diversité (c'est-à-dire leur non spécificité à un domaine particulier). Ce qui est particulièrement vrai pour le Web.

La recherche d'information est à distinguer de *l'extraction d'information*. La première activité consiste à retrouver un document satisfaisant à une requête. Tandis que la seconde consiste à *construire* une réponse en s'appuyant sur l'étude d'un corpus de documents ou une base de connaissance.
Malgré leur différence, ces deux activités partagent des points communs. En effet, il s'agit dans les deux cas de distinguer une partie gérant la connaissance et une autre les requêtes. La différence réside essentiellement dans la manière de restituer la connaissance.

L'*information retrieval* va renvoyer le document contenant la réponse tandis que l'*information extraction* va construire une réponse adaptée dans le langage naturel à partir des éléments

1. Recherche d'information
2. Extraction d'information

inférés d'une base de connaissance.

Dans ce document, nous illustrons notre méthode par un exercice d'*information retrieval*. Mais le travail serait similaire pour de l'extraction d'information. Cependant, la génération de réponse à partir de connaissances est un exercice difficile qui n'est pas le sujet de ce document.

1.1.1 Principes généraux

Les activités de recherche ou d'extraction d'information se décomposent en deux parties principales :

— la représentation de la connaissance,
— l'analyse de la requête.

Dans l'article *Indexation sémantique des documents multilingues*[3], les auteurs expliquent que, quelles que soient les méthodes utilisées, il s'agit de caractériser les documents et les requêtes par ce qu'ils appellent des *descripteurs*. Ces *descripteurs* sont indexés, et l'étape de recherche consiste à trouver les *descripteurs* des documents les mieux adaptés pour répondre à ceux caractérisant la requête. Il s'agit donc de trouver une fonction *RSV* (pour *Retrieval Status Value*) :

$$\begin{array}{rcccc} I_q & : & Q & \to & E \\ & & q & \mapsto & I_q(q) \end{array} \tag{1.1}$$

$$\begin{array}{rcccc} I_d & : & D & \to & E \\ & & & \mapsto & I_d(d) \end{array} \tag{1.2}$$

$$\begin{array}{rccc} RSV & : & E \times E & \to & R^+ \\ & & (I_q(q), I_d(d)) & \mapsto & RSV(I_q(q), I_d(d)) \end{array} \tag{1.3}$$

avec Q, D, E, respectivement l'ensemble des requêtes, des documents et des descripteurs. I_q est donc la fonction de représentation des requêtes, I_d est la fonction de représentation des documents.

1.1.2 Les modèles classiques

Un rapide historique de l'*information retrieval* est donné par les auteurs de *Information retrieval : An overview*[4]. Il y est rappelé que l'organisation des ressources documentaires est sûrement aussi ancienne que les ressources documentaires elles mêmes. En effet, les auteurs rappellent que les Sumériens possédaient des espaces de stockage à l'organisation spécifique réservés à leurs tablettes de glaise dès 3000 avant J.C. Mais c'est l'invention de l'écriture, et

plus encore de l'imprimerie, qui va vraiment développer l'organisation documentaire. L'apparition de l'ordinateur fait place à l'automatisation de cette organisation dès 1945, initié par l'article *As we may think* de Vannevar Bush. Dès les années 1950, l'indexation numérique de textes archivés commence à se développer. Depuis, les méthodes se perfectionnent, mais certains grands concepts restent centraux.

Le plus courant, bien qu'en perte de vitesse, est le modèle **booléen**. Il s'agit d'indexer les documents selon des mots clefs et des catégories. La recherche se fait de la même manière par *conjonction* ou *disjonction* de mots clefs et catégories sur des champs extrêmement structurés. Cependant, ce modèle n'est pas très pratique à l'usage. C'est pourquoi se sont développés les modèles **vectoriels** puis **probabilistes**. Pour les deux, il s'agit de construire une représentation du document selon divers critères, puis de trouver une représentation de la *question* (requête) se rapprochant le plus de celle d'un des documents. La difficulté vient de la différence entre la taille de la question et celle des documents. La dimension des représentations est alors souvent différente.
Les modèles utilisant les *espaces vectorialisés* s'appuient principalement sur une **étude statistique** de la distribution des mots dans les documents, tandis que les *modèles probabilistes* tentent d'ajouter autant de dimensions que possible pour affiner cette représentation en attribuant à chaque document une probabilité de correspondance à la question posée. Ces deux derniers modèles ont également l'avantage d'être plus adaptés au Web et aux corpus volumineux. On les détaillera plus amplement en 3.1.

1.2 Le Web sémantique

Le Web sémantique consiste en une standardisation des protocoles d'échanges par le W3C [3] dans le but d'enrichir la donnée et la rendre plus compréhensible par les automates. En effet, cette nécessité est apparue avec l'augmentation des échanges de données entre machines. L'échange ne se fait plus entre humains et ordinateurs, mais entre deux ordinateurs. Il a été formalisé pour la première fois en 2001 par T. Berners-Lee dans [5].

Les protocoles proposés par le W3C dans le but du *Web sémantique* ne considèrent que l'objectif d'interopérabilité entre logiciels. La structuration du Web pour l'intégrer dans la sphère du *Web sémantique*, à l'aide de divers outils de traitement, notamment de traitements du langage, relève du Traitement Automatique du Langage (TAL). C'est à une partie de ce dernier point que l'on proposera une solution.

L'enjeu est donc de structurer les documents initialement non structurés pour les rendre aisément exploitables par des calculateurs.

3. World Wide Web Consortium. Organisme de standardisation des technologies du World Wide Web

Chapitre 2

La représentation sémantique des documents

L'objectif d'interopérabilité sémantique entre différentes machines nécessite donc d'être capable de représenter les documents. Nous verrons plusieurs procédés, certains proposant un cadre conceptuel explicite, d'autres, plus abstraits, s'appuyant sur des outils statistiques.

2.1 L'annotation sémantique et les ontologies

Avant de détailler plus en avant les techniques de représentation, il nous semble important de préciser un point de vocabulaire. On parle d'**ontologie** [1] lorsque l'on définit de manière inductive, à partir d'un vocabulaire, différents champs conceptuels que l'on lie les uns aux autres par des relations. On considérera l'**annotation sémantique** comme l'exercice liant les mots d'un document à un concept d'une ontologie.

*L'**annotation sémantique** est donc l'activité attachant tout ou une partie des mots d'un document aux concepts qu'ils incarnent.*

Ainsi, l'annotation sémantique de la phrase

"La bière est faite d'orge ou de houblon."

pourrait être :

Bière	*boisson alcoolisée*
orge	*céréale*
houblon	*plante herbacée*

1. Voir 2.3.2

Cette précision est importante, ce document proposant en effet de simplifier l'*annotation sémantique*.

2.2 Principes généraux de la représentation sémantique

La représentation sémantique d'un document a pour objectif de rendre compréhensible par une machine le contexte et les concepts qui y sont sous-jacents. L'idée générale est donc de tenter de placer chaque document sur une trame sémantique, préalablement construite ou non. L'enjeu des techniques que l'on va présenter est d'automatiser le plus possible cette tâche. Apparaît également rapidement la problématique de la création de la trame. En effet, il n'existe pas de monde platonicien explicitement défini dans lequel tout objet peut être projeté. Il n'existe pas d'ontologie absolue.

Il va donc falloir tenter d'identifier des sens qui semblent pertinents et qui constitueront notre trame, puis y projeter des documents.

Nous appelons ce procédé la *représentation*. Il va s'agir de trouver un moyen de représenter numériquement des documents qui, même s'ils sont différents, auront une représentation similaire s'ils ont un sens voisin.

Plusieurs techniques ont été développées. Elles peuvent s'appuyer sur une **approche statistique** ou **linguistique**. Cependant, la seconde requiert une bonne représentation de la langue initiale dans l'algorithme. L'objectif étant de simplifier et d'automatiser ce processus, nous nous appuierons sur des méthodes plutôt statistiques et des données externes.

2.3 L'indexation et le formalisme des concepts - l'ontologie

2.3.1 Définition intuitive d'une ontologie

En philosophie, la notion d'ontologie se réfère à la réalité et à son organisation[6]. Cette définition est reprise en théorie de l'information pour caractériser un cadre dans lequel peuvent s'exprimer des concepts. Il s'agit de formaliser un espace de relations et de concepts articulés les uns relativement aux autres. En projetant des documents sur cette ontologie, il devient possible de les faire dialoguer, de représenter des actions, des relations.

Une définition populaire de l'ontologie en Intelligence Artificielle est donnée par Thomas R. Gruber : "*An ontology is an explicit specification of a conceptualization*" [2][7]. Ainsi, alors qu'un concept est une abstraction, Gruber précise que pour une intelligence artificielle, "*ce qui existe est ce qui peut être représenté*"[7]. Ainsi, une ontologie peut "*être représentée par*

2. Une ontologie est une spécification explicite de concepts

un ensemble représentatif de termes". Une ontologie est donc un ensemble de phrases de la théorie logique,[7] dont les concepts sont **induits** des références (des instanciations). De plus, ces concepts ainsi induits doivent être compris par les machines et les personnes se référant à cette ontologie. C'est pourquoi Studer[8] enrichie la formule de Gruber par *"An ontology is a formal explicit specification of shared conceptualization"* [3]

Une ontologie consiste donc en un ensemble de termes représentatifs de concepts qui sont liés les uns aux autres.

De nombreuses idées du formalisme des ontologies sont issues de la programmation objet, que ce soit au niveau du typage, de l'héritage et des propriétés.

Exemple - Une illustration simple, inspirée de l'article de Nathalie Aussenac-Gilles[9] [4], consiste à considérer des situations professionnelles. On a donc quelques concepts généraux :

— être humain,
— être une femme,
— être Française,
— être journaliste.

On a ainsi le concept de *femme* qui hérite du fait d'être *humain*, c'est à dire qu'il en gardera toutes les propriétés liées à la condition d'*humain* auxquelles sont ajoutées les propriétés du concept de *femme*.

Ce concept s'instancie dans des individus. Par exemple,

$$Florence\ Aubenas\ est\ une\ femme$$

Dans l'ontologie précédemment définie, on peut considérer *Florence Aubenas* comme une instanciation de la notion de *femme*, et par voie d'héritage, d'*humain*.

$$Florence\ Aubenas \in FEMME\ et\ FEMME \subset HUMAIN$$

De plus, les concepts sont en relation. On a déjà vu que le statut de *femme* héritait de celui d'*humain*. On peut également, par exemple, considérer que *Florence Aubenas* est *journaliste*.

Journaliste est une *profession*. On va donc considérer non plus la notion de concept en tant que propriété, mais également en tant que relation. On peut ainsi considérer que la notion de profession est une notion précise qui s'applique à deux objets : Un humain et un métier.

3. Une ontologie est une spécification explicite de concepts partagés
4. Et empreintée à J.Corman, étudiant en thèse dans MELODI

Soit la fonction $avoir\_pour\_profession$ telle que :

$$\begin{array}{rcccc} avoir\_pour\_profession & : & HUMAIN & \rightarrow & PROFESSION \\ & & x & \mapsto & avoir\_pour\_profession(x) \end{array}$$

On a ainsi,

$$avoir\_pour\_profession(FlorenceAubenas) = Journaliste$$

Avec $Journaliste \in PROFESSION$.

On voit ainsi apparaître les trois notions majeures lorsque l'on parle d'ontologie : **le sujet**, **le prédicat** et **l'objet**. On parle de **triplet sémantique**.

2.3.2 Définition formelle

Le formalisme que l'on va présenter est inspiré de l'article de Farah Harrathi, Catherine Roussey, Syvie Calabretto, Loïc Maissonnasse et Mohamed Mohsen Gammoudi : *Indexation sémantique des documents multilingues*[3]. Il décrit une ontologie initialement présenté par Gruber [7] dont les relations sont ordonnées. On peut formellement définir une ontologie comme un ensemble O :

$$O = \{C, R, \leq_c, \sigma_R, V_O\}$$

avec :
C : L'ensemble des **concepts** de l'ontologie. C'est à dire **les objets**.
R : L'ensemble des **relations** entre les concepts de l'ontologie. Il s'agit des **prédicats**.
$\leq_c$: Relation d'ordre partiel [5] sur C. Cela permet de hiérarchiser les sujets. C'est de cette manière qu'est représentée la notion d'héritage.
σ_R : Est la signature d'une relation entre deux éléments. Elle est définie de R dans $C \times C$
V_O : Ensemble du vocabulaire de l'ontologie. On a $V_O = \{V_{OC}, V_{OR}\}$. Avec V_{OC} et V_{OR} respectivement le vocabulaire des concepts et le vocabulaire des relations (autrement dit, le vocabulaire des objets et le vocabulaire des prédicats).

Relations et propriétés

Toujours suivant l'article [3], on va définir des propriétés sur cette définition qui nous permettront de mettre en évidence les opérations nécessaires pour l'usage des ontologies :

5. Une relation d'ordre est une relation binaire réflexive, antisymétrique et transitive

$$\forall c \in C, L_c(c) = \{t \in V_{OC} \textit{ tel que } t \textit{ dénote } c\} \tag{2.1}$$

$$\forall t \in V_{OC}, S_c(t) = \{c \in C \textit{ tel que } c \textit{ est labellisé par } t\} \tag{2.2}$$

$$\forall t \in V_O, L(t) = \{l \textit{ un langage du vocabulaire}\} \tag{2.3}$$

On a ainsi des opérations qui nous permettent d'interagir avec les concepts de l'ontologie. L_c et S_c sont symétriques. L_c retourne les instances possibles d'un concept, S_c le concept auquel est rattachée une instance. L renvoie quant à elle à la langue d'un mot.

Une évolution polyhiérarchique de l'ontologie

La représentation de connaissances généralistes n'est cependant pas toujours réalisable avec la structure d'ontologie précédemment introduite. En effet, certains concepts peuvent ne pas être en relations. De plus, la modification de l'état des faits du monde peut ne plus concorder avec l'ontologie ainsi créée. C'est ce que nous montre Nicolas Guarino dans *What is an Ontology ?*[10]. Il ajoute ainsi à la notion de présenté par Gruber[7] et enrichie par Studer[8] celle, inspirée de la logique modale, de mondes possibles. Il s'agit des différentes concrétisation *plausibles* que peuvent prendre les faits et les relations.

Une ontologie O consiste donc en un ensemble de formules logiques désignant aussi bien que possible les expressions de l'ensemble des concepts C et l'ensemble des relations R dans l'ensemble des mondes possibles W. R n'étant pas nécessairement un ensemble ordonné.

2.3.3 Langages et représentations

Dans cette partie, nous allons rapidement présenter différentes manières habituellement utilisées pour décrire des concepts.

***Ressource Description Framework* (RDF)** Développé par le W3C [6], ce modèle est chargé de décrire la représentation d'éléments sémantiques pour le Web. Y est notamment spécifié le triplet décrit précédemment dans l'exemple en (2.3.1) :

$$(sujet, prédicat, objet)$$

Le *sujet* représentant ce qui est décrit. On a $sujet \in I$ avec I l'ensemble des instances. Le *prédicat* représente la relation tandis que *l'objet* réfère au concept auquel est lié le sujet. Ainsi, $prédicat \in V_{OR}$ et $objet \in V_{OC}$.
Un objet appartient donc à un certain type. La notion de classe n'est cependant pas présente.

6. Voir note en 3

***Extensible Markup Language* (XML)** Le RDF est généralement exprimé à travers le formalisme du XML. Il s'agit d'un langage descriptif, structurant l'information à l'aide de balises.
Les documents XML sont structurés à partir d'une racine. C'est à dire un bloc balise ouvrante et balise fermante servant de racine au document et dans lequel se déploiera la hiérarchie du document (voir exemple en annexe A).

RDF Schéma (RDFS) Il s'agit d'une extension du RDF par le W3C [3]. Y est ainsi introduite la notion de *classes*, de *sous-classes*, de *propriété*. On peut ainsi définir un héritage de propriétés comme en programmation objet.

***Web Ontology Langage* (OWL)** Il s'agit encore d'une extension du RDFS. En effet, il ajoute à la notion de *classes*, *sous-classes*, *types* (propriété) et d'*instances*, des notions telles que des contraintes sur des classes. Par exemple, "tout humain a un sexe" [7]. Ainsi apparaît une notion de *classes abstraites*, c'est-à-dire de classes qui ne peuvent exister par elles-mêmes, mais qui partagent des propriétés entre plusieurs sous-classes :
Soit C une classe abstraite. Alors nécessairement :

$$\exists D \text{ une classe} \mid D \subset C$$

Les graphes Une grande partie des représentations que nous venons de présenter est implémentée en XML. La structure très hiérarchique de ce langage rend la représentation de ses parties sous formes d'arbres et de noeuds très naturelles. De même, il est alors aisé de représenter un réseau de noeuds à l'aide de balises désignant des noeuds destinations et cibles. Un exemple est fourni en annexe B.

On a vu comment une ontologie permet d'expliciter des concepts pour les exploiter de manière interopérable entre différentes machines. Cependant, ce n'est pas la seule représentation de document possible. Nous allons en détailler quelques-unes importantes.

2.4 GloVe : *Global Vectors for Word Representation* [1]

GloVe est une technique de **représentation statistique** du document en fonction du contexte. Elle a été publiée en 2014 par une équipe de l'université de *Stanford*.

7. Exemple tiré de Wikipédia. `https://fr.wikipedia.org/wiki/Web_Ontology_Language`

2.4.1 Le modèle

À partir d'une matrice de co-occurrences de mots X où la valeur X_{ij} affiche le nombre de fois où le mot j apparaît dans le contexte du mot i, on a donc $X_i = \sum_k X_{ik}$, qui représente le nombre de fois où chaque mot apparaît dans le contexte du mot i. On peut donc définir $P_{ij} = P(i|j) = X_{ij}/X_i$ la probabilité qu'un mot j apparaisse dans le contexte d'un mot i. L'apprentissage de la représentation vectorielle s'appuie sur l'idée que la relation entre deux mots $i = ice\ et\ j = steam$ peut être étudiée par la probabilité de leurs co-occurrences. Ainsi, les auteurs de [1] expliquent que pour un mot tel que $k = solid$, n'apparaissant que dans le contexte d'un des deux autres, on s'attend à ce que le ratio des probabilités P_{ik}/P_{jk} soit élevé. En effet, la probabilité que le mot solide apparaisse dans le contexte de la vapeur est faible tandis que la probabilité du mot solide dans le contexte de la glace semble élevée.

Ainsi, l'évaluation pour l'apprentissage des vecteurs de mots se fait avec le ratio de co-occurrences des probabilités :F

$$F(w_i, w_j, \tilde{w_k}) = \frac{P_{ik}}{P_{jk}} \tag{2.4}$$

où $w \in R^d$ sont des vecteurs de mots de dimension d et $\tilde{w} \in R$ sont des vecteurs de mots d'un contexte séparé. Cette distinction est liée à l'entraînement du modèle. En distinguant les deux espaces vectoriels, la construction du modèle peut être $w+\tilde{w}$. En effet, les deux modèles sont construits de la même manière, mais leurs points de départ différents permettent de réduire le sur-apprentissage.

Comme dans tout processus d'apprentissage, il s'agit pour l'algorithme de minimiser une fonction de coût :

$$\tilde{J} = \sum_{ij} f(X_{ij})(w_i^T \tilde{w}_j - \log X_{ij})^2 \tag{2.5}$$

où f [8] est une fonction de poids dont une expression possible est :

$$f(x) = \begin{cases} \frac{x}{x_{max}} & si\ x < x_{max} \\ 1 & sinon \end{cases}$$

2.4.2 Principe de fonctionnement

Concrètement, la représentation des documents selon le modèle GloVe[1] se déroule en quelques étapes simples.

8. La fonction f doit respecter les critères suivants :
   — $f(0) = 0$ et $\lim_{x\to 0} f(x)\log^2 x\ est\ finie$
   — f doit rester positive ou nulle
   — $f(x)$ doit rester relativement petit par rapport à une grande valeur de x

— Le contexte de co-occurrence d'un mot est défini par l'utilisateur. Il s'agit d'une distance autour du mot qui est analysé. Le poids accordé à chaque mot est une fonction de la distance.
— Comme nous l'avons vu au paragraphe précédent, le modèle, c'est-à-dire les vecteurs de mots w_k, est ajusté en minimisant la fonction de coût $\tilde{J}$.
— Ainsi, chaque mot est représenté par un vecteur. Toutes les opérations vectorielles peuvent y être appliquées, de même qu'un calcul de similarité peut être effectué.

Un exemple extrait du site text2vect [9] permet d'effectuer les calculs suivants.

Après avoir entraîné le modèle, les auteurs font les opérations sur les vecteurs des mots $paris$, $france$ et $germany$:

$$A = vector(\text{"}paris\text{"}) - vector(\text{"}france\text{"}) + vector(\text{"}germany\text{"})$$

Ensuite, ils cherchent le mot le plus proche vectoriellement du vecteur A obtenu. Pour ce faire, ils calculent la *cos similarity* [10]. Le mot le plus similaire qu'ils trouvent est *berlin*.

2.5 Les réseaux de neurones - Word2vec

L'idée d'entraîner un modèle de *machine learning* pour représenter des mots ou des documents n'est pas nouvelle. Cependant, jusqu'au **word2vec**[2], il s'agissait d'une tâche lourde en calcul.

2.5.1 Rappels sur les réseaux de neurones

La structure

Un réseau de neurones est constitué de plusieurs éléments :
— les neurones,
— la couche d'entrée,
— la couche de sortie,
— la (ou les) couche(s) intermédiaire(s).

9. Bibliothèque du logiciel R. `http://text2vec.org/index.html`. Détails de l'exemple : `http://text2vec.org/glove.html`

10. Similarité du cosinus. Il s'agit de trouver θ l'angle entre deux vecteurs A et B connus :

$$\cos\theta = \frac{A \bullet B}{||A|| \bullet ||B||}$$

où $|| \bullet ||$ représente la norme

Dans le reste de cette section, on va détailler la représentation d'un neurone (figure 2.1) ainsi que le fonctionnement d'un réseau.

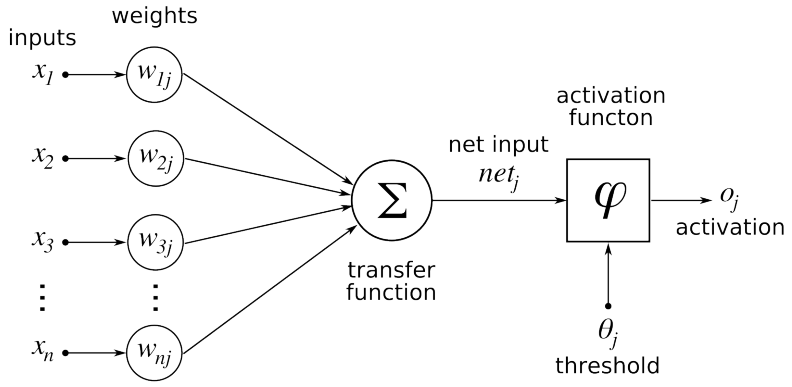

FIGURE 2.1 – Schéma d'un réseau de neurones. Ici, la fonction d'activation et la fonction de transfert sont détaillées.
source : *By Chrislb, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=224555*

Les neurones (figure 2.1) sont des petites unités de calculs très simples qui prennent en entrée une ou plusieurs valeurs et renvoient ou non une autre valeur.
La fonction principale du neurone, dite fonction d'activation, peut être très variée. Une couramment utilisée est la fonction sigmoïde :

$$f(x) = \frac{1}{1 + \exp^{-x}}$$

Mais il peut s'agir de n'importe quelle fonction.

La figure 2.1 montre clairement le fonctionnement d'un neurone. Les valeurs entrantes sont pondérées par un poids W et agrégées par une fonction de transfert. Puis la fonction d'activation retourne une valeur qui est transmise au reste du réseau selon un seuil (*threshold*).

La couche d'entrée ne contient pas de fonction d'activation. Elle est uniquement chargée de transmettre les données entrantes aux neurones.

La couche de sortie permet d'extraire les données calculées par le réseau. C'est une couche importante, notamment en catégorisation, où généralement elle contiendra autant de neurones que de catégories recherchées.

La (ou les) couche(s) intermédiaire(s) sont le coeur du réseau. Plus il y en a, plus le réseau est dit profond.

Les paramètres d'un réseau de neurones sont le nombre de couches, la taille des couches (le nombre de neurones qu'elles contiennent), les fonctions d'activation, la manière dont les couches sont connectées, les seuils et les poids.

Il y a deux manières d'intervenir sur ces paramètres :
- **l'architecture** du réseau que l'on choisit va influer sur les fonctions d'activations, les couches, le nombre de neurones, les connexions entre les neurones ;
- **la phase d'apprentissage** va quant à elle paramétrer les poids et les seuils d'activations.

Il y a deux usages majeurs aux réseaux de neurones :

La classification : il s'agit de prédire des catégories, c'est-à-dire de définir à quelle classe appartient un élément.

La régression : il s'agit de prédire une valeur numérique en fonction de l'entrée.

De même, il y a deux grandes façons d'entraîner un réseau de neurones :

L'apprentissage non supervisé : il va s'agir pour le réseau de prédire au mieux les différentes classes, sans autres informations *ad-hoc* sur les résultats. C'est une méthode assez efficace pour des problématiques de classifications.

L'apprentissage supervisé : il s'agit d'entraîner un modèle sur des données d'exemples sur lesquelles on a des informations supplémentaires aidant à l'entraînement. Ceci fait, il est alors possible de prédire des résultats sur des données sur lesquelles on ne présume rien.
Cette méthode étant utilisée par le word2vec, nous allons la détailler.

Le fonctionnement - l'apprentissage supervisé

Un réseau de neurones permet d'extraire des comportements émergents d'une grande somme de variables faibles, difficilement identifiables.
Une fois les architectures choisies, il y a donc deux phases. Une phase d'entraînement, permettant de calibrer le réseau aux données, et une autre d'usage, permettant de "prédire" un comportement.

L'entraînement

On peut considérer le réseau comme une fonction f. L'objectif de la phase d'apprentissage est, à partir de données connues, de calibrer le réseau pour qu'il s'approche le plus possible du résultat attendu.

Soit un ensemble de données d'entraînement T labellisées tel que $T = (X, Y)$ avec X les données et Y les étiquettes associées. L'objectif du réseau de neurones est de minimiser l'erreur d'association de l'étiquette à la donnée correspondante :
ainsi, pour tout $x \in X$ le réseau va tenter de trouver $y \in Y$ tel que

$$f(x) = y$$

Cependant, l'approximation par le réseau n'est pas parfaite, il y a une *erreur* ou un *coût* C pouvant [11] être représenté par :

$$C = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2$$

où $n = card(T)$ et y_i est la valeur théorique associée à la donnée x_i. L'objectif de la phase d'apprentissage va être de minimiser cette fonction.

Le calibrage du réseau se fait en ajustant les poids et les seuils d'activations de chaque neurone. Pour cela, la plupart des algorithmes utilisent une variante du *Gradient Descent*. Comme nous l'avons vu, l'objectif du réseau de neurone est de minimiser la fonction de coût (l'erreur). Le principe du gradient consiste à regarder l'impact de la variation des paramètres [12] d'un noeud sur l'erreur. Ainsi, pour chaque noeud, on calcule :

$$\frac{\partial C}{\partial w}$$

qui représente la dérivée partielle des paramètres.
Ce calcul est effectué en commençant par la dernière couche du réseau, celle la plus proche de la sortie et du résultat attendu. La fonction de coût dépendant de la valeur du résultat, et lui-même dépendant de la valeur de sortie du dernier noeud, il est alors possible de répercuter la fonction de coût sur chaque noeud du réseau :

$$C_k = f(C_{k-1})$$

où C_k est le coût pour le noeud sur la couche k.
Cette équation met en évidence la dépendance de l'état d'un noeud aux paramètres du noeud

11. La fonction présentée ici est l'*erreur quadratique moyenne*
12. Le poids, le seuil

précédent (ici les paramètres sont abstraits par f).
La *taille* de la variation est appelée le *learning rate*. Ainsi, un faible *learning rate* va rendre le processus d'apprentissage très lent. Mais un *learning rate* trop élevé risque de ne jamais atteindre la configuration optimum pour les paramètres (voir figure 2.2). Le *pas* est alors trop important, et l'algorithme "saute" le minimum à chaque variation.

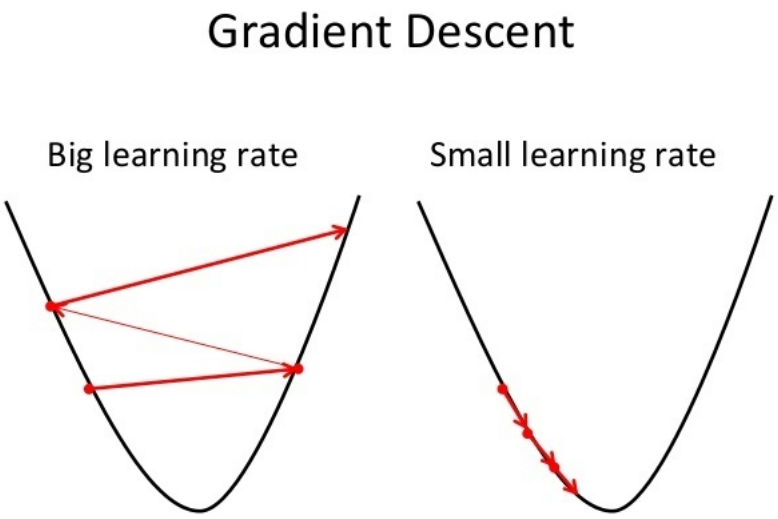

FIGURE 2.2 – Illustration de l'importance du learning rate
source : https://www.quora.com/What-is-the-meaning-of-changing-the-LR-learning-rate-in-neural-networks

2.5.2 La représentation sémantique des mots à l'aide du Word2vec[2]

En 2013, une équipe de Google, dirigée par Thomas Mikolov, propose ce que l'on nomme désormais le word2vec[2]. Il s'agit d'une technique de *machine learning* supervisée permettant une représentation sémantique des mots d'un document en s'appuyant sur le contexte. Il y a deux architectures possibles :

— *Continuous Bag of Word* (*CBOW*)
— *Continuous Skip-gram Model*

L'architecture CBOW (fig 2.3) est constituée d'une grande couche d'entrée prenant en compte les mots encadrant le mot recherché. La couche de sortie a pour objectif de prédire le mot.

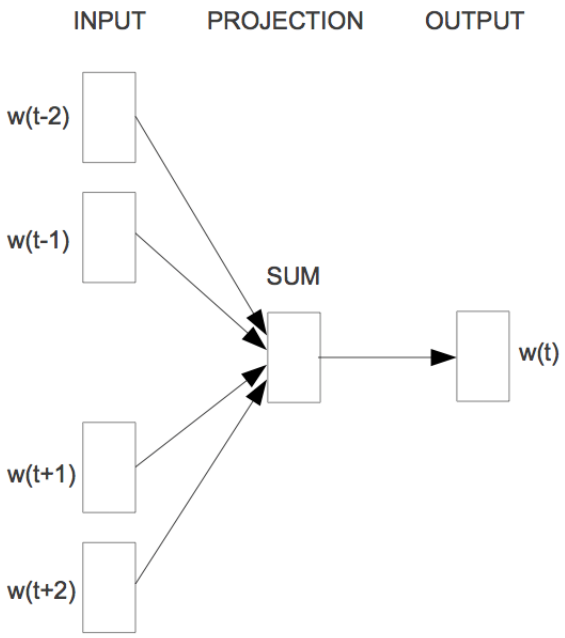

FIGURE 2.3 – Architecture CBOW du word2vec. On remarque les différents mots w en entrée entourant la position courante.

L'architecture Skip-gram (fig 2.4) est quant à elle constituée d'un unique neurone d'entrée, cependant sa couche de sortie a pour objectif de prédire les mots entourant le mot en entrée.

Selon l'architecture choisie, le réseau est entraîné sur le corpus. Bien qu'il ne soit pas possible de décrire une ontologie, le word2vec permet la contextualisation des mots selon leurs contextes.
Il s'agit d'une technique de vectorisation. Deux mots ayant un sens similaire auront alors une direction similaire.

La représentation des documents par le word2vec a été déclinée en de nombreuses variantes. Ainsi, il est possible d'utiliser la granularité du caractère ou du document à la place du mot. [13]

Les auteurs[2] donnent un exemple d'utilisation similaire à celui présenté pour l'algorithme *GloVe* (2.4).

13. Par exemples, le *doc2vec* ou le *char2vec*

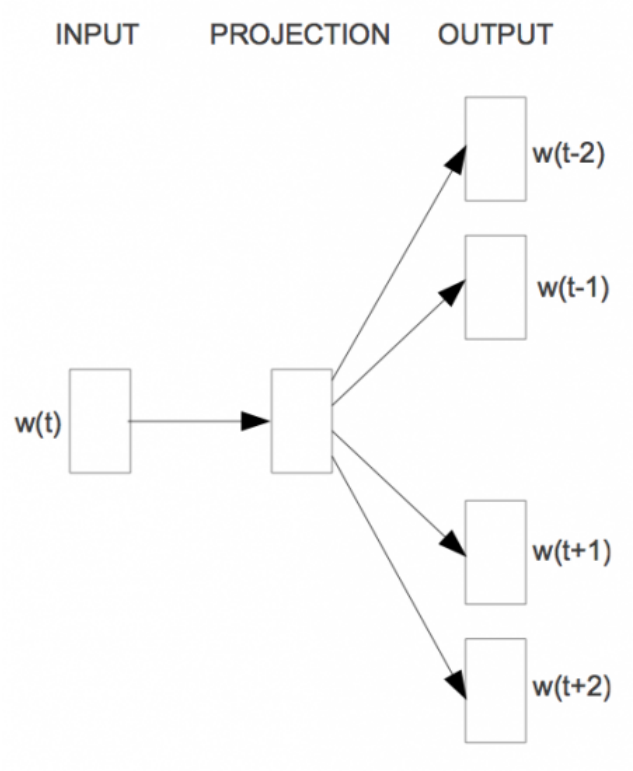

FIGURE 2.4 – Architecture skip-gram

Ils utilisent pour leur part les vecteurs des mots *biggest*, *big* et *small*.

$$X = vector(\text{"}biggest\text{"}) - vector(\text{"}big\text{"}) + vector(\text{"}small\text{"})$$

En cherchant le vecteur le plus proche du vecteur X obtenu, ils obtiennent *smallest*[14].

2.6 Graph Of Words - extraction des mots clefs

François Rousseay et Michalis Vazirgiannis [11] proposent de représenter des documents par l'identification de mots clefs à l'aide de graphes. Il s'agit d'une amélioration du concept de *bag of words*[15][12].

Le *bag of words* proposait de représenter un document à l'aide d'un vecteur dénombrant les occurrences des mots qui le composent. Le *graph of words* quant à lui tente de représenter la co-occurrence à l'aide d'un graphe liant les mots.

14. Un autre résultat extrêmement connu est $king = queen - woman + man$
15. Sac de mots

Il s'agit cette fois de connecter les mots co-occurrents. Ainsi la dépendance, c'est à dire l'ordre et la distance des mots les uns par rapport aux autres, est prise en compte.

L'illustration fournie par les auteurs [11] est la suivante (exemple 2.6 et fig 2.5) :

information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources

On obtient la représentation sous forme de graphe. Voir figure 2.5.

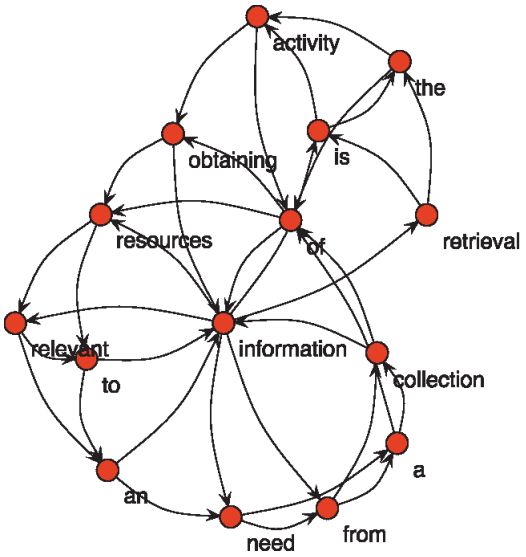

FIGURE 2.5 – Représentation de l'exemple 2.6

La pondération de chaque mot est calculée par le nombre d'arêtes arrivant sur le noeud du mot. Ainsi *information* (voir figure 2.5) a ici un poids de 5 et *retrieval* de 1.

L'extraction des mots clefs se fait en utilisant l'algorithme des *k-core*. Il s'agit d'itérer sur $k \in N^+$ jusqu'à ce que le *core* restant soit vide. On conserve alors la valeur de k précédente.

Un $k-core$ est un *sous-graphe* dont le degré est au moins égal à k.
Soit G un graphe, on dit que G_i est un $k-core$ de G si

$$G_i \subset G \; et \; pour \; tout \; x_r \in G_i, \; deg(x_k) \geq k$$

avec *deg* la fonction mesurant le degré d'un noeud x_r ; $r \in N^+$ représentant une indexation des noeuds.

Le degré d'un noeud est le nombre de connexions que le noeud possède avec d'autres noeuds.

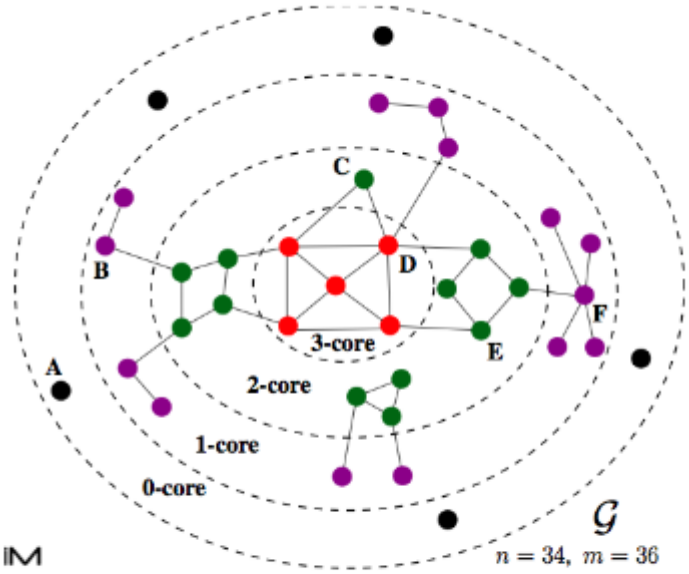

FIGURE 2.6 – Illustration de la *k-core décomposition*

Chapitre 3

Les bases de connaissance

Une base de connaissance est une ontologie instanciée. C'est-à-dire peuplée.
L'objet de cette partie est de détailler une méthode de création d'une ontologie inspirée de l'article [3]. Nous présenterons les grands principes derrière les méthodes de création d'une ontologie, puis nous rappellerons l'idée générale ainsi que les contraintes majeures. Ensuite, nous nous attarderons sur les concepts que nous exploiterons dans la méthode développée en partie II de ce document.

3.1 Les méthodes de représentation des documents du corpus

Une des difficultés lors de la création d'une ontologie consiste à définir la granularité de celle-ci. Faut-il instancier des *classes* déjà existantes, ou bien doit-on enrichir l'ontologie avec de nouvelles *classes* ? Nathalie Aussenac-Gilles[9] rappelle l'importance de fixer le paradigme d'évaluation pour pouvoir répondre à cette question. De manière générale, c'est surtout l'usage qui permet de trancher entre un paradigme cognitif (concept défini pour représenter une entité mentale tel qu'un raisonnement), linguistique (s'appuyant sur les différences véhiculées par la langue), philosophique, voire pragmatique[9]. C'est à travers ce prisme d'interprétation que la sélection de ressources externes, de corpus orthogonaux et tous les choix techniques pourront s'effectuer.

3.1.1 La structuration de la connaissance

On distingue généralement deux approches pour structurer la connaissance. **L'approche ascendante** qui va tenter de discerner un modèle conceptuel dans les documents. Et **l'approche descendante** qui part d'un modèle pré-existant pour reconnaître des instanciations dans les documents.

Dans la pratique, il s'agit généralement d'une combinaison (méthode mixte) de ces deux approches.

3.1.2 L'analyse des documents

L'approche statistique consiste à modéliser les concepts et les relations en se basant sur la fréquence des mots et leur distribution au sein des documents. Plusieurs problèmes se posent :

— l'identification des mots vides [1],

— la gestion des mots composés.

L'approche statistique est très efficace pour mettre en évidence la distribution d'un vocabulaire au sein d'un corpus. Mais il est souvent compliqué de lui attacher une sémantique.

L'approche linguistique quant à elle consiste à s'appuyer sur une connaissance précise d'une langue, et notamment des règles syntaxiques de dérivation, pour identifier des *patterns* (schémas) sémantiques. Ainsi, il est possible d'identifier les verbes d'action, le sujet, l'objet. L'inconvénient de cette démarche est la lourdeur de sa mise en oeuvre. Elle nécessite une expertise de la langue et un travail préparatoire conséquent pour développer les outils d'analyse. De plus, la complexité du langage naturel ne nous garantit pas que la totalité de l'information ait été traitée par les schémas conceptuels pré-établis.

Méthode mixte : À l'usage, les méthodes sont généralement combinées pour extraire des documents à la fois une représentation statistique et sémantique.

Le support des ressources externes permet, de plus, de préciser et d'enrichir la connaissance ainsi structurée. En effet, après avoir dégagé une typographie (linguistique ou sémantique) du texte, l'usage de ressources externes telles que DBmedia [2], WordNet [3] permet de préciser la sémantique des mots analysés statistiquement, ou les relations repérées par des schémas linguistiques.
L'usage de ressources externes est l'illustration d'un procédé *descendant*.

3.2 Les étapes de la structuration

Nous allons ici détailler les étapes de l'indexation par une approche statistique que nous appuierons sur des ressources externes. Nous nous référerons majoritairement à l'article [3].

1. Un mot vide est un mot non discriminant. C'est à dire très répandu et pas suffisamment spécialisé. Les jours de la semaine, par exemple.
2. Ontologie créée en 2007 par les travaux de l'université de Leibniz basés sur les données de Wikimedia.`https://wiki.dbpedia.org/`
3. Ontologie linguistique développée par l'université de Princeton.`https://wordnet.princeton.edu/`

Les étapes de l'indexation :

1. extraction des mots pleins
2. extraction des termes composés
3. pondération
4. désambiguïsations
5. extraction des relations

3.2.1 Extraction des mots pleins et pondération

Comme nous l'avons défini antérieurement, un mot plein est un mot pertinent pour la ségrégation du document. Nous allons réutiliser les travaux de [3] et [13], leur méthode étant particulièrement adaptée pour les corpus spécialisés.
Soit deux corpus spécialisés de domaines orthogonaux (par exemple, mécanique et agriculture) A et B et leur vocabulaire associé V_a et V_b, alors le vocabulaire V_v est l'ensemble des mots présents dans les deux vocabulaires. C'est l'ensemble des mots vides :

$$V_v = \{V_a \cap V_b\}$$

On appelle V_p le vocabulaire des mots pleins du corpus étudié (V_a). Il correspond à l'ensemble des mots de V_a non vides : $V_p = V_a \backslash V_v$. Pour affiner ce vocabulaire, et ne pas être trop restrictif dans l'exclusion des mots conjoints aux deux vocabulaires, mais apportant du sens, les auteurs de [3] utilisent la loi de Zipf[14] qui stipule que plus un mot est court, plus il est fréquent. Ainsi, l'on ajoute au vocabulaire V_v les mots de $\{V_a \cap V_b\}$ n'étant ni trop courts, ni trop fréquents.

Le concept d'information mutuelle : pour la sélection des mots composés, il est important de présenter le concept d'information mutuelle. Il s'agit d'essayer de déterminer la dépendance statistique entre des variables aléatoires [4].
Pour deux variables aléatoires X et Y, l'*Information Mutuelle* IM est définie par

$$IM(X,Y) = \sum_{(x,y)\in(X\times Y)} P(X=x, Y=y) \log \frac{P(X=x, Y=y)}{P(X=x) \times P(Y=y)}$$

Si deux variables aléatoires sont indépendantes, IM vaut 0 et elle augmente avec la corrélation. L'objectif est donc de maximiser l'information mutuelle. Ce qui signifie trouver les mots les plus liés les uns au autres qui formeront les mots composés.
Pour ce faire, on remarque qu'il suffit de maximiser la valeur :

$$\log \frac{P(X=x, Y=y)}{P(X=x) \times P(Y=y)} \tag{3.1}$$

4. Ici des mots

Cependant, les auteurs de [3] pointent que les mots *vides* bruitent le résultat. En effet, leur fréquence est telle que leur présence dans des mots composés est négligeable. Ils donnent l'exemple de *"laboratoire de"*. Même si la co-occurrence des mots "laboratoire" et "de" est élevée dans les documents, la fréquence du mot "de" sera probablement bien plus grande. Ainsi, avec cette mesure, le mot composé ne sera pas retenu. C'est pourquoi ils proposent ce qu'ils appellent *l'Information Mutuelle Adaptée* IMA en modifiant légèrement (3.1) qui devient :

$$(3.1) = \begin{cases} \log \frac{P(X=x,Y=y)}{P(X=x)\times P(X=x)} & Si\ y\ est\ un\ mot\ vide \\ \log \frac{P(X=x,Y=y)}{P(X=x)\times P(Y=y)} & Sinon \end{cases} \tag{3.2}$$

Ainsi, on a :

$$IMA(X,Y) = \sum_{(x,y)\in(X\times Y)} P(X=x,Y=y)\times(3.2) \tag{3.3}$$

Ce qui vaut :

$$IMA(X,Y) = \sum_{(x,y)\in(X\times Y)} P(X=x,Y=y)\times \begin{cases} \log \frac{P(X=x,Y=y)}{P(X=x)\times P(X=x)} & Si\ y\ est\ un\ mot\ vide \\ \log \frac{P(X=x,Y=y)}{P(X=x)\times P(Y=y)} & Sinon \end{cases} \tag{3.4}$$

Ainsi, on ne pondère pas l'indice d'information mutuelle par les mots préalablement identifiés comme *vides*. On y substitue la fréquence du mot non vide.

La pondération des mots ainsi extraits est le plus souvent effectuée à l'aide de la formule $TF * IDF$ [5][15]. Il s'agit de pondérer les occurrences en fonction de leur distribution au sein du document et au sein du corpus. Ainsi, cette mesure se décompose en une mesure de la fréquence des termes TF et une mesure de la distribution de ces termes au sein des documents du corpus IDF. *Inverse* parce que l'on pondère le score par le nombre de documents contenant le terme. En effet, on considère qu'une occurrence est d'autant plus pertinente qu'elle est spécifique à un document.
Il y a différentes manières de calculer TF et IDF. Nous présentons la variante normalisée :

$$TF * IDF_{i,j} = 0,5 + 0,5 \times \frac{tf_{i,j}}{tf_{i,j} + 0,5 + 1,5 * \frac{dl_j}{\Delta l}} * \frac{\log \frac{N+0,5}{n_i}}{\log N + 1} \tag{3.5}$$

avec :

— N : le nombre total de documents dans le corpus,
— n_i : le nombre de documents contenant le terme i,
— $tf_{i,j}$: la pondération locale du terme i dans le document j,
— Δl : la longueur moyenne des documents du corpus en nombre de mots.

5. D'après l'anglais, *term frequency-inverse document frequency*

Cependant, les auteurs de [3] précisent qu'un terme composé est plus riche sémantiquement que les termes qui les composent. Ainsi, pour conserver cette distinction, il faut en tenir compte dans la pondération. C'est pourquoi ils proposent une variante du $TF * IDF$ qu'ils nomment le $CTF * IDF$:

$$CTF * IDF_{i,j} = (1 - \frac{1}{longueur(i)}) + TF * IDF_{i,j} + \frac{1}{longueur(i)} * \sum_{k \in i} TF * IDF_{k,j} \quad (3.6)$$

où i représente l'occurrence étudiée, ce qui peut être un groupe de mots (pour les mots composés), ou bien un singleton. Ainsi, lorsque l'on ne considère qu'un mot, on retrouve le $TF * IDF$ classique.

3.2.2 Extraction des concepts à l'aide de ressources sémantiques externes

Cette partie est importante car c'est le coeur du travail de sémantisation des documents. On va décrire une méthode classique d'usage de ressources sémantiques externes pour appareiller des concepts au vocabulaire précédemment extrait. Nous continuons de nous appuyer sur l'article "indexation sémantique des documents multilingues" [3] qui décrit très bien cette méthode.

Nous avons défini en (2.3.2) le formalisme d'une ontologie, et nous avons construit en (2.3.2) les propriétés nécessaires pour leur manipulation. C'est donc en s'appuyant sur ce formalisme que nous allons détailler cette section.

On entend par élément du vocabulaire tout mot ou groupe de mots (mots composés) identifiés comme non vides dans le vocabulaire du corpus étudié. Nous l'avons noté V_p en (3.2.1).

$$\forall x \in V_p, on\ cherche\ y \in V_{OC} \mid y = S_c(x) \quad (3.7)$$

On rappelle que V_{OC} représente le vocabulaire des concepts de l'ontologie O.

Deux cas d'ambiguïtés peuvent apparaître :

- **Langagière** lorsque des termes appartenant à des langues différentes ont la même syntaxe.
- **sémantique** lorsqu'un mot possède des sens différents.

On lève l'ambiguïté langagière par l'application de (2.3) :
En effet, si l'on est confronté à une ambiguïté langagière, alors on a $card(L(t_i)) > 1$ pour t_i

le terme numéro i d'un document. Ainsi,

$$\exists \epsilon \geq n \ (n \in N) \ tel \ que \ card(L(t_{i \pm \epsilon})) = 1 \tag{3.8}$$

C'est-à-dire qu'il est possible de trouver un terme proche du terme ambigu qui ne pose pas de problème pour le langage.
Alors, on utilise le langage de ce terme pour lever l'ambiguïté.

On lève l'ambiguïté sémantique par l'application de (2.2) et (2.1).
Soit t_i un terme ambigu dans le document. On appelle C_i l'ensemble des concepts auxquels se réfère t_i. Par ailleurs, on définit la distance $d(t_x, t_i)$ d'un terme t_x à t_i par le nombre de mots le séparant de t_i. Ainsi, on cherche à minimiser $d(t_x, t_i)$ tel que :

$$S_c(t_x) \in L_c(C_i) \tag{3.9}$$

On cherche ainsi le terme le plus proche non ambigu dont le concept de référence est l'un des possibles pour le terme t_i. Cela devient alors le concept de t_i.

3.2.3 Extraction des relations à l'aide de ressources sémantiques externes

De manière similaire à l'identification des concepts, l'identification des relations est facilitée par l'usage de ressources externes. Comme dans la section précédente, nous nous référerons à (2.3.2) et (2.3.2).

L'identification des concepts composant une relation est facilitée par l'usage d'ontologies externes. Ainsi, pour chaque phrase, on considérera que deux termes sont en relation si leurs références conceptuelles partagent une relation.

Soit t_i et t_j les termes d'une même phrase. t_i et t_j partagent le concept $\leq^c$ si et seulement si

$$\exists C_i \ et \ \exists C_j \ tels \ que \ S_c(t_i) = C_i \ et \ S_c(t_j) = C_j \tag{3.10}$$

$$\exists \leq^c \ tel \ que \leq^c (C_i, C_j) \tag{3.11}$$

Il s'agit ici d'identifier au sein d'une phrase les termes référents à des concepts entre lesquels il existe une relation.

Dans cette partie, on a donc revu les grands principes de la représentation sémantique de la donnée. On a remarqué la diversité des méthodes, des statistiques aux ontologies, en passant par les réseaux de neurones. Et on a rappelé que la préoccupation de la gestion de ressources documentaires précédait grandement l'ère informatique.

Nous allons désormais appliquer les différentes techniques que nous venons de décrire pour tenter de simplifier la représentation des documents à travers l'annotation sémantique.

Deuxième partie

Application

Après avoir rappelé les concepts et les enjeux autour de la représentation sémantique de documents dans la première partie, nous allons désormais proposer une méthode d'annotation sémantique s'appuyant sur des ressources externes de manière à réduire la lourdeur des procédures. L'objectif n'étant pas d'obtenir une sémantique aussi puissante qu'une ontologie, mais une structuration suffisamment efficace pour la plupart des usages sur un corpus spécialisé, et en particulier pour de la recherche d'information.

Dans un premier temps, nous présenterons notre méthodologie, en détaillant notamment la manière dont nous utiliserons une partie des outils présentés en première partie. Nous y détaillerons également nos métriques d'évaluation. Ensuite nous exposerons nos résultats à travers l'exemple de la recherche de document. Nous en discuterons puis nous évoquerons les futurs travaux possibles dans ce domaine.

Chapitre 4

Méthodologie

4.1 Le traitement du corpus

En nous inspirant de la méthode de création d'ontologie assistée par des ressources externes exposée au chapitre 3 et proposée par [3], nous proposons de traiter le corpus selon les étapes suivantes pour chaque document :

- nettoyage de la donnée, suppression des mots en s'appuyant sur *un vocabulaire orthogonal*[1], *lemmatisation* des mots,
- extraction des mots clefs en s'appuyant sur l'algorithme du *Graph of Word* présenté en (2.6),
- indexation des concepts attachés aux mots clefs obtenus à l'aide d'une ontologie externe.

On interrogera le corpus ainsi indexé à l'aide d'un moteur de recherche très simple.

4.1.1 Nettoyage de la donnée

Le nettoyage de la donnée s'effectue principalement en deux temps. Il s'agit tout d'abord de constituer une liste de *stops words* selon la méthode d'extraction des mots vides présentée en (3.2.1), les listes de mots vides présentes dans les bibliothèques de traitement du langage telles que *NLTK* n'étant pas satisfaisantes.

Pour affiner cette liste, nous appliquons la loi de Zipfs[14] [2]. Nous prenons 5 comme seuil de longueur maximum d'un mot vide. Tout mot plus long que 5 caractères appartenant à la liste des mots vides sera considéré comme un mot plein.

1. Voir 3.2.1
2. Voir 3.2.1

De plus, les mots sont **lemmatisés**. C'est-à-dire qu'ils sont ramenés à une forme canonique. Cela permet d'ignorer les déclinaisons d'un mot, et de ne considérer que sa forme principale (canonique).
Par exemple, les mots *petit*, *petits*, *petites* et *petite* sont tous de la famille du mot *petit* qui en est la forme canonique.

4.1.2 Extraction des mots clefs

Présenté en (2.6), l'algorithme du *Graph of Word*[11] permet d'extraire le coeur le plus dense du graphe formé par les mots composant le document. C'est ce qu'on considérera comme mots clefs.

4.1.3 Indexation des concepts associés

Une fois les mots clefs extraits, l'on va s'appuyer sur une ontologie pour obtenir un concept associé.

Nous allons utiliser *ConceptNet*[3] comme ressource ontologique externe et publique, mais il en existe beaucoup d'autres tels que *DBpedia*[4], *Yago*[5] entre autres. *ConceptNet* est une ontologie qui agrège de nombreuses sources. Elle n'est pas spécialisée, et les concepts extraits sont souvent vagues. Ils ne sont pas extrêmement discriminants . Autrement dit, l'*Information Mutuelle*[6] entre les concepts issus de ConceptNet n'est pas extrêmement élevée. Cependant, cette base d'ontologie dispose d'une interface par *api* et est simple à manipuler, ce qui est suffisant pour de premières expériences.

L'apport de *ConceptNet* pour la représentation sémantique de documents, notamment en comparaison avec le *word2vec*, le *GloVe* et d'autres représentation basées sur la statistique a été mis en évidence par Robert Speer, Joshua Chin et Catherine Havasi [16].

4.2 Les outils d'évaluation

L'évaluation de notre méthode se fera par la mesure de la pertinence des résultats. Nous utiliserons le même moteur de recherche pour toutes les requêtes, et nous calculerons les scores de *précision*, de *rappel* et le *F score* sur des jeux de données étiquetées. Nous comparerons les courbes *ROC* obtenues et nous regarderons si l'annotation sémantique des documents améliore le classement des résultats.

3. `http://conceptnet.io/`
4. `https://wiki.dbpedia.org/`
5. https ://github.com/yago-naga/yago3
6. Voir 3.2.1

4.2.1 La matrice de confusion

Une grande partie des outils que nous présenterons par la suite s'appuie sur cette matrice [7]. Il s'agit d'une représentation de la qualité des résultats d'un *classifieur*. Pour construire la matrice, on considère les valeurs observées (réelles) notées Y par rapport à $\hat{Y}$, les valeurs prédites. On peut alors construire le tableau suivant :

	Valeurs de Y	
Valeurs de $\hat{Y}$	Vrais Positifs	Faux Positifs
	Faux Négatifs	Vrais Négatifs

On a donc, pour tout classifieur f :

$$Y = f(x)$$

avec x, la donnée envoyée au classifieur. Dans notre cas, il s'agit d'une requête envoyée à un moteur de recherche.

4.2.2 F score - rappel - précision

Sur la base du tableau précédent, on peut définir les scores suivants [8] :

Précision

Soit $\alpha \in \hat{Y}$ *tel que* $\alpha \in Y$, alors on appelle *précision*

$$précision = \frac{\alpha}{Y} \tag{4.1}$$

Il s'agit de mesurer la proportion de vrais positifs, c'est-à-dire de mesurer la justesse, parmi l'ensemble retourné.
Plus la *précision* se rapproche de 1, plus les résultats renvoyés correspondent aux résultats attendus.

La précision correspond au ratio : $\frac{Vrais\ Positifs}{Vrai\ Positif + Faux\ Positifs}$ présenté dans la matrice de confusion en (4.2.1).

7. Également appelée matrice de contingence
8. Une illustration se trouve en annexe C

Rappel (ou sensibilité)

Soit $\alpha \in \hat{Y}$ *tel que* $\alpha \in Y$, alors on appelle *rappel* [9] :

$$rappel = \frac{\alpha}{\hat{Y}} \tag{4.2}$$

Il s'agit de mesurer la proportion de vrais positifs parmi l'ensemble $\hat{Y}$ des documents réellement attendus.
Plus le *rappel* se rapproche de 1, moins l'algorithme se trompe dans la classification.

Le rappel correspond au ratio : $\frac{Vrais\ Positifs}{Vrai\ Positif + Faux\ Négatifs}$ présenté dans la matrice de confusion en (4.2.1).

Le score F

Le score F est défini par :

$$F = \frac{précision \cdot rappel}{précision + rappel} \tag{4.3}$$

Encore une fois, plus sa valeur est élevée, meilleur est la qualité de la classification.

4.2.3 La position dans le classement

Pour évaluer les résultats, nous regardons également l'ordre des valeurs retournées par le moteur de recherche. En effet, on s'attend à ce que les documents les plus pertinents soient dans les premiers résultats.
On va donc mesurer la distance dans l'index :

$$rank(d_i) = \frac{1}{i}$$

avec $i \in \mathbf{N}^+$ la position dans l'index et d le document.

4.3 La courbe ROC et l'AUC

4.3.1 la courbe ROC (receiver operating characteristic)

Dans son article de 2003, Tom Fawcett[17] nous rappelle le fonctionnement et le bon usage de la courbe ROC. Il s'agit, nous rappelle-t-il, d'une technique de visualisation et de sélection d'algorithmes de *classification binaire* basée sur leur performance. Il s'agit de représenter sur un graphe le *taux de vrais positifs parmi les résultats positifs* (le rappel) par rapport

9. En statistique on parle également de sensibilité

au *taux de faux positifs parmi les résultats négatifs*. Cette proportion est appelée *FP Rate* et elle vaut :

$$FP\ rate = \frac{Faux\ Positifs}{Faux\ Positifs + Vrais\ Négatifs} \tag{4.4}$$

.

Le classifieur binaire

La courbe ROC évalue un classifieur binaire. C'est une fonction qui partage une valeur entrante entre deux états (classes) possibles.

Cependant, les valeurs obtenues lors de la mesure de la performance d'un algorithme ne sont pas toujours discrètes, et encore moins binaires. C'est pourquoi, il est généralement nécessaire de répartir les résultats d'un score en deux valeurs, *accepté* ou *rejeté*, en fonction d'un *seuil* (*threshold*) θ :

$$f(x) = \begin{cases} 1 & Si\ x \geq \theta \\ 0 & Sinon \end{cases} \tag{4.5}$$

avec f une fonction d'évaluation des résultats.

la courbe

L'axe des abscisses de la courbe ROC représente le *FP Rate* tandis que l'axe des ordonnées affiche la mesure du *rappel*.

On remarque quelques points importants :

- Le point d'origine (0, 0) représente un résultat totalement classé comme négatif. Il n'y a évidemment pas de Faux Positifs.
- A l'inverse, le point (1,1) accepte tout comme Positif. Il y a donc beaucoup de Faux Positifs, mais évidemment pas de Faux Négatifs.
- Enfin, le point (0,1) représente un classification parfaite.

Cette courbe permet d'étudier la qualité d'un algorithme selon un critère d'acceptabilité des résultats.

4.3.2 l'AUC

Tom Fawcett[17] présente également *L'AUC* (*Aera under Curve*). C'est la mesure de l'aire sous la courbe ROC. Il s'agit donc d'une valeur scalaire plus aisément exploitable que la description en deux dimensions qu'est la courbe ROC. Elle permet de mesurer la performance d'un algorithme. En effet, plus la courbe ROC se rapproche du point $(1, 1)$, plus l'*AUC* sera élevé.

Nous rappelons qu'une classification aléatoire produit une courbe ROC de la forme $y = x$ avec $y = f(x)$. C'est donc une droite passant par l'origine, d'accroissement 1 pour 1. Ainsi, l'AUC, pour ce classifieur aléatoire, vaut $0,5$. Un modèle ayant une valeur d'AUC inférieure à $0,5$ sera donc considéré comme moins performant que le hasard.

4.4 La *p-value*

la p-value est la mesure de probabilité qu'un modèle statistique obtienne les même valeurs que celles observées par le hasard (Sous *hypothèse nulle*). Ainsi, plus elle est faible, moins il est probable que les résultats soient dûs au hasard.

Dans notre étude, l'*hypothèse nulle* serait que la classification soit égale à celle obtenue par un tirage aléatoire des résultats. Or, ce tirage est représenté par la diagonale de notre courbe ROC. Ainsi, plus l'AUC sera différent de $0,5$, moins l'*hypothèse nulle* pourra être retenue.

4.5 Le moteur de recherche

Il s'agit d'un point critique puisqu'il va guider l'évaluation des résultats. Cependant, ce n'est pas le sujet de ce document. On a donc choisi de s'appuyer sur un serveur *Elastic-Search* avec une configuration minimale pour les index utilisés. Seul un analyseur *simple* est activé. Les opérations effectuées lors de l'indexation des documents se limitent ainsi à une mise en minuscule des termes de chaque document. Les chiffres sont ignorés. De plus, pour éviter les approximations liées au partitionnement, nous configurons chaque index de manière identique, avec un unique *shard* et sans réplication.

ElasticSearch a cependant l'avantage d'être un outil simple et rapide à installer, ce qui permet de nombreux tests sur de nombreuses données.

On détaille la configuration utilisée en annexe D. On y précise également la structure des requêtes.

Chapitre 5

Présentation des résultats

5.1 Rappel de la procédure de traitement

On a donc traité le corpus suivant le schéma 5.1.

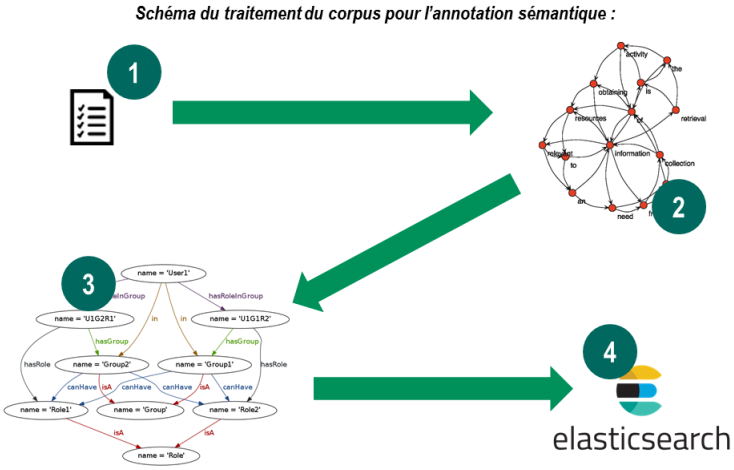

FIGURE 5.1 – **1 :** Nettoyage de la donnée - **2 :** Extraction des mots clefs - **3 :** Extraction des concepts - **4 :** Indexation dans *ElasticSearch*

Le détail du code se trouve en annexe E

5.2 Les données

Les résultats présentés ici ont été obtenus avec des données publiques issues de l'*Association of Computing Machinery*. Il s'agit d'une collection [1] d'*Abstracts* d'articles publiés dans la revue *Communications of the ACM (CACM)* entre 1958 et 1979. Les données sont donc plutôt techniques dans le domaine de l'informatique. Le jeu de donnée fourni un ensemble de questions et de fichiers attendus.

Par exemple, à la question, *"What articles exist which deal with TSS (Time Sharing System), an operating system for IBM computers ?"*, sont attendus les fichiers suivants :

— *CACM-1410,* [2]
— *CACM-1572,*
— *CACM-1605,*
— *CACM-2020,*
— *CACM-2358.*

Les données CACM sont constituées de 3204 documents pour un vocabulaire de 26869 mots.

Pour extraire les mots vides [3], on a utilisé un autre corpus spécialisé dans un domaine qui semble assez disjoint de l'informatique. Nous nous sommes appuyés sur le corpus anglais fourni par le projet *MuchMore* [4]. Il s'agit d'un projet travaillant sur une approche billinguale anglais et allemand de ressources du domaine médical. Il s'agit d'une collection d'Abstract d'articles médicaux issus du site de publications scientifiques *Springer*.

Les données de MuchMore sur le corpus anglais sont constituées de 7823 documents pour un vocabulaire de 74591 mots.

Mots vides : après disjonction des vocabulaires et application de la Loi de *Zipf* [5] avec un seuil fixé à 5, on obtient une liste de 1872 mots du corpus principal considérés comme *vides*.

Nous interrogeons **ConceptNet** pour chaque mot clef extrait avec la méthode du *Graph Of Word* [6], et nous ne conservons que deux types de relations.

1. Source : `http://www.search-engines-book.com/collections/`
2. Le contenu de ce fichier est fournit en annexe F
3. Ref 3.2.1
4. `http://muchmore.dfki.de/index.html`
5. Ref 3.2.1
6. Ref 2.6

RelatedTo nous permet d'obtenir les liens logiques entre des éléments :

$$RelatedTo(apple) = fruit$$

$$RelatedTo(lemon) = fruit$$

On obtient ainsi des relations entre des objets, *la pomme* et *le citron* étant tous les deux *un fruit*.

PartOf nous permet d'obtenir des concepts plus généraux :

$$PartOf(tree) = forest$$

Ce qui nous fournit l'information qu'un *arbre* est inclus dans le concept de *forêt*.

Avec ces deux exemples apparaît une faiblesse de la base *ConceptNet*. On remarque que la relation entre *apple* et $fruit$ n'est pas du type $PartOf$. Il s'agit d'une différence dûe à l'histoire de la base d'ontologie qui s'est appuyée sur des sources diverses ayant une structuration parfois différente. Il s'agit d'un défaut de cette ontologie qui peut apporter du bruit dans les résultats.

5.3 Résultats

L'application de notre méthode sur cet échantillon donne les résultats suivants :

5.3.1 Les scores de rappel, précision et F

méthode	précision	rappel	F_score	score position
annotation sémantique	0.30	0.31	0.15	0.75
standard	0.26	0.28	0.14	0.62

On observe une amélioration par rapport à la méthode standard de recherche de document de

15%	*pour la précision*
11%	*pour le rappel*
7%	*pour le F_score*
21%	*pour le score de position*

5.3.2 La courbe ROC et l'AUC

Pour évaluer la performance propre de notre modèle, nous calculons sa *courbe ROC* ainsi que son *AUC*. Pour cela, comme décrit en (4.3), nous avons dû choisir un classifieur binaire

d'appréciation des résultats que l'on va qualifier comme *acceptés* ou *rejetés*. Nous avons donc établi une fonction attribuant un score aux résultats, et un seuil au delà duquel la valeur était retenue.

Soient R_k les résultats renvoyés par le moteur de recherche pour la requête k.
On considère les résultats comme valides si :
Il y a au moins deux valeurs attendues parmi les dix premiers résultats de la recherche.

De plus, nous attribuons un score S aux résultats.
Soit l'intersection C des valeurs renvoyées par le modèle et des valeurs attendues, on définit le score de classification S par :

$$S = \frac{card(C)}{min(card(Valeurs\ Attendues), 10)} \tag{5.1}$$

En effet, nous considérons comme valable les 10 premiers résultats seulement.

On obtient les courbes *ROC* en figure 5.2.

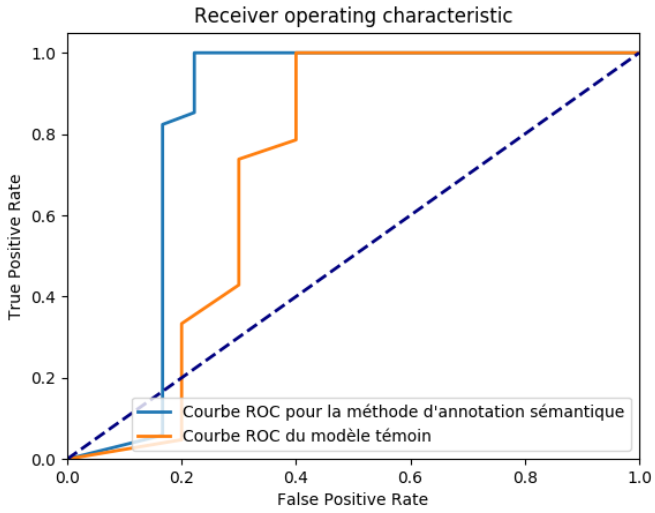

FIGURE 5.2 – Courbes ROC obtenues sur les données testées.

Les valeurs AUC associées sont **0,83** pour la méthode d'annotation sémantique et **0,72** pour la recherche standard.

On note pour les deux méthodes une distance très nette par rapport au hasard. De plus, on remarque une légère augmentation de la qualité des résultats avec l'annotation sémantique. Il y a notamment moins d'erreurs de *rappel*.

5.4 Discussion et limites

On observe une augmentation sensible de la qualité de *l'information retrieval* avec l'usage de l'annotation sémantique. Cependant, comme toutes les méthodes de traitement du langage, la qualité du résultat dépend grandement de celle des données. Il faut prendre ces résultats avec prudence, bien que l'intérêt de la méthode semble tout de même démontré.

On va donc discuter des différents biais observés.

La qualité du corpus influe évidemment énormément sur les résultats. Des documents mal structurés, pauvres en information, ne vont pas être très signifiants.

La qualité de la base d'ontologie support est primordiale. Certains corpus spécifiques ne trouvent pas leur correspondance conceptuelle dans des bases d'ontologie généraliste. Cela influe énormément sur l'exploitation du document annoté. La qualité des relations est également très importante. L'ambiguïté relevée en (5.2) va par exemple bruiter la qualité de l'annotation sémantique. De la même manière, on remarque que le concept de $fruit$ dans *ConceptNet* est en relation avec *banana* via : $IsInstaceOf(banana, fruit)$ tandis que la relation avec *orange* est de la forme $PartOf(orange, fruit)$.
Cette ambiguïté sémantique dans la structure de l'ontologie bruite les résultats. En effet, plus la quantité des relations devant être prise en compte pour chaque mot clef est élevée, plus il y aura de faux positifs dans les résultats. À l'inverse, ne pas les prendre en compte ici revient à exclure des concepts qui tombent sous un même sens, mais ne s'expriment pas de la même manière dans la base d'ontologie.

La méthode d'évaluation est elle-même soumise à discussion. Il s'agit d'une méthode commode pour évaluer la qualité d'une sémantisation de documents par l'usage, en l'occurrence par la recherche. Mais c'est alors faire l'hypothèse forte que l'usage en question et l'annotation sémantique sont corrélés. De plus, bien que cette hypothèse ne semble pas déraisonnable pour l'exercice de recherche de documents, les procédures impliquées dans la recherche dépassent de loin la simple analyse sémantique. Comptent également la méthode de construction de l'index, la méthode de recherche à travers celui-ci (*N-GRAM*, *hidden Markov model*). De plus, il faut également se garder de considérer l'annotation sémantique comme une solution efficace de recherche d'information. Si l'objectif est de maximiser l'efficacité d'un outil de recherche, alors il faudra certainement combiner plusieurs outils dont l'*information retrieval* ne serait qu'une petite part.
Ainsi, la mesure de la qualité d'une annotation sémantique par l'usage d'un moteur de recherche est loin d'être évidente. Mais elle donne toutefois un indice sur l'enrichissement de l'information contenue dans un document.

Nous avons donc vu que bien qu'intéressants, les résultats en eux-même étaient à prendre avec prudence, du fait de leur forte dépendance aux données et de la méthode d'évaluation à la marge de l'annotation sémantique.

Chapitre 6

Enjeux et futurs travaux

Dans ce dernier chapitre, nous allons traiter des enjeux et des futurs travaux pour approfondir le sujet.

6.1 Enjeux

Comme l'a rappelé Nathalie Aussenac-Gilles dans son article [9] sur le sujet, l'annotation sémantique n'est pas à opposer aux ontologies ou aux autres formes de représentation des documents. Nous l'avons vu, cette technique s'appuie au contraire sur tous les outils à disposition pour enrichir et améliorer la précision de l'annotation.

Une donnée sémantiquement contextuée est une donnée compréhensible et manipulable par d'autres automates. Les usages actuels les plus importants sont les suivants.

Le Web sémantique et notamment la structuration des résultats s'appuient sur l'analyse sémantique des documents. Il s'agit de rendre possible l'interaction automatisée entre différentes ressources.

En entreprise, la structuration automatique des CVs s'appuie beaucoup sur l'analyse sémantique.

Les outils de *Name Entities Recognition (NER)* sont souvent confrontés à des ambiguïtés qu'il faut résoudre. Obtenir un contexte sémantique est une manière d'aider à la solution.

Les assistants vocaux ou textuels sont confrontés à la résolution d'ambiguïtés similaires aux outils de NER.

L'archivage et la documentation s'appuient sur de nombreuses normes d'indexations qui se doivent d'être extrêmement précises et contraignantes pour la cohérence de la documentation, mais qui sont fastidieuses à appliquer. Une annotation automatique ou semi-automatique des documents pourrait constituer une aide à la documentation.

L'inférence d'intention , la possibilité d'extraire d'un context les intentions probables d'un individu nécessite également une compréhension du cadre conceptuel. En particulier pour les interfaces Homme-Machine, les véhicules autonomes, ou les assistants.

6.2 Futurs travaux

Comme nous l'avons vu en première partie, les travaux et les résultats sur la contextualisation sémantique de ressources sont nombreux. La baisse des coûts de calcul et de stockage a permis de développer de nombreux projets, qui sont autant d'outils d'enrichissement, sur lesquels s'appuyer pour approfondir ces travaux. Ainsi, il est intéressant de continuer à explorer plus en avant l'impact des réseaux de neurones sur les représentations. Notamment transposer le travail de représentation neuronale de documents, présenté en (2.5), sur un espace conceptuel. Ainsi, en entraînant des réseaux sur des projections de concepts, on devrait être capable de classifier de manière automatique des documents dans une ontologie. Les applications de la sémantique de la donnée sont nombreuses, mais la création d'ontologies spécifiques dans le secteur professionnel est encore une tâche lourde et peu répandue. La simplification de cette tâche est un travail important qui permettra d'exploiter au mieux le potentiel de la sémantique des données.

Conclusion

Nous avons donc présenté la théorie autour de la sémantique de la donnée, et particulièrement des ontologies. Nous avons rappelé les différentes méthodes de représentation d'un document. Nous avons constaté la complexité de la création d'ontologie, et nous avons proposé une approche s'appuyant sur l'algorithme de *Graph Of Words* développé par Michalis Vazirgiannis puis un usage de ressources sémantiques externes, inspiré des travaux de Farah Harrathi, afin de produire une annotation sémantique de documents spécialisés de manière simplifiée.

Dans un second temps, nous avons appliqué cette méthode dans un exercice d'*information retrieval* et l'avons évaluée à l'aide d'un moteur de recherche. Nous avons ainsi pu mettre en évidence son intérêt tout en remarquant les limites de la recherche de documents pour l'évaluation de l'enrichissement sémantique effectué.

Annexe A

Langage et représentation - XML/RDF

Un exemple de description RDF dans le langage XML. Ici [1], la description d'un certain Eric Miller. On distingue plusieurs catégories et sous-catégories. *foaf* est un formalisme d'ontologie RDF permettant de décrire des individus et des relations entre individus.

```
<rdf:RDF>
    <foaf:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <rdf:value>Eric Miller, em@w3.org</rdf:value>
        <foaf:name>Eric Miller</foaf:name>
        <foaf:phone rdf:resource="tel:+1-(617)-258-5714" />
        <foaf:mbox rdf:resource="mailto:em@w3.org" />
        <foaf:nick>em</foaf:nick>
        <foaf:img rdf:resource="http://www.w3.org/People/EM/s000782.JPG" />
        <foaf:workInfoHomepage rdf:resource="http://www.w3.org/People/EM"
            >
        <foaf:workplaceHomepage rdf:resource="http://www.w3.org/" />
        <contact:office>
            <contact:contactLocation>
            <rdf:value>MIT CSAIL</rdf:value>
            <contact:homePage rdf:resource="http://csail.mit.edu/" />
                <contact:address>
                    <contact:Address>
                    <rdf:value>The Stata Center, Building 32-G516, 32
                        Vassar Street, Cambridge MA 02139</rdf:value>
                    <contact:city>Cambridge</contact:city>
                    <contact:country>USA</contact:country>
                    <contact:postalCode>02139</contact:postalCode>
                    <contact:street>The Stata Center, Building 32-G516, 32
                        Vassar Street</contact:street>
```

1. Source W3C : `https://www.w3.org/People/EM/contact#me`

```
22                      <loc:coordinates>42.361860,-71.091840</loc:coordinates
                            >
23                      </contact:Address>
24                  </contact:address>
25              </contact:contactLocation>
26          </contact:office>
27          <foaf:knows rdf:resource="http://www.w3.org/People/Berners-Lee/
                card#i"/>
28          <foaf:knows rdf:resource="http://www.w3.org/People/Connolly/#me"/>
29          <foaf:knows rdf:resource="http://www.w3.org/People/djweitzner/
                public/foaf.rdf#DJW"/>
30      </foaf:Person>
31      <rdf:Description rdf:about="http://dig.csail.mit.edu/data#DIG">
32          <rdf:value>Decentralized Information Group</rdf:value>
33      <foaf:member rdf:resource="http://www.w3.org/People/EM/contact#me"/>
34      </rdf:Description>
35      <rdf:Description rdf:about="http://www.w3.org/People/Connolly/#me">
36          <rdf:value>Dan Connolly</rdf:value>
37      <rdfs:seeAlso rdf:resource="http://www.w3.org/People/Connolly/home-
            smart.rdf"/>
38      </rdf:Description>
39      <rdf:Description rdf:about="http://www.w3.org/People/Berners-Lee/card#
            i">
40          <rdf:value>Tim Berners-Lee</rdf:value>
41      <rdfs:seeAlso rdf:resource="http://www.w3.org/People/Berners-Lee/card"
            />
42      </rdf:Description>
43      <rdf:Description rdf:about="http://www.w3.org/People/djweitzner/public
            /foaf.rdf#DJW">
44          <rdf:value>Danny Weitzner</rdf:value>
45      <rdfs:seeAlso rdf:resource="http://www.w3.org/People/djweitzner/public
            /foaf"/>
46      </rdf:Description>
47 </rdf:RDF>
```

Annexe B

XML et Graphes

On voit sur cette image les liens entre un document xml et un arbre.

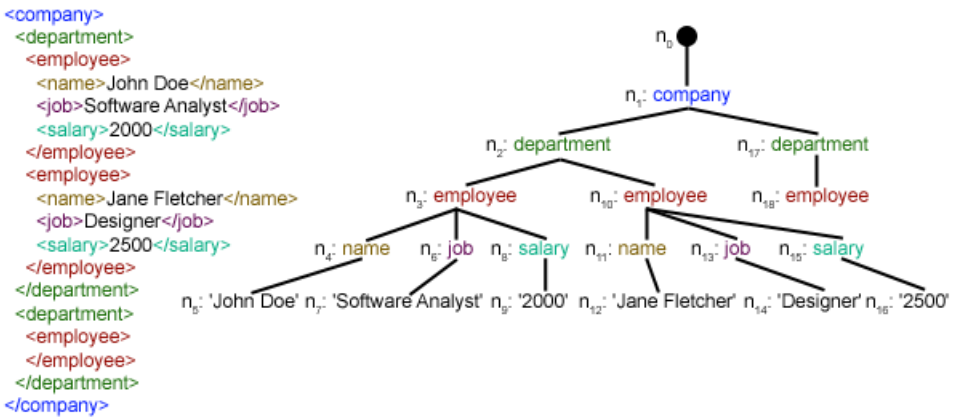

FIGURE B.1 – Liens entre Graphes et XML.
Source : `https://www.geeksforgeeks.org/xml-parsing-python/`

Annexe C

Schéma détaillant les concepts de précision et rappel

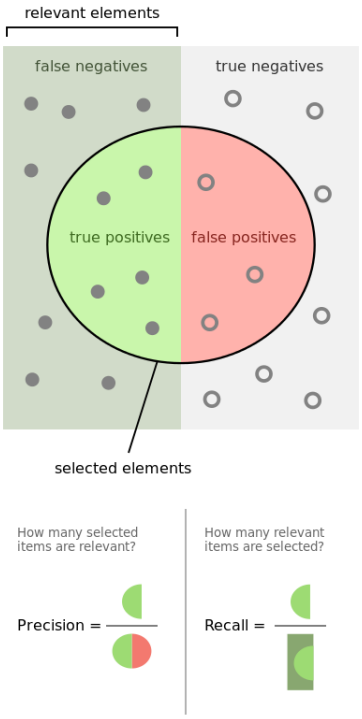

FIGURE C.1 – Source : `https://commons.wikimedia.org/wiki/File:Precisionrecall.svg`

Annexe D

Configuration d'*ElasticSearch*

D.1 Configuration des indexes

```
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 0
  },
  "mappings": {
    "medas_classique": {
      "properties": {
        "content": {
          "type": "text",
          "analyzer": "simple"
        },
        "file_name": {
          "enabled": false
        }
      }
    }
  }
}
```

```
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 0
  },
  "mappings": {
    "medas": {
      "properties": {
        "concepts": {
          "type": "text",
          "analyzer": "simple"
        },
        "keywords": {
          "type": "text",
          "analyzer": "simple"
        },
        "doc": {
          "type": "text",
          "analyzer": "simple"
        },
        "file_name": {
          "enabled": false
        }
      }
    }
  }
}
```

FIGURE D.1 – Configuration des index. Le témoin à gauche et celui enrichi par annotation sémantique à droite

D.2 Structure des requêtes

FIGURE D.2 – Structure des requêtes envoyées au moteur de recherche. À gauche, la requête témoins, à droite, la requette de l'index annoté sémantiquement.

Le champ *question* représente la question fournie au moteur de recherche après traitement. On remarque que pour la version annotée, on valorise les concepts et les mots clefs.

Annexe E

Détail d'implémentation du code

E.1 Nettoyage de la donnée

```
1  import codecs
2  import os
3  from os.path import join
4
5  from nltk.stem import WordNetLemmatizer
6  from tqdm import tqdm
7
8  # input data folder location :
9  INPUT_DATA_FLD = join(os.getcwd(), 'raw_data')
10
11 ORTHO_DATASET = join(os.getcwd(), 'ortho_data')
12
13 # output data folder location
14 OUTPUT_DATA_FLD = join(os.getcwd(), 'cleaned_data')
15
16 # empty world until THRESHOLD :
17 THRESHOLD = 5
18
19 # empty word list file
20 EMPTY_WORD_PATH = join(os.getcwd(), "empty_word_list")
21
22
23 def read_doc(path):
24     with codecs.open(path, "r", encoding="utf-8", errors="ignore") as fd:
25         return fd.read().lower()
26
27
```

```
28 def write_doc(path, string):
29     with open(path, "w") as fd:
30         fd.write(string)
31
32
33 def main():
34     """
35     Nettoie la donn e des documents contenus dans INPUT_DATA_FLD:
36     1) Lemmatisation de la donn e
37     2) Exclusion des mots vides extrait     l'aide du corpus orthogonal du
           dossier
38     ORTHO_DATASET. Le THRESHOLD d fini la longueur maximale des mots
           vides pour
39     respecter la loi de Zipf
40     3) ecriture des documents ainsi nettoy s dans le dossier
           OUTPUT_DATA_FLD
41     """
42     lemmatizer = WordNetLemmatizer()
43
44     lst_dir_main = [x for x in os.listdir(INPUT_DATA_FLD) if
45                     os.path.isfile(join(INPUT_DATA_FLD, x))]
46
47     lst_dir_ortho = [x for x in os.listdir(ORTHO_DATASET) if
48                      os.path.isfile(join(ORTHO_DATASET, x))]
49
50     lst_word_main = list()
51     lst_word_ortho = list()
52
53     # Lecture des documents
54     for doc_name in tqdm(lst_dir_main):
55         doc = read_doc(join(INPUT_DATA_FLD, doc_name))
56         if len(doc) > 0:
57             lst_word_main.extend(doc.split())
58         lst_word_main = list(set(lst_word_main))
59
60     # Creation de la liste des mots vides
61     for doc_name in tqdm(lst_dir_ortho):
62         doc = read_doc(join(ORTHO_DATASET, doc_name))
63         if len(doc) > 0:
64             lst_word_ortho.extend(doc.split())
65         lst_word_ortho = list(set(lst_word_ortho))
66
67     empty_word = [x for x in tqdm(lst_word_main) if
68                   x in lst_word_ortho and len(x) < THRESHOLD]
69     write_doc(EMPTY_WORD_PATH, ";".join(empty_word))
70
71     # Ecriture des documents nettoy s
72     for doc_name in tqdm(lst_dir_main):
73         doc = read_doc(join(INPUT_DATA_FLD, doc_name))
```

```
74          if len(doc) > 0:
75              tmp = doc.split()
76              doc = " ".join(
77                  [lemmatizer.lemmatize(x) for x in tmp if
78                   x not in empty_word and "." not in x and "----" not in x])
79              write_doc(join(OUTPUT_DATA_FLD, doc_name), doc)
80
81
82 if __name__ == '__main__':
83     main()
```

E.2 Extraction des mots clefs

```
1  import json
2  import os
3  from os.path import join
4
5  from tqdm import tqdm
6
7  # data source : cleaned data
8  DATA_SOURCE = join(os.getcwd(), "cleaned_data")
9
10 # output dest
11 OUTPUT_DEST = join(os.getcwd(), "keywords_folder")
12
13
14 def read_data(path):
15     with open(path, "r") as fd:
16         return fd.read().lower()
17
18
19 def write_data(dest_path, data):
20     with open(dest_path, "w") as fd:
21         fd.write(data)
22
23
24 def find_word_idx(lst_tuple, word):
25     for idx, tpl in enumerate(lst_tuple):
26         if word == tpl[0]:
27             return idx
28     return len(lst_tuple)
29
30
31 def inverse_dic(dic):
32     return {v: k for k, v in dic.items()}
33
34
```

```
35 def count_word(data_dic):
36     """
37     calcule le degr  de chaque noeud
38     """
39     ret_dic = dict()
40     for word1, word2 in data_dic.keys():
41         if not word1 in ret_dic.keys():
42             ret_dic[word1] = 0
43         ret_dic[word1] += 1
44         if not word2 in ret_dic.keys():
45             ret_dic[word2] = 0
46         ret_dic[word2] += 1
47     return ret_dic
48
49
50 def graph_of_word(data):
51     """
52     Chaque mot du document est index  sur la base d'une relation avec le
           mot
53     suivant. Le nombre de couple de mots est incr ment .
54     N'est conserv s que les couples de mots ayant le maximum de relation.
55     (il S'agit des noeuds au degr  le plus  lev  )
56     """
57     ret_dic = dict()
58     ret_list = list()
59     data_lst = data.split()
60     data_len = len(data_lst)
61     i = 0
62     while i < data_len - 1:
63         word = data_lst[i]
64         word_next = data_lst[i + 1]
65         if (word, word_next) in ret_dic.keys():
66             ret_dic[(word, word_next)] += 1
67         elif (word_next, word) in ret_dic.keys():
68             ret_dic[(word_next, word)] += 1
69         else:
70             ret_dic[(word, word_next)] = 1
71         i += 1
72
73     deg_dict = count_word(ret_dic)
74     if len(deg_dict.values()) > 0:
75         max_degree = max(deg_dict.values())
76         new_dic = {k: v for k, v in deg_dict.items() if v >= max_degree}
77     else:
78         new_dic = {k: v for k, v in deg_dict.items()}
79     for x in new_dic.keys():
80         ret_list.append(x)
81     ret_list = set(ret_list)
82     return ret_list
```

```
83
84
85 def main():
86     """
87     Extractions des mots clefs suivant la m thode du Graph Of Words.
88     """
89     os.makedirs(OUTPUT_DEST, exist_ok=True)
90     lst_doc = [x for x in os.listdir(DATA_SOURCE) if
91                os.path.isfile(join(DATA_SOURCE, x))]
92
93     for doc_name in tqdm(lst_doc):
94         # Lecture des documents bas s sur la source DATA_SOURCE
95         doc = read_data(join(DATA_SOURCE, doc_name))
96         data = {
97             # Extraction des mots clefs
98             "keywords": list(graph_of_word(doc)),
99             # Documents initial
100            "doc_words": doc.split()
101            }
102        # Ecriture du r sultat dans le dossier OUTPUT_DEST
103        write_data(join(OUTPUT_DEST, doc_name), json.dumps(data))
104
105
106 if __name__ == '__main__':
107     main()
```

E.3 Extraction des concepts

```
1 import hashlib
2 import json
3 import os
4 from os.path import join
5
6 import elasticsearch
7 import elasticsearch.helpers
8 import requests
9
10 # Source keyword files
11 SOURCE_PATH = join(os.getcwd(), "keywords_folder")
12
13 # indexation
14 ES_HOST = "elasticsearch address"
15 ES_TYPE = "medas"
16 ES_INDEX = "medas"
17
18
19 def read_file(path):
```

```
20      with open(path, "r") as fd:
21          return fd.read().lower()
22
23
24  def is_number(s):
25      try:
26          float(s)
27          return True
28      except ValueError:
29          pass
30
31      try:
32          import unicodedata
33          unicodedata.numeric(s)
34          return True
35      except (TypeError, ValueError):
36          pass
37      return False
38
39
40  def get_concept(word):
41      concept_lst = list()
42      obj = requests.get('http://api.conceptnet.io/c/en/{}'.format(word)).
            json()
43      for node in obj["edges"]:
44          if node["rel"]["@id"] == "/r/RelatedTo" or node["rel"][
45              "@id"] == "/r/PartOf":
46              concept_lst.append(node["end"]["label"])
47      return concept_lst
48
49
50  def main():
51      """
52      Recup re les concepts "RelatedTo' et 'PartOf' des mots clef de
            chaques documents
53      en appelant l'api ConceptNet.
54      Ensuite envoie le document, les mots clefs et les concepts extraint
            dans un index
55      elasticsearch
56      :return:
57      """
58      es_client = elasticsearch.Elasticsearch(ES_HOST)
59
60      list_source = [x for x in os.listdir(SOURCE_PATH) if
61                     os.path.isfile(join(SOURCE_PATH, x))]
62      get_id_query = {
63          "_source": '_id',
64          "query": {
65              "match_all": {}
```

```
66              }
67          }
68      id_list = list()
69      for ret in elasticsearch.helpers.scan(es_client, index=ES_INDEX,
70                                            doc_type=ES_TYPE, query=
                                                  get_id_query):
71          id_list.append(ret["_id"])
72      # Le hash du nom de fichier sert de clef unique dans lasticsearch
73      list_source = [x for x in list_source if
74                     hashlib.sha224(x.encode('utf8')).hexdigest() not in
                           id_list]
75      for num, file_name in enumerate(list_source):
76          if num % 2 == 0:
77              concept_dict = dict()
78          concept_list_inter = list()
79          concept_list_disjoint = list()
80          doc_id = hashlib.sha224(file_name.encode('utf8')).hexdigest()
81          if doc_id in id_list:
82              continue
83          # lecture du document
84          doc = json.loads(read_file(join(SOURCE_PATH, file_name)))
85          keywords = doc["keywords"]
86          for word in keywords:
87              # r cup re les concepts associ s aux mots qui ne sont pas
                    des nombres
88              if not is_number(word) and len(
89                      [x for x in ["%"] if x in word]) == 0:
90                  concept_dict[word] = list(set(get_concept(word)))
91
92          key_list = list(concept_dict.keys())
93          size = len(key_list)
94
95          # poour chaque concepts r cup r . Extrait ceux partag s entre
                les mots
96          # de ceux qui ne le sont pas. On a ainsi une liste de concepts
                disjoints
97          # et une de concepts partag s. Pour une ventuelle pond ration
                des termes
98          for idx in range(size):
99              i = 0
100             set_word = set(concept_dict[key_list[idx]])
101             while i < size:
102                 if i != idx:
103                     concept_list_inter.extend(
104                         set_word.intersection(set(concept_dict[key_list[i
                                ]])))
105                 i += 1
106         concept_list_inter = list(set(concept_list_inter))
107
```

```
108         for idx in range(size):
109             i = 0
110             set_word = set(concept_dict[key_list[idx]])
111             while i < size:
112                 if i != idx:
113                     concept_list_disjoint.extend(
114                         set_word.union(set(concept_dict[key_list[i]])))
115                 i += 1
116         concept_list_disjoint = list(
117             set(concept_list_disjoint) - set(concept_list_inter))
118 
119         data = {
120             "file_name": file_name,
121             "concepts": list(set(concept_list_inter +
                    concept_list_disjoint)),
122             "keyword": keywords,
123             "doc": doc["doc_words"]
124             }
125         # envois des r sultats dans ElasticSearch
126         es_client.index(ES_INDEX, ES_TYPE, id=doc_id, body=data)
127 
128 
129 if __name__ == '__main__':
130     main()
```

Annexe F

Une exemple de fichier CACM

On présente ici le contenu du fichier *CACM-1410* de la base CACM.

```
1  Interarrival Statistics for Time Sharing Systems
2
3  The optimization of time-shared system performance
4  requires the description of the stochastic
5  processes governing the user inputs and the program activity.
6  This paper provides a statistical description
7  of the user input process in the SDC-ARPA general-purpose
8  Time-Sharing System (TSS).  The input process
9  is assumed to be stationary, and to be defined by the
10 interarrival time distribution.  The data obtained
11 appear to justify satisfactorily the common assumption
12 that the interarrival times are serially independent.
13 The data do not appear to justify, except as a very
14 rough approximation, the usual assumption off an
15 exponential distribution for interarrival time.  A much
16 more satisfactory approximation to the data can
17 be obtained with a biphase or triphase hyperexponential distribution.
18
19 CACM July, 1966
20
21 Coffman, E. G.
22 Wood, R. C.
23
24 CA660704 JB March 2, 1978  9:45 PM
25
26 1410       5              1410
27 1410       5              1410
28 1410       5              1410
```

```
29 1604        5               1410
30 1951        5               1410
31 2373        5               1410
32 1224        6               1410
33 1410        6               1410
34 1410        6               1410
35 1410        6               1410
36 1604        6               1410
37 1751        6               1410
38 1810        6               1410
39 1951        6               1410
40 2374        6               1410
```

Bibliographie

[1] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove : Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[3] Farah Harrathi, Catherine Roussey, Sylvie CALABRETTO, Loïc Maisonnasse, and Mohamed Mohsen Gammoudi. Indexation sémantique des documents multilingues. In INFORSID, editor, *27ème Congrès INFORSID*, pages 31–50, Toulouse, France, May 2009.

[4] Kuyoro Shade O., Ibikunle Frank A, and Abel Samuel B. Information retrieval : An overview. *International Journal of Advanced Research in Computer Science*, 3(5), Sept-Oct 2012.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5) :34–43, May 2001.

[6] Khadim Drame. *Contribution to ontology building and to semantic information retrieval : application to medical domain.* Theses, Université de Bordeaux, December 2014.

[7] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *International Journal of Human-Computer Studies*, pages 907–928. Kluwer Academic Publishers, 1993.

[8] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering : Principles and methods. *Data Knowl. Eng.*, 25 :161–197, 1998.

[9] Nathalie Aussenac-Gilles. Donner du sens à des documents semi-structurés : de la construction d'ontologies à l'annotation sémantique. In Lisette Calderan, Pascale Laurent, Hélène Lowinger, and Jacques Millet, editors, *Le document numérique à l'heure du web*, Sciences et techniques de l'information, pages 105–140. ADBS, 2012. Chapitre 05 : Donner du sens à des documents semi-structurés : de la construction d'ontologies à l'annotation sémantique.

[10] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology ? pages 1–17, 05 2009.

[11] François Rousseau and Michalis Vazirgiannis. Main core retention on graph-of-words for single-document keyword extraction. In Allan Hanbury, Gabriella Kazai, Andreas Rauber, and Norbert Fuhr, editors, *Advances in Information Retrieval*, pages 382–393, Cham, 2015. Springer International Publishing.

[12] Zellig S. Harris. Distributional structure. *WORD*, 10(2-3) :146–162, 1954.

[13] Loïc Maisonnasse, Catherine Roussey, Sylvie Calabretto, and Farah Harrathi. Approche statistique versus approche linguistique pour l'indexation sémantique des documents multilingues. *Document numérique*, 14(2) :193–214, 2011.

[14] George K. Zipf. *The Psycho-Biology of Language. An Introdution to Dynamic Philology.* The MIT Press, 1965.

[15] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5) :513–523, August 1988.

[16] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5 : An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975, 2016.

[17] T. Fawcett. Roc graphs : Notes and practical considerations for researchers. Technical report, HP Laboratories, 2004.