Table des matières

Table des figures

Liste des tableaux

Glossaire

Français

AS Annotation Sémantique.

EI Extraction d'Information.

EN Entités Nommées.

REN Reconnaissance des Entités Nommées

RE Recherche d'Entités.

RI Recherche d'Information.

Anglais

AP Average Precision.

API Application Programming Interface.

CQL Combined Algorithm.

CQL Content Query Language.

ER Entity Retrieval.

FA FA Algorithm.

HTTP HyperText Transfer Protocol.

IDF Inverse Document Frequency.

QA Question Answering.

GUID Globally Unique Identifier.

MAP Mean Average Precision.

Mpro Minimal Probing.

NER Named Entities Recognition.

NRA No Random Access.

RA Random Access.

REL Related Entity Finding.

RDF Ressource Description Framework.

SA Sequential Access.

TAS Typed Annotated Search.

TA Threshold Algorithm.

TF Term Frequency.

UML Unified Modeling Language.

URI Uniform Resource Identifier.

WIE Web Information Extraction.

WQA Web-based Question Ansewering.

W3C The World Wide Web Consortium.

XML Extensible Markup Language.

1

Introduction générale

Bien que simple et intuitive, la recherche d'information telle qu'elle est largement utilisée aujourd'hui, n'est pas toujours adaptée à certains besoins. Dans plusieurs domaines d'application allant des forums de discussion sur la médecine par exemple aux newsgroups en passant par les articles de journaux, les données sont organisées autour de thèmes plus ou moins structurés qui se présentent sous forme de catégories (santé publique dans le domaine médical, politique dans les journaux), ou d'entités nommées (nom d'une clinique, nom d'un politicien). Ce que les utilisateurs cherchent dans ce cas n'est pas une liste ordonnée de documents mais des informations que ceux-ci contiennent (catégories, entités) [1].

Trouver des entités à la place des documents dans le web est un axe de recherche récent dans la recherche d'information (RI). Les travaux de l'état de l'art [2, 3] motivent leurs approches de la recherche d'entités dans le contexte du web. Dans notre travail, nous nous intéressons à des domaines d'application spécifiques où la plupart des documents sont écrits autour d'entités et sont organisés par thèmes. Parmi ces domaines, nous pouvons trouver les forums de discussion, les articles de journaux, les wikinews [1], etc.

Dans le contexte de la recherche de documents scientifiques par exemple, les utilisateurs pourraient être intéressés par découvrir les documents contenant les entités relatives à une maladie particulière (entités apparaissant dans son contexte, comme les symptômes d'une maladie), ou encore, dans les sources de données historiques, les entités relatives à une guerre entre deux pays (noms de chefs d'états, dates, lieux, etc.). Les utilisateurs peuvent aussi être intéressés par les entités composées par une certaine entité (ex., entités portant le nom du *président Ahmed Benbella* : *Université Ahmed Benbella*, *Aéroport Ahmed Benbella*...). Ils peuvent également être intéressés par les documents associés à ces entités.

Nous considérons alors le problème de la recherche des entités et des documents les contenant en réponse aux requêtes des utilisateurs.

1. Wikinews est un projet de la Wikimedia Foundation visant à établir une source d'informations libre qui résume des actualités et suit un point de vue neutre. http ://fr.wikipedia.org/wiki/Wikinews

Les entités recherchées pouvant être connues ou inconnues aux utilisateurs, ces derniers devraient avoir le choix de poser leurs requêtes de différentes manières : recherche par une entité (R1E), recherche par plusieurs entités (RPE), recherche par mots clés (RMC).

1.1 Opportunités

Notre travail [4–6] est facilité par la disponibilité de vraies données dans plusieurs domaines qui nous permettent de développer nos algorithmes et notre évaluation de manière réaliste. Nous avons effectivement considéré deux sources de données différentes : le corpus 20NewsGroups [2] et le corpus Reuters-21578. Le premier contient 20 000 messages, collectés de 20 différents newsgroups (environ 1000 messages par groupe). Ce corpus a été choisi pour sa richesse en entités et la variété de thèmes (catégories) de ses documents. Le deuxième corpus est une collection de 20 000 documents textuels téléchargée du site de l'institut d'informatique et d'électronique Gaspard-Monge [3]. Cette collection de textes a été extraite de celle de Reuters-21578 [4]. Elle a été choisie pour la richesse des thèmes utilisés (chaque document est classé selon 135 catégories).

Notre travail est rendu possible par l'existence de logiciels libres pour la recherche d'information tels que Lucene [5] que nous avons utilisé dans la partie indexation des corpus. Il est connu que Lucene permet de traiter de grandes quantités de données [7], une seule machine peut facilement contenir un index de millions de documents, le passage à l'échelle peut donc être obtenu facilement [8].

Notre travail est également rendu possible par l'apparition des systèmes d'annotation automatique de documents semi ou non structurés tels qu'Open Calais [6], Alchemy API [7] et Zemanta [8]. Ces systèmes offrent des plateformes "scalables" pour analyser les

2. http ://qwone.com/ jason/20Newsgroups/
3. http ://www-igm.univ-mlv.fr/ mconstan/enseignement/m2pro/tal/tal-td2/node1.html
4. http ://www.daviddlewis.com/resources/testcollections/reuters21578/
5. Apache Software Foundation, "Lucene," http ://lucene.apache.org/.
6. http ://www.opencalais.com/calaisAPI/
7. http ://www.alchemyapi.com/
8. http ://developer.zemanta.com/

documents et permettent de rattacher des méta-données sémantiquement riches aux documents en les catégorisant en thèmes et en extrayant les entités nommées qu'ils contiennent.

Notre idée est de construire un système de recherche d'entités en nous basant sur l'annotateur, pour prendre en charge des requêtes construites par une ou plusieurs entités (Recherche par entités) ainsi que les requêtes de mots clés (Recherche d'entités par des mots clés) et retourner des entités relatives aux requêtes avec les documents résultats pour chaque entité. Nous utilisons également des outils d'extraction d'entités tels que JRCNames [9/ 10], pour détecter les variantes d'une même entité (ex. NATO et OTAN) ou encore pour résoudre les ambiguités des acronymes [11].

1.2 Défis et Contributions

Les entités nommées traitées peuvent avoir plusieurs types (Washington : ville, personne), et apparaître dans des documents appartenant à différentes catégories (Politique, Finance, Sports...). Cela nous pose deux défis majeurs : associer les entités ou mots clés constituant une requête aux différents types d'entités, et utiliser les types d'entités identifiées ainsi que les catégories des documents les contenant pour présenter les résultats le mieux possible aux utilisateurs. Ces défis se traduisent naturellement en une question qui a été abordée dans plusieurs travaux scientifiques, à savoir la diversité des résultats de requêtes.

En effet, les différentes interprétations d'une même entité (ex., ville, personne, organisation...) ainsi que les catégories des documents les contenant (ex., Politique, Médecine, Sports...), peuvent être exploitées pour diversifier les documents à retourner. Nous nous trouvons alors devant une problématique de diversification qui est différente de l'interprétation qui lui a été conférée auparavant. Jusque là, la diversité des résultats d'une requête a été formulée comme la recherche d'un ensemble de documents pertinents à la requête et qui diffèrent le plus possible les uns des autres dans leur

9. https ://ec.europa.eu/jrc/en/language-technologies/jrc-names
10. Cet outil contient les noms de plus de 205 000 entitiés distinctes : personnes, organisations...
11. Un acronyme est un sigle, formé des initiales (OTAN, ovni).

contenu. Ainsi, le problème correspond à définir un seuil minimal de pertinence et à calculer la somme des distances de contenu de paires de documents dans l'ensemble retourné (la distance pouvant par exemple correspondre au cosinus de vecteurs tf*idf des documents). Dans le contexte de notre travail, la diversité est une notion qui repose sur les types et catégories utilisés pour annoter les documents. De plus, les documents eux-mêmes sont organisés par entité rendant le traitement des requêtes plus complexe.

Il a été démontré que le problème de diversité des résultats d'une requête est NP-Complet puisqu'il s'agit de trouver un sous-ensemble divers de taille N dans un ensemble plus grand [9–12]. Plusieurs algorithmes gloutons ont alors été développés. Le plus commun consiste à trouver les N documents les plus pertinents dans une première étape. Dans une seconde étape, il s'agit de tester si le remplacement d'un des N documents par un nouveau document sûrement moins pertinent mais dont la pertinence est supérieure à un seuil, augmente la diversité de l'ensemble de N documents. Cette phase est appliquée jusqu'à ce que la pertinence des documents restant à considérer ne satisfasse plus la condition du seuil.

Dans notre travail, l'utilisation des divers types et catégories extraits par annotations et stockés préalablement dans des index nous permet de contourner la complexité du problème de diversification des documents. La pertinence sera ensuite assurée en fixant un seuil sur les documents déjà diversifiés qui ne sont pas uniques par rapport aux types ou aux catégories afin de ne pas affecter la diversité. Ainsi, la diversité est appliquée dans deux situations.

Dans la première situation, elle est appliquée sur les entités. Les entités relatives à une requête que nous nommons entités contextuelles (apparaissant dans le contexte de la requête) et les entités composées sont trouvées et retournées à l'utilisateur. Ceci constitue une forme de "diversité d'interprétations" des requêtes dans le cas où l'utilisateur ne connait pas ce qu'il recherche ou lorsqu'il connait son entité mais qu'elle compose d'autres entités (ex., l'entité *Houari boumediene* peut correspondre au président "*Houari boumediene*", à "*Université Houari boumediene*" ou à "*Aéroport Houari boumediene*").

Dans la deuxième situation, la diversité est appliquée sur les documents associés à

chaque entité retournée. Ces derniers sont sélectionnés sur la base de deux conditions : soit que le type de l'entité ou la catégorie du document est unique (comparés aux autres documents associés à la même entité), soit que leur pertinence par rapport à l'entité est au dessus d'un seuil. Dans ce dernier cas, un index sur les catégories des documents et un index sur les entités que les documents contiennent sont utilisés. Nous pouvons ainsi affirmer que c'est notre définition de la diversité qui nous permet d'indexer le traitement et de simplifier la complexité des requêtes.

A notre connaissance, ce type de diversité n'a pas été proposé dans les travaux de l'état de l'art qui sont relatifs à la recherche d'entités.

Un prototype a été réalisé pour tester la qualité et les performances de notre approche. Les résultats obtenus en utilisant ce prototype sur les deux corpus 20NewsGroups et Reuters démontrent l'efficacité de l'approche pour retourner des entités pertinentes, composées et contextuelles (entités relatives à la requête et apparaissant dans son contexte) et particulièrement dans le cas de la recherche par une entité (R1E) et par plusieurs entités (RPE). Des tests utilisateurs démontrent aussi que la diversification est bonne dans plus de 60 % des cas pour la diversification par types et plus de 40 % pour la diversification par catégories.

L'utilisation de l'API scalable et du logiciel libre Lucene facilite le traitement de grandes quantités de données et permettra l'augmentation de la taille du corpus.

1.3 Exemple de motivation

Dans le cadre de ce travail, les entités recherchées peuvent être connues lorsque l'utilisateur saisit une entité (Recherche par une Entité "R1E") ou plusieurs entités (Recherche par Plusieurs Entités "RPE") ou inconnues s'il ignore le nom de l'entité et saisit des mots clés relatifs (Recherche par Mots Clés "RMC").

Supposons que l'utilisateur *Sami* souhaite avoir des informations sur les marques de voitures. *Sami* peut poser différentes requêtes (voir la Figure 1.1).

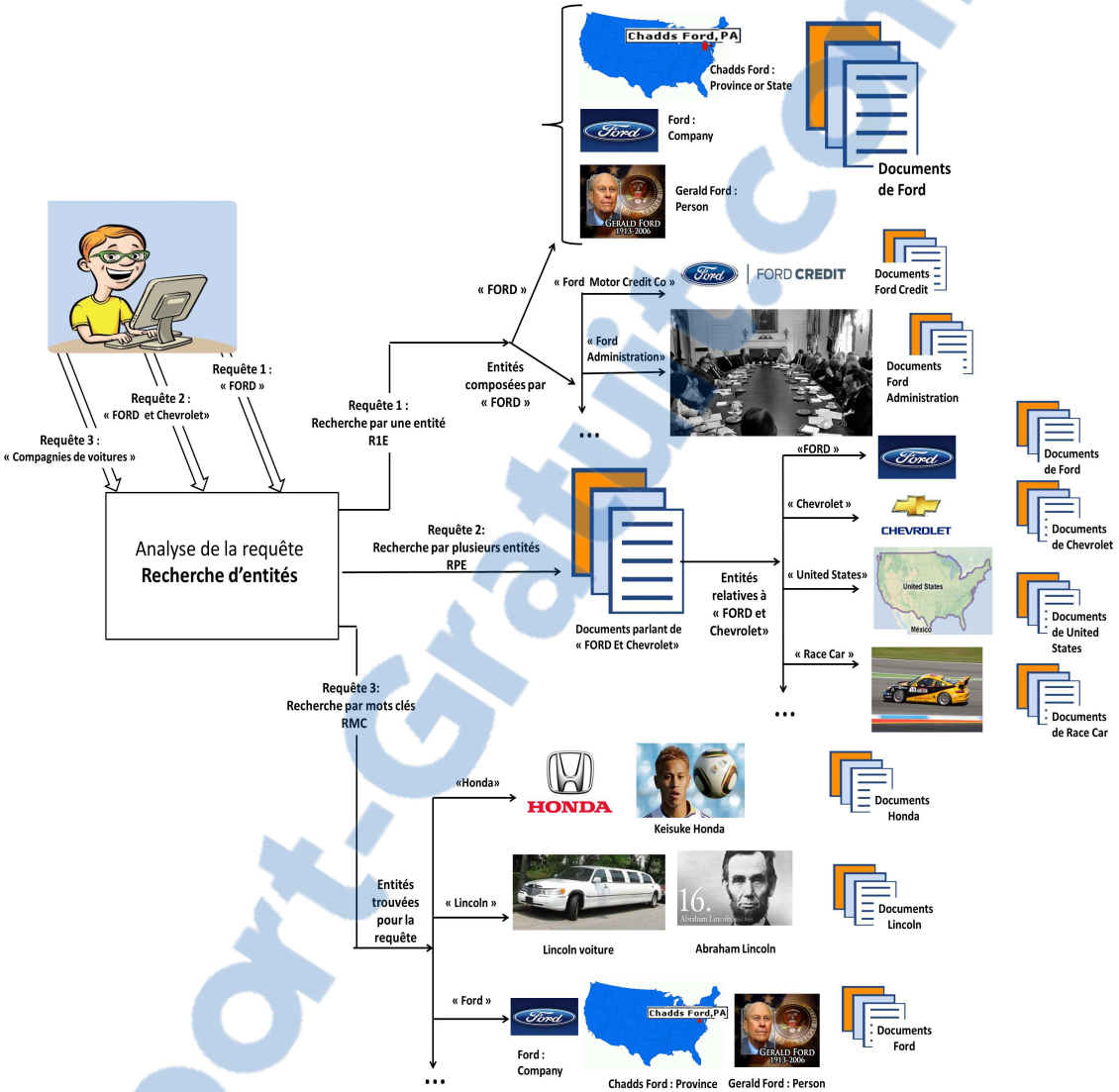

Figure 1.1: Exemple de motivation

`Requête 1`. Dans le premier scénario, *Sami* veut avoir des informations sur une marque de voiture particulière. Ce cas représente la recherche d'une entité : R1E. *Sami* saisit l'entité (connue) qu'il veut trouver : "Ford". Il aura comme résultat l'entité "Ford" en tête de la liste des résultats et des entités composées, par exemple : l'entité "Ford Motor Credit Co", le service financier de la compagnie de voitures Ford et "Ford Administration", l'administration du président Gérlad Ford ainsi que d'autres entités composées. Nous supposons que c'est intéressant de retourner à l'utilisateur les entités composées en plus de l'entité Ford avec ses résultats en tête de la liste des résultats.

Nous avons nommé cette approche : diversification d'interprétations.

Sami pourra explorer les documents relatifs aux entités trouvées.

L'entité "Ford" possède plusieurs types. Il peut s'agir de Gerald Ford l'ancien président des Etats Unis, de la marque de voiture Ford ou encore d'une région en Amérique. Nous supposons d'une part que c'est plus intéressant de prendre en compte les différents types d'une même entité et de retourner à l'utilisateur les documents les plus pertinents de chaque type et d'autre part les différentes catégories des documents pour une entité lorsqu'elle n'a qu'un seul type. Nous avons nommé cette approche : diversification par types dans le premier cas et diversification par catégories dans le second cas.

`Requête 2.` Dans le deuxième scénario, *Sami* veut avoir des informations sur deux marques de voitures (il cherche un lien ou une comparaison entre les marques). Ce cas représente la recherche par plusieurs entités : RPE. *Sami* exprimera sa requête en utilisant naturellement un "et" entre les entités (connues). Sa requête sera : "Ford et Chevrolet". Il est intéressant de retourner à *Sami* les entités relatives à la requête que nous avons nommé contextuelles. Ces dernières apparaissent dans le même contexte que la requête. Nous proposons de retourner aussi les documents relatifs à chaque entité. L'utilisateur pourra explorer ces documents.

`Requête 3.` Dans le troisième scénario, *Sami* veut avoir des informations sur les différentes companies de voitures. Ce cas représente la recherche d'entités par mots clés : RMC. *Sami* saisit sa requête : "compagnie de voiture". Nous proposons aussi de retourner les entités relatives (contextuelles) à cette requête et les documents répondant à chaque entité.

Les résultats de chaque entité contextuelle trouvée (pour les cas RPE et RMC) seront traités comme le cas de la recherche par une entité R1E (diversification par types si l'entité a plusieurs types sinon diversification par catégories). Nous avons nommé la recherche des entités composées et contextuelles : diversification d'interprétations.

1.4 Plan de la thèse

Nous présentons notre travail en structurant les chapitres en deux parties, s'organisant comme suit :

- La première partie constitue une introduction aux notions de base et un récapitulatif des travaux de la littérature qui sont relatifs à notre thème. Dans la sous partie notions de base, nous définissons les axes relatifs à notre thèse, tels que : la recherche d'information et l'extraction d'information, nous parlerons dans ce point sur l'annotation sémantique, sur les systèmes d'annotation automatiques et sur l'indexation des documents. Nous présentons par la suite, la recherche des entités ainsi qu'un tableau comparatif entre la recherche d'entités et la recherche d'information classique. Nous finirons cette sous partie par la présentation de deux manières de traiter les résultats pour les retourner à l'utilisateur, à savoir, la sélection selon la pertinence ou selon la diversité.
  Dans le deuxième sous partie nous présenterons un état de l'art que nous avons divisé en deux sous parties, la première portera sur la recherche d'entités et la seconde sur la diversité.

- La deuxième partie a été divisée en trois sous parties, la première pour le contexte de notre travail, la deuxième est consacrée à la présentation des étapes de notre approche ainsi que les différents algorithmes. Dans la dernière sous partie, nous présentons les résultats des expériences réalisées. Cette deuxième partie sera consacrée dans sa globalité, à la description détaillée de notre approche.

- Nous finirons, comme tout travail de recherche, par une conclusion qui récapitule la problématique que nous avons traitée et les résultats obtenus, ainsi que quelques améliorations prévues pour une continuation de ce travail.

Première partie

Notions de base et état de l'art

2

Notions de base et prérequis

2.1 Introduction

Notre travail s'inscrit dans un contexte de traitements spécifiques sur des documents traités au préalable dans le but d'en tirer des résultats pertinents et diversifiés. Nous présentons dans ce chapitre le processus général de notre approche en définissant les problématiques relatives. Nous nous focalisons sur certains aspects, à savoir : la recherche d'information d'une manière générale et l'annotation et l'indexation comme étapes de prétraitement. Nous considérons ensuite le traitement des résultats d'une manière générale suivi de la recherche d'entités et de la diversité des résultats étant les principaux traitements nécessaires à notre approche.

Dans ce qui suit, nous présentons une introduction sur la recherche et l'extraction d'information. Nous définissons après l'annotation sémantique et les systèmes d'annotation automatique ainsi que l'indexation des documents textuels. Nous présentons également la recherche des entités avec une comparaison entre la recherche d'information classique et la recherche d'entités ainsi que le traitement des résultats et le traitement des requêtes Top-k. Nous finissons ce chapitre par la définition de la diversité des résultats de la recherche.

2.2 La recherche d'information

La recherche d'information (RI) suscite depuis fort longtemps l'attention de la communauté scientifique, elle se définie généralement par l'identification de documents qui satisfont le mieux le besoin en informations d'un utilisateur. Ces documents doivent être trouvés parmi une large collection de documents non structurés [13].

Avec l'expansion d'internet, la mise en œuvre de solutions capables d'exploiter le contenu du web et d'améliorer la performance de la recherche est devenue primordiale. Des techniques ont été proposées et des applications ont été réalisées, leur objectif est de fournir aux utilisateurs des réponses pertinentes par rapport aux besoins qu'ils expriment.

2.2.1 Recherche dans le web

L'objectif de la recherche d'information dans le web est de satisfaire les besoins des utilisateurs en information.

Selon la taxonomie présentée dans [14], les utilisateurs effectuent leurs recherches de plusieurs manières : de manière navigationnelle (me donner l'url du site que je veux atteindre), transactionnelle (me montrer des sites où je peux effectuer une transaction, par exemple, télécharger un fichier ou trouver une carte) ou informationnelle (chercher des informations dans plusieurs pages web).

a - Les requêtes navigationnelles

Le but de ces requêtes est d'atteindre un site particulier par saisie directe de l'url ou par parcours manuel ou automatique des liens hypertextes entre sites. Ce mode de recherche de documents du web nécessite la connaissance d'un minimum d'urls pertinentes et intéressantes pour la recherche à effectuer. Or, la taille actuelle du web ne permet pas de constituer cette connaissance. L'utilisation des moteurs de recherche représente une solution intéressante au problème soulevé par le mode de recherche précédent. Le principe consiste à décrire les documents cibles par une requête de mots clés. Après évaluation sur les documents de son index, le moteur de recherche renvoie une liste d'urls (des réponses) jugées pertinentes par rapport à la requête soumise [15].

b - Les requêtes transactionnelles

Le but de ces requêtes est d'atteindre un site où des intéractions vont se passer. Ces intéractions constituent des transactions définies par ces requêtes. Les principales catégories de ces requêtes sont le shopping, le téléchargement de fichiers (images, chansons, etc.), l'accès à certaines bases de données (par exemple, les pages jaunes), la recherche des serveurs (par exemple, les jeux), etc.

Le résultat de ces requêtes est difficile à évaluer, seul le jugement binaire est possible pour savoir si les résultats sont appropriés ou non appropriés. Cependant, la plupart des informations obtenues (par exemple, prix des marchandises, etc.) ne sont pas fournies

en utilisant les moteurs de recherche [14].

c - Les requêtes informationnelles

Les requêtes informationnelles sont les plus proches aux requêtes classiques de la RI. Leur but est d'acquérir des informations supposées être présentes sur une ou plusieurs pages web dans une forme statique. Aucune intéraction n'est prévue sauf la lecture et aucun document n'est créé en réponse à la requête de l'utilisateur [14]. Néanmoins, les moteurs de recherche pourraient conduire à des pages dynamiques. Selon l'auteur de [14], pour les requêtes informationnelles du web, près de 15 % de toutes les recherches effectuées ont comme résultat une bonne collection de liens portant sur le sujet, plutôt qu'un bon document.

L'exploitation de la richesse du web

Les systèmes de Question/Réponse (*Web-based Question Answering*, WQA) [16–18] proposent une manière d'exploiter la richesse du web. Leur but est de trouver des réponses (les pages web) qui sont en rapport avec les mots clés de la question (la requête). Un exemple est présenté dans la Figure 2.1.

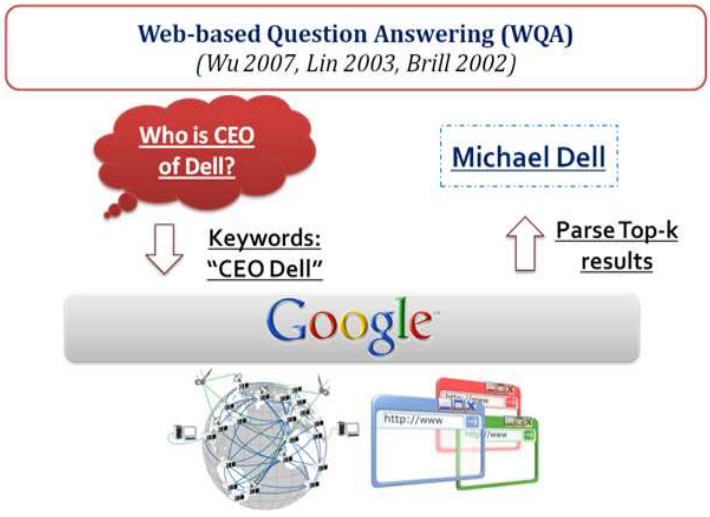

Figure 2.1: Web-based Question Answering [19]

D'après la Figure 2.1, l'utilisateur saisit sa requête "Qui est le PDG de Dell ?". Le système considère les mots clés de cette requête "PDG" et "Dell" et traite les K meilleures pages pour retourner les réponses à l'utilisateur.

Un travail présenté dans [19] a cité certaines limites de cette approche, parmi ces limites :

– L'utilisateur ne trouve pas ce qu'il cherche exactement, il a comme réponse les pages web et doit chercher sa réponse dans les pages.

– Les moteurs cherchent indirectement, les utilisateurs ne peuvent pas décrire et formuler ce qu'ils veulent, il doivent utiliser des mots clés.

– La technique de recherche repose sur la recherche Top-K des pages pour trouver la réponse, certaines pages sont ignorées.

2.3 L'extraction d'informations

Des milliers de sites web apparaissent chaque jour, il est donc devenu crucial pour les utilisateurs de bénéficier d'un accès rapide à l'information demandée. Ceci a motivé, en particulier, la création de systèmes d'extraction d'information qui sont devenus de plus en plus nombreux depuis quelques années. Toutefois, plusieurs défis se dressent devant ces systèmes avant qu'ils atteignent des performances optimales [20]. En effet, il faut tenir compte du fait que les données textuelles contiennent souvent de l'information non structurée, ce qui rend l'extraction d'information plus complexe.

En plus du développement des systèmes d'extraction d'information, les travaux de recherche essayent de répondre à cette problématique de surcharge d'informations en développant d'autres outils spécifiques tels que : les moteurs de recherche, les analyseurs morphologiques [1] et syntaxiques, etc.

Notons que ces outils peuvent présenter une certaine forme de dépendance : par exemple, un système d'extraction d'information peut faire appel à un analyseur morphologique [21].

1. L'analyseur morphologique détermine la formation des mots, c'est-à-dire, le nombre, le genre, la conjugaison, etc. et aussi la dérivation et la composition.

2.3.1 Principe de l'extraction d'information

L'extraction d'information consiste à analyser (souvent superficiellement) des textes pour en obtenir des informations en vue d'une application précise. L'extraction d'information ne cherche pas à comprendre les textes dans leur ensemble, elle fait la recherche d'une information spécifique et extrait d'un texte donné des éléments pertinents. Généralement le type d'une information pertinente pour une application donnée est défini à l'avance [22].

L'extraction d'information dans le web (*Web Information Extraction*, WIE) [23–25] sert à identifier et extraire l'information du web systématiquement (voir Figure 2.2).

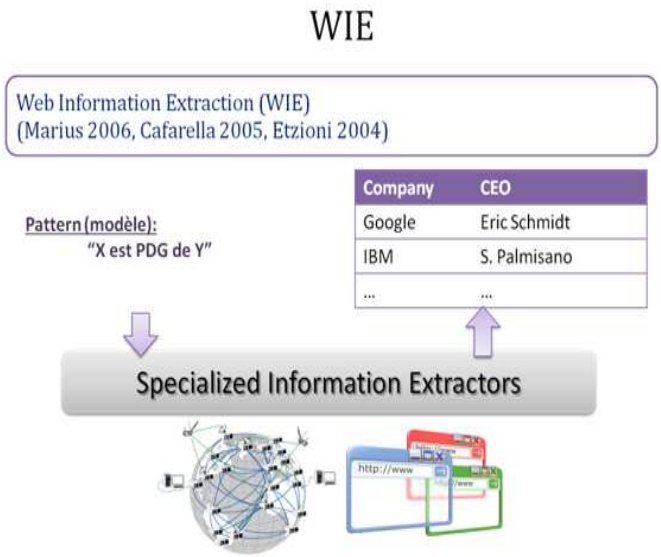

FIGURE 2.2: Web Information Extraction [19]

D'après la Figure 2.2, les systèmes d'extraction d'informations se basent sur les modèles de phrase (*phrase patterns*), par exemple, "X est PDG de Y" pour extraire les résultats.

Le auteurs de [19] présentent certaines limites de cette application :

– Les modèles de requête utilisés sont simples, exemple de la (Figure 2.2), : "X est PDG de Y".

– Le manque d'interactivité.

2.3.2 Reconnaissance des entités nommées

Un autre axe de l'extraction d'informations est la reconnaissance des entités nommées. D'une manière générale, la reconnaissance des entités nommées "REN" (en anglais *Named Entity Recognition*, NER) vise à identifier les séquences textuelles référant à une entité et à lui donner un type sémantique. Les entités nommées sont des unités lexicales particulières, c'est-à-dire, des objets textuels (un mot ou un groupe de mots) "catégorisables" dans des classes telles que noms de personnes, noms d'organisations ou d'entreprises, noms de lieux, quantités, distances, dates, etc.

La reconnaissance des entités nommées est une sous-tâche de l'extraction d'informations qui prend en entrée un bloc de texte non annoté et produit un bloc de texte annoté contenant les entités nommées trouvées. Chaque entité reçoit une étiquette en fonction de son type sémantique.

La reconnaissance des entités nommées consiste à :

- Identifier des unités lexicales dans un texte ;
- Les catégoriser ;
- Eventuellement, les normaliser.

Depuis quelques années, la recherche dans le domaine de la reconnaissance des entités nommées (REN) n'arrêtait pas d'évoluer et différentes méthodes sont apparues :

– Méthodes symboliques [26, 27] : approches linguistiques qui se basent sur des règles génériques (règles contextuelles) écrites à la main (patrons d'extraction).

Exemples :

- Prénom + Mot avec une majuscule = Personne.
- Lieu + verbe d'action = Organisation.
- Mot inconnu + "Inc." = Organisation.

– Méthodes à base d'apprentissage [28] : approches statistiques ou numériques qui se basent sur un mécanisme d'apprentissage à partir d'un grand corpus de textes pré-étiquetés d'où la nécessité de larges corpus annotés.

– Approches mixtes [29] : approches hybrides qui présentent une combinaison entre les deux méthodes précédentes en faisant par exemple :

- L'apprentissage de règles puis révision par un expert.

• L'élaboration de règles par un expert puis extension automatique de la couverture.

2.4 L'annotation Sémantique

Inscrite dans le paradigme du web Sémantique et ayant comme objectif l'enrichissement du contenu textuel, l'annotation sémantique (AS) constitue un moyen de mise en correspondance entre texte et modèle sémantique [30].

Dans le cadre du web sémantique, "une annotation est un commentaire, une note, une explication ou toute autre remarque externe qui peut être rattachée à un document web ou à une partie de celui-ci" [2] [31].

Les auteurs de [30], présentent plusieurs définitions de l'annotation selon le domaine de recherche où elle est utilisée. Dans ce qui suit, nous présentons quelques définitions.

– Dans le contexte du web [32], une annotation est une information graphique ou textuelle attachée à un document et placée souvent dans ce document. Cette place est donnée par une ancre.

– Dans le contexte des interfaces homme-machine, les auteurs de [33] définissent l'annotation comme un commentaire sur un objet tel qu'il existe :

• Un auteur qui crée l'objet à annoter.

• Un annotateur qui commente cet objet.

• Un lecteur qui donne du sens à ce commentaire. Une même personne remplie souvent plusieurs rôles.

Une autre définition est présentée dans [34], annoter un document, c'est : attacher à l'une de ses parties une description qui correspond à l'usage que l'on souhaitera en faire plus tard.

2.4.1 Recherche par types annotés

Une manière particulière d'effectuer la recherche en exploitant l'annotation est la recherche par types annotés (*Typed Annotated Search*, TAS) présentée dans différents travaux [2, 35, 36]. Ces approches de recherche proposent de guider la recherche d'une

2. Définition du W3C.

information en utilisant son type, par exemple, le nom d'une personne. Les techniques utilisées reposent sur les outils d'extraction d'information déjà disponible pour extraire les types des données. Un exemple est présenté dans la Figure 2.3 : "Trouver l'inventeur de la télévision", la recherche se fera en cherchant les noms de personnes en se basant sur le mot clé 'invente' et 'télévision'.

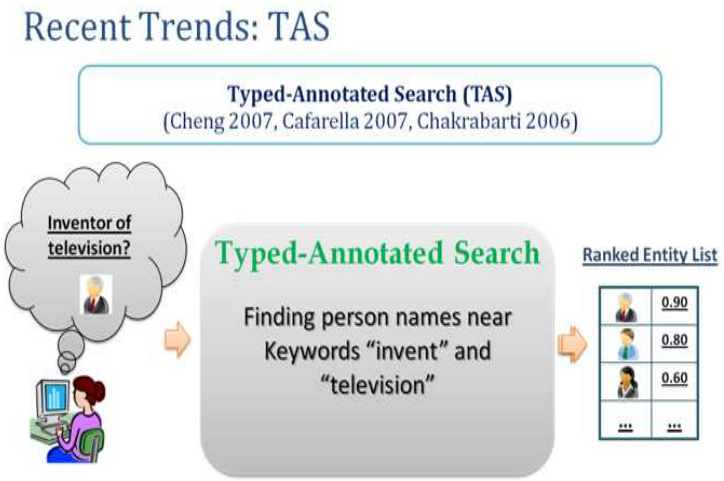

Figure 2.3: Typed Annotated Search [19]

Selon le travail présenté dans [19], les limites sont :

– Le nombre limité des types extraits par les outils disponibles de l'extraction d'informations.

– Le manque de flexibilité par rapport aux modèles existants (*proximity patterns*).

2.4.2 Systèmes d'annotation automatique

A partir de 2007, plusieurs technologies sémantiques intéressantes ont vu le jour. Le problème qu'ils avaient à résoudre est d'expliquer le sens des choses aux ordinateurs. Pour pouvoir résoudre ce problème, l'information doit être annotée par des descripteurs et des relations. L'idée est donc de créer des métadonnées (*metadata*) qui décrivent les données [37]. Pour ce faire, plusieurs systèmes automatiques sont apparus. Parmi ces systèmes nous pouvons citer :

– Open Calais [38] : une initiative de Thomson Reuters [3] lancée en 2007, elle a pour but d'enrichir sémantiquement les textes. Cette initiative permet d'identifier et méta-taguer les personnes, les companies, les faits et les événements et retourne des résultats sous format RDF (Resource Description Framework). Elle permet également de se connecter aux sources de données du web de données "*Linked Data Cloud*" (voir la Figure 2.4 suivante [39]), qui incluent Wikipedia, DBPedia [4], Shopping.com, the Internet Movie Database (*IMDB*), etc. [38]. La notion du web de données est présentée dans la Section 2.4.3.

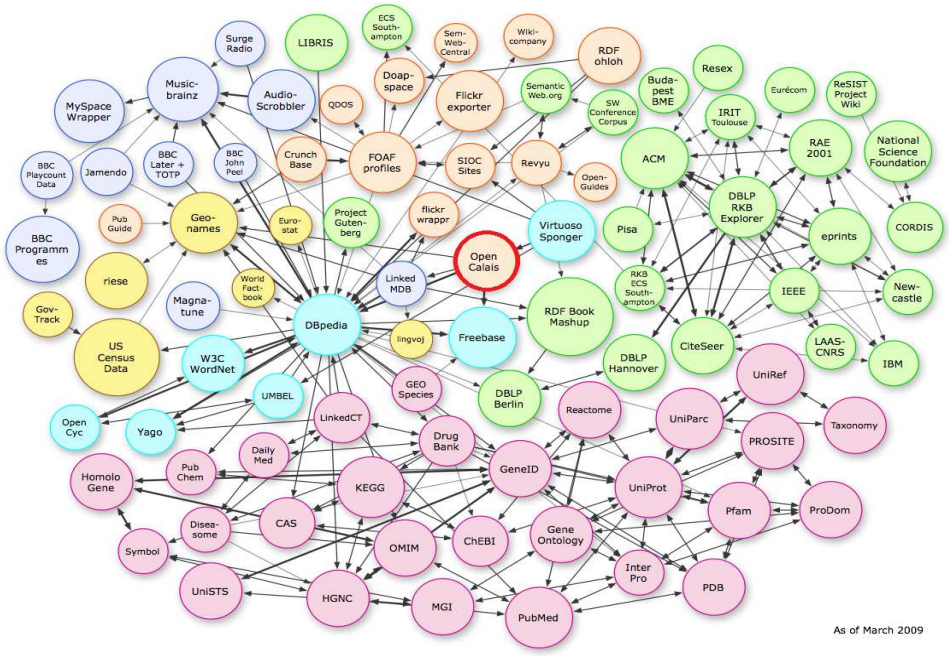

FIGURE 2.4: Linked Data Cloud avec Open Calais [39]

– Evri API [40] : lancée en 2008, cette API permet d'identifier les entités dans les documents texte ou HTML, d'extraire des liens entre les personnes, les lieux et les objets et trouver des faits à partir d'une grande base de données [40].

– Alchemy API [41] : est également une API libre lancée en 2009, elle utilise le traitement du langage naturel et les algorithmes d'apprentissage pour extraire des

3. http ://thomsonreuters.com/

4. DBpedia est un projet universitaire et communautaire d'exploration et extraction automatique de données dérivées de Wikipédia. http ://fr.wikipedia.org/wiki/DBpedia

méta-données sémantiques d'un contenu, telles que des informations sur des personnes, des lieux, des companies, des faits, des relations, etc. [41].

– Zemanta [42] : Contrairement aux API précédentes et dont le but est d'analyser un contenu pour extraire des informations sémantiques et ouvrir l'horizon à de différentes applications, Zemanta retourne des résultats pertinents pour le contexte d'un contenu, par exemple, articles relatifs, hypertextes, images, etc., dans le but d'apporter et de promouvoir un contenu contextuel pertinent à un contenu donné, comme les blogs par exemple [42]. Cette API a été lancée en 2008.

Avec l'apparition de ces systèmes d'annotations automatiques, il est possible de rattacher automatiquement aux documents, des métadonnées sémantiquement riches, en catégorisant et en liant ces documents à des entités (des personnes, lieux, des organisations, etc.) [43]. L'utilisation d'un système automatique peut être alors utile pour annoter automatiquement un corpus de fichiers semi ou non structurés, ce qui permettra d'extraire les entités nommées existantes, les catégories (thèmes ou topics) des documents ainsi que d'autres informations.

Présentation détaillée d'Open Calais

Open Calais [38] est un service web qui rattache automatiquement au contenu soumis des métadonnées sémantiques.

Open Calais extrait des entités nommées ('X' : une personne, 'Y' : une organisation, etc.), des faits ('X' travaille pour l'organisation 'Y') et des évènements ('X' a été nommé PDG de 'Y' le jour 'J'). Ces métadonnées sont ensuite stockées dans une archive centralisée et sont retournées sous forme de modèles RDF (*Resource Description Framework*) accompagnés d'un identifiant unique (*GUID : Globally Unique Identifier*) [43].

Open Calais extrait des documents non structurés : la ou les catégories, les entités, les faits et les évènements (voir la Figure 2.5).

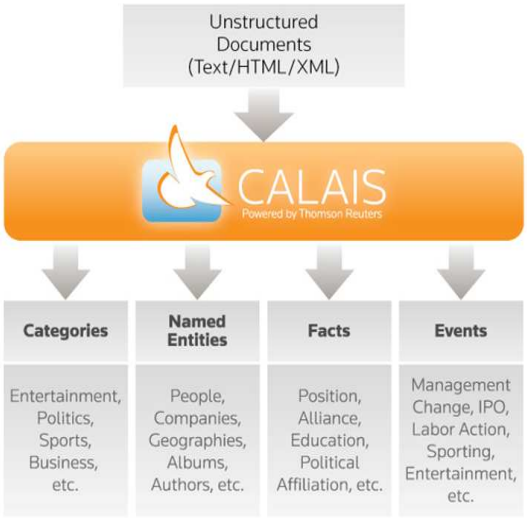

Figure 2.5: Objectif d'Open Calais [38]

La Figure suivante est un exemple d'exécution d'Open Calais Submission Tool [5] (voir Figure 2.6)

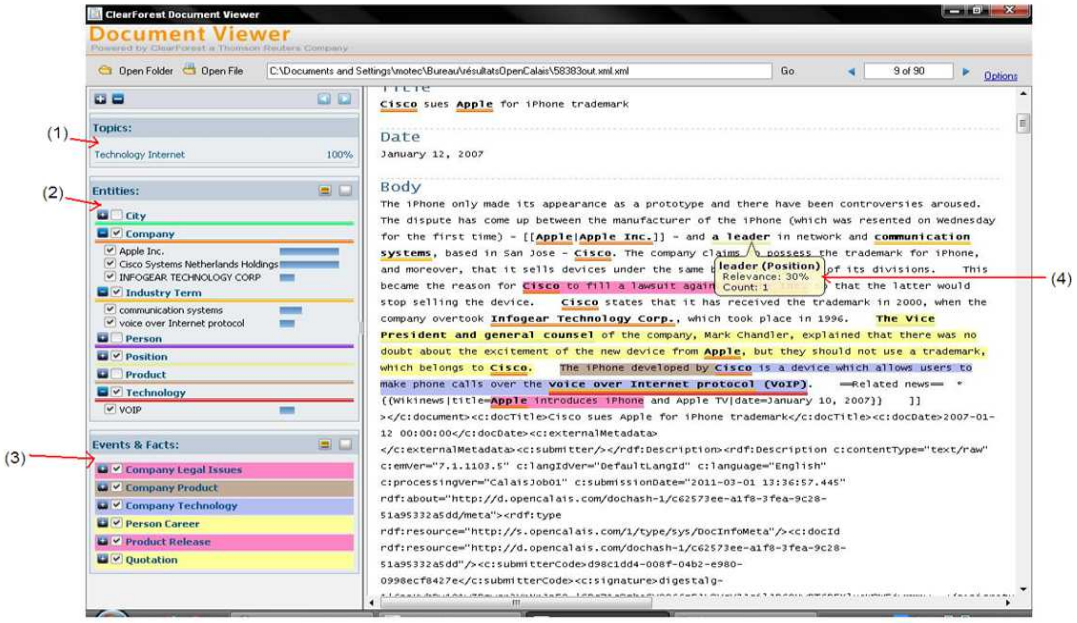

Figure 2.6: Exemple d'exécution d'Open Calais Submission Tool

5. Outil d'annotation d'Open Calais, téléchargeable à : http ://www.opencalais.com/documentation /calais-submission-tool/install-tool

D'après la Figure 2.6, Open calais submission Tool a extrait :

(1) Topic : est le thème ou la catégorie du document ainsi qu'un pourcentage de pertinence (le score), dans cet exemple, la catégorie est : technologies, internet à 100 %.

(2) Les différentes entités extraites (avec types : personnes, compagnies, pays, ...).

(3) Evènements et faits.

(4) Des informations sur chaque entité taguée : son type, sa pertinence, le nombre d'occurrences, le score et parfois d'autres informations.

Le résultat de l'extraction est en RDF et les entités extraites ont des URI.

Remarque importante :

Nous utiliserons le mot "catégorie" tout au long de cette thèse pour désigner le *topic* du document.

2.4.3 Linked data

Le web de données ou "*Linked data*" est une méthode de partage, d'exposition et de connexion des données du web via leurs URIs [6]. L'utilisation des URIs permet l'identification unique des objets. Le principe est de relier les données pour qu'une personne ou une machine puisse explorer le web de donnée et interroger automatiquement les données, quels que soient leurs lieux de stockage, et sans avoir à les dupliquer [39]. L'idée principale est d'augmenter la valeur et l'utilisabilité [7] des données en les connectant aux données relatives. Linked Data est basé sur RDF des documents, qui est utilisé pour faire des déclarations et pointer des objets du monde.

Afin d'accéder au web de données, il existe des navigateurs de données qui permettent de parcourir les différentes sources en utilisant RDF [44]. Par exemple, si un utilisateur est en train de consulter des données sur un produit, il peut être intéressé par des informations sur la société de ce produit. En suivant le lien RDF, il peut accéder aux informations sur cette société même si elle appartient à un autre ensemble de données.

Tim Berners-Lee (le père fondateur du web sémantique) a inventé et défini le terme

6. http ://en.wikipedia.org/wiki/Linked_Data
7. http ://fr.wikipedia.org/wiki/Utilisabilit%C3%A9

Linked Data et son synonyme Web of Data [8] au sein d'un ouvrage portant sur l'avenir du Web sémantique. Il a présenté un ensemble de règles (*Linked Data principales*) pour publier des données sur le web de manière à ce que toutes les données publiées forment un espace unique et global de données [44] :

1. Nommer les objets en utilisant des URI (*Uniform Resource Identifiers*).
2. Utiliser des URI en HTTP afin que les utilisateurs puissent les explorer.
3. Fournir des informations utiles quand quelqu'un consulte une URI.
4. Les données doivent être inter-connectées (contiennent des liens d'autres URI) avec d'autres données.

Ces principes sont une idée de base pour la publication et la connexion des données (réseau global) en utilisant l'infrastructure du web et en respectant son architecture ainsi que ses normes [44].

2.5 L'indexation

Il est impossible d'effectuer pour chaque requête, des recherches directes dans l'ensemble de documents (corpus). Des traitements sont alors nécessaires et doivent être effectués au préalable une seule fois ou à chaque fois que le corpus change. Ces traitements connus sous le nom de l'indexation ont pour but de représenter un document en utilisant les informations qu'il contient.

2.5.1 Indexation vs Annotation

Selon les auteurs de [45], les notions d'annotation et d'indexation semblent équivalentes. On se retrouve dans les deux cas avec une indexation des concepts du document, qui sera utilisée dans le cadre d'une tâche d'exploitation. Néanmoins, il existe les différences suivantes [45] :

– Indexer, c'est décrire un document pour le retrouver ;

8. http ://fr.wikipedia.org/wiki/Web_des_donn%C3%A9es

– Annoter, c'est décrire l'interprétation du document par un lecteur, en vue de n'importe quelle tâche d'exploitation future de ce document.

– On indexe pour une recherche ultérieure, on annote pour donner des traces de son interprétation, pour documenter la tâche que l'on est en train d'accomplir. Ces traces pourront alors être destinées à soi-même, ou partagées [45].

2.5.2 Indexation des documents textuels

L'indexation des documents textuels consiste à identifier des mots clés extraits des textes intégraux pour permettre un ciblage sémantique des documents [46]. L'indexation élargit le champ de la recherche et autorise des requêtes plus souples.

L'indexation de documents textuels est réalisée par un certain nombre de traitements :

– Le filtrage des mots (analyse morphologique).

– La lemmatisation (*Stemming*) (analyse lexicale).

– La modélisation.

– La pondération.

Représentation de documents

Il existe plusieurs modèles de représentation de documents [47–49]. L'un des modèles les plus importants est le modèle vectoriel [47, 50]. La représentation vectorielle des documents consiste à transformer les différents documents d'une collection donnée en vecteurs.

Supposant un document d construit par une ensemble de termes TR tel que : TR=$\{$ tr_1, ..., tr_n $\}$. Le modèle de représentation m du document d doit conserver les termes représentatifs du document d appelés "termes d'indexation". Ces termes d'indexation sont sélectionnés selon leurs fréquences d'apparition dans le document et selon le nombre de documents les contenant. Le modèle de représentation doit également être réduit pour faciliter les traitements de la recherche des informations.

Dans le modèle vectoriel standard, un document d est représenté par un vecteur

V_i. La collection de documents peut donc être représentée par une matrice dont les colonnes représentent les termes d'indexation et les lignes représentent les documents de cette collection [51].

$V_i = (W_1, ..., W_j, ..., W_t)$

W_j est le poids d'un terme tr_j dans le document d_i et $j = 1 ... t$ [51].

Pondération des index

Il existe plusieurs approches pour pondérer les termes d'indexation, c'est-à-dire, les termes significatifs d'un document [50, 52]. Nombre d'entre elles se basent sur les facteurs *tf* : la fréquence d'apparition d'un terme dans un document (*term frequency*) et *idf* : la fréquence d'apparition de ce même terme dans toute la collection considérée (*inverse document frequency*) [50].

Ces facteurs permettent de considérer l'importance (pondération) locales et globales d'un terme et donc d'approximer la représentativité d'un terme dans un document, surtout dans les corpus de documents de tailles homogènes [46].

2.5.3 Techniques d'indexation

L'indexation peut être faite d'une manière manuelle, d'une manière automatique ou bien d'une manière semi automatique, c'est-à-dire assistée.

Dans notre travail, nous nous intéressons principalement à l'indexation automatique, néanmoins, nous présentons brièvement les autres types d'indexation.

L'indexation manuelle

Ce type d'indexation repose habituellement sur un jugement de signification plus ou moins intuitif, lié à l'indexeur. Le travail à réaliser pour la mise au point d'une indexation est assez important : connaissance du contenu de l'information, choix des concepts à représenter et traduction de ces concepts en descripteurs. De plus, les mêmes notions peuvent être exprimées de manières très diverses. En raison de ces divers problèmes, des méthodes d'indexation automatique sont donc apparues [53].

L'indexation automatique

Cette indexation utilise des méthodes logicielles pour extraire et établir une liste ordonnée, c'est-à-dire, un index de tous les mots et leurs occurrences apparaissant dans les documents. Cette liste doit correspondre le mieux au contenu informationnel d'un document [53].

L'indexation semi-automatique

Les systèmes actuels tentent de remplacer l'homme pour une importante part de son expertise et de lui épargner de nombreuses tâches ; mais, ils ne le remplacent pas complètement. L'indexation automatique suppose une intervention totale du système, ce qui est loin d'être le cas, car l'intervention humaine est dans certains cas nécessaire (par exemple, le domaine médicale), d'où l'indexation semi-automatique [45].

2.6 La recherche d'entités

Trouver des entités sur le web est une nouvelle tâche de recherche qui va au-delà de la recherche classique de documents.
Dans certains cas, ce que l'utilisateur cherche réellement dans le web n'est pas des pages web, mais des informations que celles-ci contiennent, c'est-à-dire, des unités ou des entités ; d'où l'idée de rechercher directement les entités voulues.
Ces entités peuvent être [3] :

- Email d'une personnalité ou d'une compagnie ?
- Date d'un événement ?
- Prix d'un matériel donné ?
- Téléphone du service client d'une société ?

Une définition plus détaillée des entités sera présentée dans le chapitre suivant (Chapitre 3, Section 3.2.1).

Il est intéressant de noter qu'un recensement des différents types de requêtes réelles [9] du web a été présenté dans [54]. Les auteurs de ce travail classifient les requêtes en quatre types et présentent le pourcentage de chaque type, comme suit :

– Requête d'entité "*Entity query*" ($\sim$ 40 %), exemple : "1978 cj5 jeep" (marque de voiture).

– Requête de type "*Type query*" ($\sim$ 12 %), exemple : "doctors in barcelona".

– Requête d'attribut "*Attribute query*" ($\sim$ 5 %), exemple : "zip code atlanta".

– Autre "*Other query*" ($\sim$ 36 %), même si $\sim$ 14 % de ces 36 % contiennent un contexte d'entité ou un type.

Des travaux proposent d'effectuer la recherche d'une information précise [2, 3]. De telles techniques utilisent les outils d'extraction d'informations pour extraire les données recherchées, puis relient les unités extraites avec les mots clés de la requête (voir la Figure 2.7 de [55]).

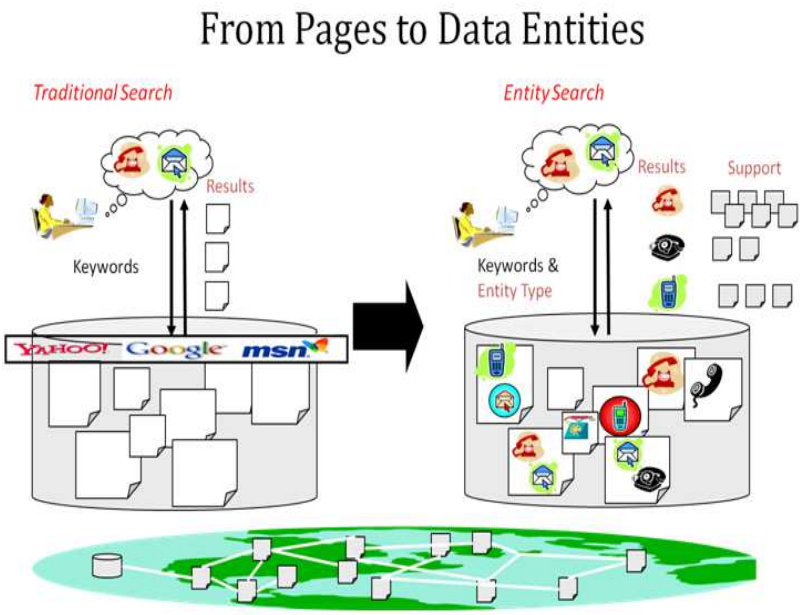

Figure 2.7: Principe de la recherche d'entités [55]

9. A partir des logs de requêtes réelles de Yahoo !

Trouver des entités à la place des documents dans le web est donc un axe de recherche récent dans la RI.

Le web actuel est sous forme de pages. Pour trouver les entités, il est nécessaire d'effectuer un pré-traitement afin d'identifier les entités qui existent dans les documents. Les travaux de l'état de l'art sont détaillés dans le chapitre suivant (Chapitre 3, Section 3.2.2).

Recherche d'information vs Recherche d'entités

Dans cette section, nous allons comparer la recherche d'information classique à la recherche d'entités (RE). Plusieurs critères ont été utilisés pour cette comparaison (voir le Tableau 2.1 suivant).

Critères de comparaison	Recherche d'information dans le web	Recherche d'entités
Apparition	Les premiers moteurs de recherche : Lycos en 1994, Altavista en 1995, etc.	En 2007 le concept a été présenté dans [3]
Objectif	Faire une recherche ciblée de documents pertinents répondant à une requête	Exploiter la richesse du web et extraire les données enfouies dans les pages non structurées
Sources considérées	Différentes sources de données du web	Documents structurés, semi ou non structurées
Type de requête	Mots clés	Entités, mots clés, CQL (Content Query Language) [19]
Type d'informations retournés	pages web	Liste ordonnée d'entités et documents (supports)
Modèles de RI	Vectoriels [47] Booléens [48] Probabilistes [49]	Vectoriels [56] Probabilistes [57]
Ranking	Classement de documents	Classement d'entités

Table 2.1: Recherche d'information vs Recherche d'entités

2.7 La pertinence des résultats de la recherche

Pour être retournés à l'utilisateur, les meilleurs résultats d'une recherche doivent être trouvés et classés en utilisant des méthodes spécifiques, c'est-à-dire, par des algorithmes qui permettent d'attribuer un score à un couple (document, requête). Le

traitement top-k est très étudié dans le domaine de la recherche d'information. Il s'agit de retourner les K meilleures réponses d'une manière rapide et efficace.

Traitement des requêtes Top-k

Comme c'est dit précédemment, le traitement Top-K (*Top-K processing*) consiste à trouver les K meilleurs résultats qui répondent à une requête de l'utilisateur parmi N résultats, où $N >> K$.

La qualité est mesurée par rapport aux scores et les scores sont assignés aux éléments en utilisant une fonction de classement.
Score(r) = $g(f_1(x_1), ..., f_m(x_m))$. g est une fonction d'agrégation qui exprime les préférences de l'utilisateur.

Exemple : Score(r)= (w_1*prix) + (w_2*distance), où w_i est un poids.
Une requête est une combinaison d'attributs gradés (ayant un score) et la fonction d'agrégation donne un score global aux objets en se basant sur les grades d'attributs.

Exemple de requête [58] :
Trouver une maison, requête de préférence :
Select $h.id$ from house h
where new (age), cheap (price, size), large (size)
order by min (new, cheap, large) stop after 5. Le 'min' est une agrégation globale.

Algorithmes fondamentaux

Il existe de nombreux algorithmes pour le calcul du Top-k. Dans ce qui suit nous présentons quelques uns.

– Approche naïve :

Elle consiste à déterminer les objets ayant les meilleurs grades (scores). La base de données est accédée pour trouver les grades des attributs.

Les étapes du calcul naïf des réponses Top-K sont les suivantes :

• Ordonner les scores d'attributs en listes inversées.

• Combiner les attributs par la fonction d'agrégation et calculer le score global de l'objet.

• Retourner les K objets ayant le score global le plus élevé. L'idée est de pré-calculer des listes inversées de scores partiels (des attributs) et agréger (agrégation monotone) les scores partiels des attributs à l'exécution.

– Algorithmes FA et TA :

Le problème de traitement des requêtes Top-k peut être modélisé par des listes d'items ordonnés par leurs scores locaux. Les travaux les plus importants du traitement des requêtes top-k ont été présentés par Ronald Fagin et al. [59, 60], et l'algorithme le plus efficace pour résoudre les requêtes Top-k sur des listes ordonnées est selon les auteurs de [61] l'algorithme TA (*Threshold Algorithm*).

L'algorithme TA [59] est basé sur un autre algorithme FA (Fagin Algorithm) [60], mais la différence majeure entre les deux algorithmes est le mécanisme d'arrêt de chaque algorithme qui permet à TA de s'arrêter plus tôt (grâce à l'utilisation du seuil "*threshold*").

Le Tableau 2.2 suivant récapitule les deux algorithmes : FA (Fagin Algorithm) [60] et TA (Threshold Algorithm) [59].

Caractéristiques	FA (Fagin Algorithm)	TA (Threshold Algorithm)
Accès aux listes Inversées	Accéder à toutes les listes en parallèle et séquentiellement	Accéder à toutes les listes en parallèle et séquentiellement
Arrêt du parcours (condition d'arrêt)	Une fois les K items repérés en commun dans toutes les listes	Score du $K^{\text{ème}}$ item $\geq \theta$ (un seuil)
Calcul du score	1- Accéder aux scores partiels manquants des attributs (accès direct) 2- Calculer les scores	Pour chaque item trouvé dans une liste : 1- Accéder aux autres listes pour récupérer les scores locaux de l'item pour compléter son score global (accès direct). • Calculer le score globale de l'item et stocker dans un buffer de taille k les K meilleurs items. 2- Calculer le seuil ; seuil = $somme$(scores de la ligne en cours) ; • Si score $k^{\text{ème}}$ item $\geq$ seuil, s'arrêter. Sinon refaire (1).
Sortie	Retourner les K meilleurs items	Retourner les K items existant dans le buffer

Table 2.2: Récapitulatif des algorithmes FA et TA

Le Tableau 2.3 suivant présente une comparaison entre les deux algorithmes FA (Fagin Algorithm) et TA (Threshold Algorithm).

Comparaison	FA vs TA
Accès Séquentiel (SA)	Nombre d'accès SA de TA $\leq$ SA de FA.
Accès direct	Effectué dans FA pour les scores manquants uniquement.
Buffer	Buffers limités dans TA à k, buffers non limités dans FA.
Optimalité	TA est plus optimal dans cette classe d'algorithmes.

Table 2.3: Comparaison entre algorithmes FA et TA

Autres algorithmes :

Le Tableau 2.4 suivant présente les caractéristiques de l'algorithme NRA (No Random Access) [59] : cet algorithme traite le cas où les accès directs peuvent être impossibles ou coûteux.

Caractéristiques	Algorithme NRA
Accès aux listes	Accéder à toutes les listes séquentiellement en parallèle.
Calcul du score	Après chaque déplacement de curseur : Calculer le $W(r)$: le plus bas score '*worst*' et $B(r)$: meilleur score de chaque item r rencontré '*Best*'. Ordonner les items par $W(r)$ et terminer par $B(r)$. Le seuil θ = la somme des scores de la liste courante.
Arrêt du parcours	$W(r)$ du $K^{\text{ème}}$ objet $\geq \theta$.
Sortie	Retourner les K meilleurs items.

Table 2.4: Algorithme NRA (No Random Access)

– CA (Combined Algorithm) [59] : cet algorithme traite le cas où les accès directs ne sont pas interdits (sont possibles) mais sont par contre coûteux.
L'optimalité des algorithmes proposés dépend du ratio C_R/C_S. C_R est le coût d'un accès direct et C_S est le coût d'un accès séquentiel.
La motivation de cet algorithme CA qui est une combinaison entre TA (Threshold algorithm) et NRA (No Random Access Algorithm) est de trouver un algorithme optimal où le ratio d'optimalité soit indépendant de C_R/C_S.
Le principe de l'algorithme CA est le suivant :
Posons h= $[C_R/C_S]$, l'idée de CA est d'exécuter l'algorithme NRA mais à chaque 'h étape' exécuter une phase d'accès directs et mettre à jour des informations (les limites B et W).

– Mpro (Minimal Probing) [58] : cet algorithme traite le cas où les accès directs sont coûteux. L'idée est d'exécuter uniquement les calculs nécessaires.
Le Tableau 2.5 suivant présente les caractéristiques de l'algorithme Mpro.

Caractéristiques	Algorithme Mpro (Minimal Probing)
Accès aux listes	Un item de score est accédé séquentiellement et les autres sont déduits. Un ordonnancement de calcul est donné globalement ou par item.
Calcul du score	Calculer le 1er item incomplet et réordonner la liste. Le calcul d'un item r est nécessaire s'il n'existe pas K items r_1, ...,r_k tel que Best(r) < Best(r_i) pour chaque r_i de 1 à k.
Arrêt du parcours	Quand K items avec un score complet sont en tête de liste.
Sortie	Retourner les K meilleurs items.

Table 2.5: Algorithme Mpro (Minimal Probing)

2.8 La diversité des résultats de la recherche

Les approches classiques de classement des réponses renvoient les résultats les plus pertinents aux requêtes des utilisateurs. Ces approches ignorent pourtant certains facteurs importants qui contribuent à la satisfaction des utilisateurs ; par exemple, le résultat peut être redondant.

Le scénario classique de la recherche consiste à identifier un nombre de documents, qui sont susceptibles de satisfaire les besoins d'information d'un utilisateur, en réponse à une requête qu'il a exprimé. En règle générale, les résultats qui sont les plus pertinents à la requête de l'utilisateur sont retournés comme réponse. L'hypothèse sous-jacente est que l'utilité d'un résultat pour l'utilisateur est indépendante de l'utilité des autres résultats retournés. Cela a été reconnu très tôt comme une hypothèse simpliste. En effet, le contenu d'un document retourné peut être redondant, étant donné un autre résultat précédemment examiné. Ceci est vrai si les deux documents sont semblables dans le contenu. Ce problème est appelé la sur-spécialisation "*over-specialization*" [11]. En outre, si la requête de l'utilisateur peut avoir plusieurs sens "*semantic heterogeneity*", la présentation des résultats pertinents à un seul sens uniquement peut laisser l'utilisateur insatisfait [10]. La diversification est généralement utilisée pour résoudre ces problèmes [62]. Les travaux de l'état de l'art sur la diversité seront donnés dans le chapitre suivant (Chapitre 3, Section 3.2).

2.9 Conclusion

Le contexte de cette thèse est relatif à certains domaines de recherche, nous les avons résumé dans ce chapitre. Dans le chapitre suivant, nous détaillerons encore plus les problématiques relatives à notre thèse, à savoir, la recherche d'entités et la diversité des résultats. Nous présenterons alors différents travaux existants autour de ces problématiques.

3

Etat de l'art

3.1 Introduction

Dans ce chapitre, nous présentons les travaux les plus importants sur la recherche d'entités et sur la diversité. La section suivante portera sur la recherche d'entités, nous avons jugé qu'il était nécessaire de détailler mieux ce que c'est qu'une entité avant de présenter les travaux relatifs à la recherche d'entités. La seconde partie de ce chapitre (Section 3.3) est consacrée aux travaux de la diversité.

3.2 Etat de l'art sur la recherche d'entités

Le concept de la recherche d'entités a été présenté en [1], il a pour but d'exploiter la richesse du web afin d'en tirer les données enfouies dans les pages non structurées. La recherche d'entités deviendra une des meilleures techniques d'exploitation du contenu du web. Dans ce qui suit, nous présentons un exemple représentatif de ce qui est considéré comme entité.

3.2.1 Une entité, c'est quoi au juste ?

Les travaux en recherche d'information ont porté une attention particulière aux noms propres de personnes, de lieux et d'organisations, appelés entités nommées. Celles-ci ont été étendues aux dates, aux expressions numériques, aux marques ou aux fonctions, avant de recouvrir un large spectre d'*expressions linguistiques* [63]. L'ensemble des expressions concernées paraissent selon l'auteur de [64] désigner des entités du monde réel. Les entités nommées sont des éléments plus particulièrement sollicités au sein des documents dans un processus de recherche d'information [64].

Un échantillon "représentatif" de ce qui est considéré comme entité nommée est présenté dans [63], comme montré dans la Figure 3.1.

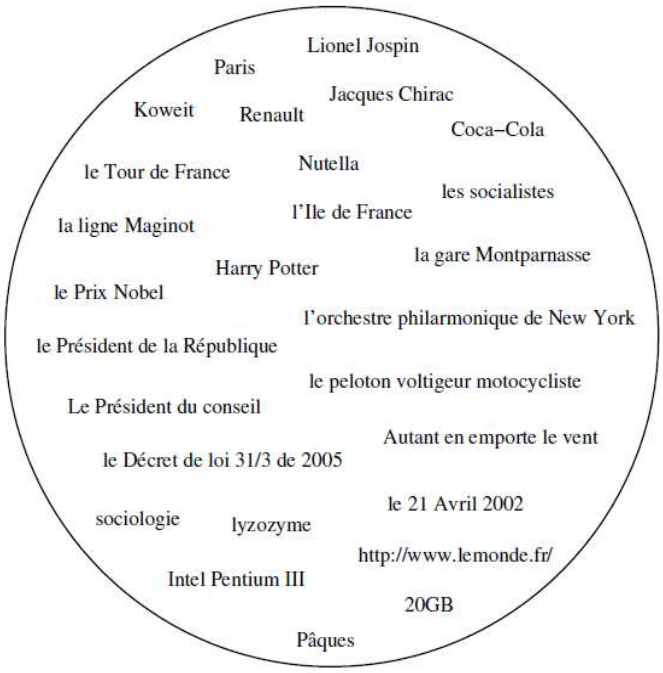

Figure 3.1: Echantillon d'unités lexicales considérées comme des entités nommées de [63]

D'après [63] (voir la Figure 3.1), il y a les grands classiques, reconnus par presque tous les systèmes de reconnaissance des entités nommées, exemples : Lionel Jospin, Jacques Chirac, Koweit, Paris. Sont reconnues également des unités telles que : le Festival du film de Berlin, la gare Montparnasse, le Président de la République, la ligne Maginot, Tour de France, etc. Certains systèmes s'intéressent aussi à des entités telles que : Intel Pentium III Processor ou 20GB ou encore le Prix Nobel.

3.2.2 Travaux relatifs à la recherche d'entités

Dans les travaux de l'état de l'art, les entités sont recherchées pour réaliser différentes tâches pour de différents buts (objectifs). Nous commençons par donner une taxonomie des différentes tâches présentée dans un travail de recherche [56] et nous présentons par la suite notre catégorisation des domaines d'utilisation de la recherche d'entités, c'est-à-dire, les objectifs réalisés par la recherche d'entités, ainsi que les travaux relatifs.

Taxonomie des différentes tâches de recherche d'entités

Dans le travail de recherche présenté dans [56], l'auteur présente une taxonomie des tâches de recherche relatives aux entités (*Entity-Related Search Tasks*), par la Figure suivante 3.2.

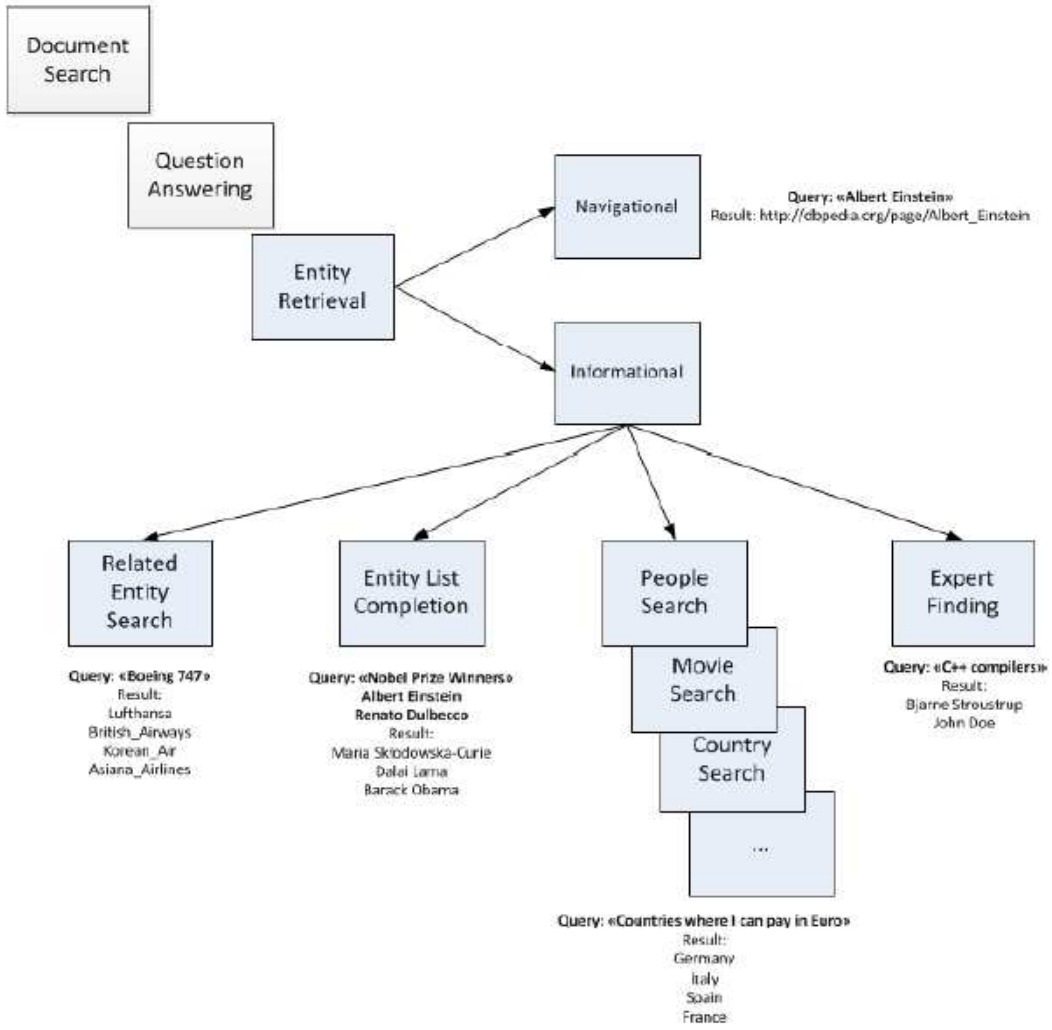

Figure 3.2: Taxonomie des tâches de recherche des entités [56]

La recherche traditionnelle peut nécessiter un effort de la part de l'utilisateur pour identifier l'information voulue parmi les résultats retournés. L'auteur de [56] a classifié certaines applications qui visent à satisfaire l'utilisateur en lui facilitant la recherche. Ces applications sont les suivantes :

– **Recherche d'expert "*Expert Finding*" [65]**

Cette tâche est exécutée dans un contexte d'entreprise. Elle consiste à trouver des personnes [66], par exemple, un expert (employé, associé...) qui a un savoir particulier sur un sujet donné.

– **Recherche d'entités "*Entity Retrieval*"**

Trouver des entités pour de différents types est un challenge qui va au delà de la recherche d'information classique et au delà de la recherche d'un seul type d'entités (comme la recherche d'expert présentée précédemment). La motivation est que les requêtes ne cherchent pas des documents, mais une liste d'entités spécifiques : pays, acteurs, chansons, etc.

Des exemples de requêtes cherchant ces entités spécifiques sont donnés dans [56] : "Conducteurs du formula1 qui ont gangné le grand prix de Monaco", "Membres du groupes Coldplay", etc. Un outil commercial a été présenté par Google pour effectuer cette tâche, *Google Squared*[1].

Pour des requêtes différentes, les entités peuvent être de différents types : personne, pays, etc., mais pour une même requête, les entités sont d'un seul type.

– **Complétion[2] de la liste d'entité "*Entity List Completion*"**

C'est une tâche similaire à la recherche d'entités. Dans ce cas, l'utilisateur en plus de sa requête, doit fournir au système de recherche des exemples d'entités pertinentes. Par exemple, pour la requête "pays où je peux payer en Euro" l'utilisateur doit sélectionner Allemagne, Italie, Espagne [56]. Le système retourne les entités complémentaires qui ne sont pas fournies par l'utilisateur.

– **Question/réponse "*Question Answering*"**

Cette tâche a été présentée précédemment en chapitre 1 (Section 2.2.1). Il est important de mentionner que la recherche d'entités est un peu similaire aux requêtes de

1. http ://googleblog.blogspot.com/2009/06/square-your-search-results-with-google.html
2. Action de rendre complet. http ://fr.wiktionary.org/wiki/compl%C3%A9tion

Question/Réponse (QA). Les questions dans le contexte de QA, sont caractérisées par le qui, quand, où, pourquoi et combien [67], et comme dans le cas de recherche d'entités, les réponses attendues sont précises et l'utilisateur s'attend à une liste d'items.

– **Entités relatives "*Related Entities*"**

Une autre tâche consiste à trouver des entités similaires ou connexes à d'autres entités. Dans ce cas, l'utilisateur peut soumettre une requête formée d'une entité. Pour une entité donnée, soit "New York" par exemple, on pourrait s'attendre à trouver des entités associées comme les endroits à visiter à New York (par exemple, "Empire State Building", "Statue de la Liberté") ou des gens célèbres (par exemple,"Rudy Giuliani"), etc. [56].

Les entités associées peuvent être présentées à l'utilisateur en tant que listes regroupées par types : endroits, personnes, etc.

Dans ce cas, le système offre à l'utilisateur une possibilité de navigation plutôt qu'une liste des entités (comme dans la recherche d'entités). Un prototype commercial qui a été proposé pour exécuter cette tâche est Yahoo ! Correlator [3] [56].

Des initiatives d'évaluation par des compagnes de TREC [4] et INEX [5] ont portées sur les tâches précédentes, la Figure 3.3 suivante récapitule ces compagnes.

3. http ://sandbox.yahoo.com/what-is-correlator

4. Le Text REtrieval Conference (TREC) : un programme conçu comme une série d'ateliers dans le domaine de la recherche d'information. Son but est d'encourager les travaux dans le domaine de la recherche d'information en fournissant l'infrastructure nécessaire à une évaluation objective à grande échelle. http ://fr.wikipedia.org/wiki/Text_REtrieval_Conference

5. The INitiative for the Evaluation of XML-Retrieval, créée en 2002 dans le but d'évaluer des algorithmes de recherche dans les documents xml. http ://en.wikipedia.org/wiki/XML_retrieval

Campaign	Task
TREC Enterprise (2005-08)	Expert finding
TREC Entity (2009-11)	Rel. entity finding
	List completion
INEX Entity Ranking (2007-09)	Entity search
	List completion
SemSearch Chall. (2010-11)	Entity search
	List search
INEX Linked Data (2012-13)	Ad-hoc search

Figure 3.3: Compagnes d'évaluation de tâches relatives aux entités [68]

Les tâches présentées dans la Figure précédente sont [68] :

– TREC Enterprise (2005-2008) porte sur la recherche d'experts "*Expert finding*" définie précédemment. Cette tâche est relative à l'entreprise (par exemple, l'intranet d'une grande organisation). Son principe est le suivant :
Soit un requête donnée en entrée, retourner les personnes qui sont expertes dans le thème de la requête ("*query topic*"). Le résultat est une liste d'expert potentiels.

– TREC Entity track contient la tâche entités relatives "*Related Entity Finding*" et la tâche "*List completion*".

• Le principe de la recherche d'entités relatives est le suivant : soient une entité en entrée (définie par son nom et sa page "*homepage*"), un type de l'entité cible (personne, organisation, emplacement), et une description de la nature de la relation. Retourner les pages (*homepages*) des entités relatives.

• Le principe de la *list completion* est similaire à la recherche d'entités relatives. Un ensemble d'entités données en exemple et identifiées par leurs URIs est nécessaire aussi.

– INEX Entity Ranking track (2007-2009) porte sur le classement des entités. Elles sont représentées par leurs pages Wikipedia.

– Semantic Search Challenge (2010-2011) porte sur la recherche d'entités et sur *list search* qui regroupe la tâche "*Related Entity Finding*" et la tâche "*List completion*". Les entités sont représentées par des URIs.

– INEX Linked Data track (2012-2013) : les entités sont représentées par des pages wikipedia enrichies par des propriétés RDF de DBpedia et YAGO [6]. Le but est de reconnaitre et faire une désambiguïsation des entités dans le texte, c'est-à-dire, "*entity linking*".

Domaines d'utilisation de la recherche d'entités

La recherche d'entités est exploitée pour de différents buts. Un des buts consiste à faire le requêtage des données pour extraire les entités, un autre but est d'exploiter la recherche d'entités pour concevoir un moteur de recherche dédié aux entités, un troisième but est l'analyse des résultats des moteurs de recherche classique pour faciliter la navigation à l'utilisateur. Nous présentons dans ce qui suit cette catégorisation des domaines d'utilisation de la recherche d'entités avec les travaux les plus importants de chaque catégorie.

a - Requêtage du contenu

Les premiers travaux [1–3] sur la recherche d'entités proposent principalement des approches efficaces pour le passage à l'échelle dans le contexte du web. Leur but est d'exploiter la richesse du web et de faire le requêtage de son contenu afin d'en tirer les données enfouies dans les pages non structurées. L'idée est de transformer le répertoire des pages web en un répertoire d'entités nommées.

Le projet WISDM [7] regroupe ces premiers travaux [1–3, 19, 69].

– Le travail présenté dans [3] est une première implémentation de ce concept. Il a été motivé par la construction d'un système précédent des mêmes auteurs nommé : "*Meta-Querier*" [70]. Ce système a pour but de trouver et interroger les sources de données du web. La question qui s'était posée est la suivante : étant donné que les utilisateurs peuvent interroger de multiples sources et que ces sources sont parcourues

6. Yago est une base de connaissance dévelopée en 2008 par l'institut Allemand Max Planck. en.wikipedia.org/wiki/YAGO_(database)

7. WISDM : Web Indexing and Search for Dynamic Mining, http ://wisdm.cs.uiuc.edu/

et les données sont tirées, comment identifier directement et intégrer des données enfouies dans les pages résultats non structurées. Le nouveau système [3] propose alors de rechercher directement l'information voulue, c'est-à-dire, les entités d'une manière globale en parcourant toutes leurs occurrences dans toutes les pages. Ce système doit permettre une intégration agile sans schémas rigides pour une meilleure scalabilité ainsi qu'une flexibilité.

L'interface du système est la suivante (*Figure* 3.4) :

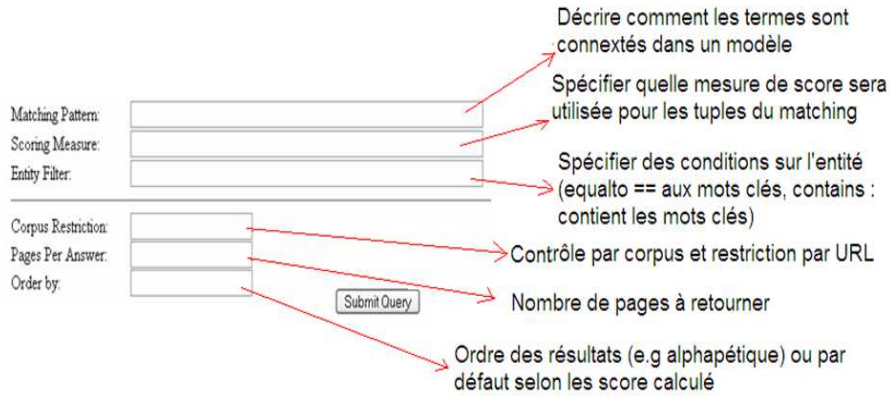

Figure 3.4: Interface du système

Les mesures de score, ("*scoring*" des résultas) représente un problème de recherche à part entière, d'où un travail suivant sur le classement des entités "*Entity Rank*" [2].

– Le travail présenté dans [2] propose une méthode de classement des résultats "*Ranking*". Cinq caractéristiques sont prises en compte :

- Caractéristique 1 : le contexte. Utiliser le contexte qui entoure l'entité.
- Caractéristique 2 : l'incertitude. Les extractions ne sont pas exactes, exemple : un numéro peut être extrait comme étant un numéro de téléphone, alors que c'est une référence d'un matériel.
- Caractéristique 3 : la multitude. Plusieurs sources considérées.
- Caractéristique 4 : la discrimination. Les pages web sont de qualité variée.

• Caractéristique 5 : l'associativité. Des associations sont considérées vraies accidentellement. Exemple de [2] : trouver l'email du prof. Luis Gravano's. L'email info@acm.org apparait fréquemment avec "Luis", "Gravano". Cette association est accidentelle, du fait que info@acm.org apparait dans plusieurs pages ou le nom du prof est écrit.

Les auteurs ont généralisé ensuite leur système pour une recherche efficace d'entités en ligne.

– Dans [1], le web est considéré comme un répertoire d'entités où les entités sont de différents types (téléphone, email...). Chaque entité est un ensemble d'instances qui sont extraites d'un corpus.

Exemple [1] : #phone = { "800 − 201 − 7575", "244 − 2919", "(217)344 − 9788"... }.

Le '#' est utilisé pour distinguer les types d'entités des mots clés.

Le problème de la recherche d'entité défini dans ce travail est le suivant :

Étant donnée une collection d'entités $E = \{ E_1, E_2, ... , E_N \}$ dans une collection de documents $D = \{ d_1, d_2, ... , d_n \}$.

L'entrée est une requête d'entités ("*Entities*") et mots clés ("*Keywords*") : (E_1, E_2, ... , E_m, K_1, K_2, ... , K_l).

La sortie sont des tuples $t = <e_1, ..., e_m, K_1, ..., K_l>$, ordonnés selon le score(t), score du tuple t.

Le traitement se fait en deux étapes, dans deux modules (*Local Query Processing* et *Global Query Processing*) :

• *Local Query Processing* : fait la recherche dans les listes inversées des instances d'entités avec les mots clés. Si le résultat est trouvé, le tuple est instancié. Son score local est calculé selon la fiabilité de l'extraction d'entité et la relation contextuelle entre l'instance d'entité et le mot clé (sa position dans le document). Le tuple sera envoyé à *Global Query Processing* pour l'agrégation.

• *Global Query Processing* : envoi la requête aux nœuds distribués du *local query processing*, les tuples seront associés à leurs scores locaux pour les agréger globalement en considérant des facteurs comme la popularité des pages. Le classement de

sortie (score global) se base sur les scores finaux des tuples.

En d'autres termes, le modèle de classement des entités utilisé dans [1] se base sur deux étapes principales :

- Traitement local : les tuples sont instanciés avec leurs scores locaux, c'est-à-dire, considérer l'incertitude des entités extraites et la relation contextuelle entre les instances d'entités et les mots clés dans le matching.
- Traitement global : les scores locaux doivent être agrégés par rapport à tous les documents. La popularité des documents, la fréquence des mots clés et des entités sont prises en compte. Les tuples seront classés selon le score d'agrégation issu des deux étapes.

b - Conception d'un moteur de recherche d'entités

– Le premier moteur de recherche d'entités a été présenté dans [71]. Le système proposé considère le problème de la recherche des entités nommées pour répondre à une requête de recherche dans un corpus de fichiers textes. Le système crée un document de concordance pour chaque entité. Ce document représente toutes les phrases contenant l'entité dans tout le corpus. Les documents de concordance seront indexés et recherchés en utilisant des logiciels libres de recherche et le résultat est une liste des entités classées.

L'expérimentation de ce travail [71] montre la performance du système, et particulièrement sur quelques classes d'entités comme : le type personne "*people*".

– Les auteurs de [19, 69] proposent le système DoCQS (*Data-oriented Content Query System*), un système général pour le requêtage du web qui se base sur les tables relationnelles. Les auteurs recourent à la recherche d'entités et à l'extraction d'information pour trouver les bonnes entités. Ils proposent ainsi un langage de requête CQL (*Content Query Language*) basé sur SQL qui est utilisé pour interroger des tables structurées préalablement. De nouvelles structures d'index et des algorithmes de traitements de la requête sont proposés.

Pour trouver les entités, le système fait appel à des modèles flexibles de matching

pour permettre de rechercher l'entité avec son type '#type'.

Le système fait également appel aux mesures de scores, à l'agrégation et au classement de résultats (les travaux précédents [1–3]). La recherche des entités se fait en plusieurs étapes : spécifier les tables sources, jointure des tables, filtrer les tuples, agrégation et le classement des résultats.

La méthode proposée a pour but de faciliter les traitements. Une solution est donnée en s'inspirant du succès que l'index inversé a connu dans la RI. La proposition est de concevoir des index en considérant les entités comme mots clés pour l'index. Deux index inversés sont conçus pour les documents et pour les entités.

- Index inversé des documents (*Document-inverted index*) : il est conçu comme suit, pour chaque mot clé on garde les documents contenant ce mot clé avec sa position dans ce document (même principe que l'index inversé classique). La même chose est faite pour l'entité sauf qu'en plus de sa position dans le document, on garde la valeur de l'instance. L'entité est considérée comme mot clé.
- Index inversé de l'entité (*Entity-Inverted index*) : cet Index est créé pour les entités populaires. Il garde pour une entité donnée : les documents les contenant, sa postion (*keyword position*), identificateur de son type (*entity id*) et la position du type de l'entité (*entity position*). Cet index est créé pour les paires <keyword, entity>, quand ils sont en relation.

L'interface du système DoCQS est présentée par les figures 3.5 et 3.6 suivantes. La première Figure 3.5 concerne la saisie des requêtes en CQL et la deuxième Figure 3.6 est celle de la recherche d'entités par type ('#type').

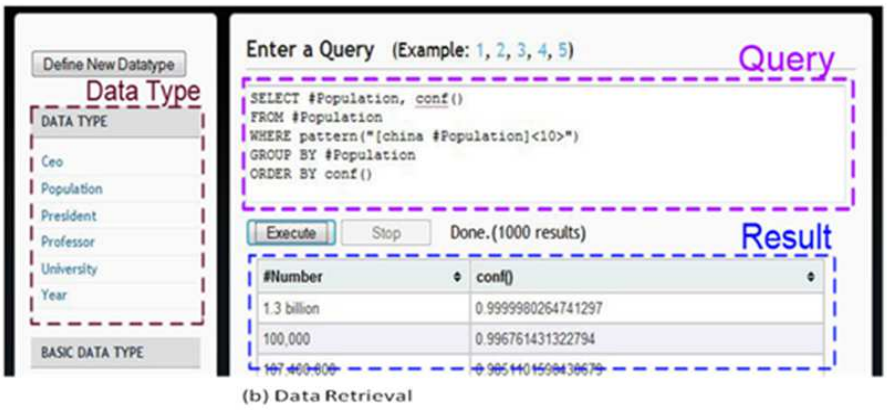

Figure 3.5: Interface du système DoCQS pour les requêtes CQL [19]

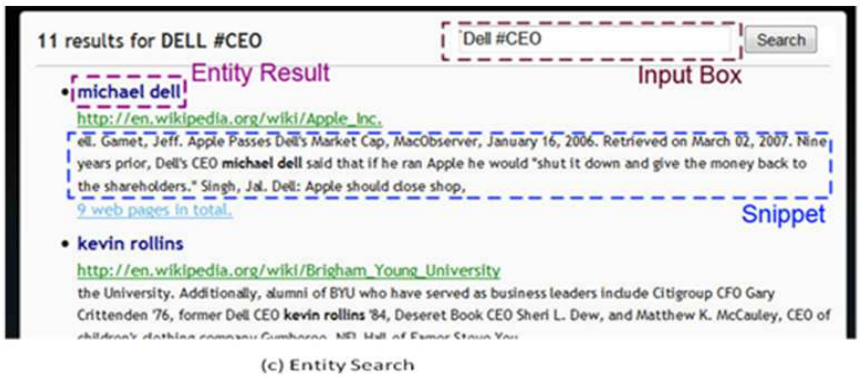

Figure 3.6: Interface du système DoCQS pour la recherche d'entités [19]

– Les auteurs de [34] motivent leur travail dans le contexte des applications dynamiques du web qui nécessitent un accès efficace aux données du web. Ils affirment que ceci est faisable uniquement en utilisant la recherche d'entités (exemple : moteurs de recherches de publications), alors que la plupart des systèmes et applications actuelles se basent sur la recherche par mots clé pour trouver les données. Dans ce travail [34], les auteurs proposent l'utilisation de stratégies plus puissantes comme : les générateurs de requêtes (voir la Figure 3.7).

Pour un ensemble d'entités, les générateurs de requêtes sont capables de déterminer automatiquement un ensemble de requêtes de recherche pour trouver ces entités par un moteur de recherche d'entités. L'apport des générateurs de requêtes a été démontré pour l'intégration des données du web.

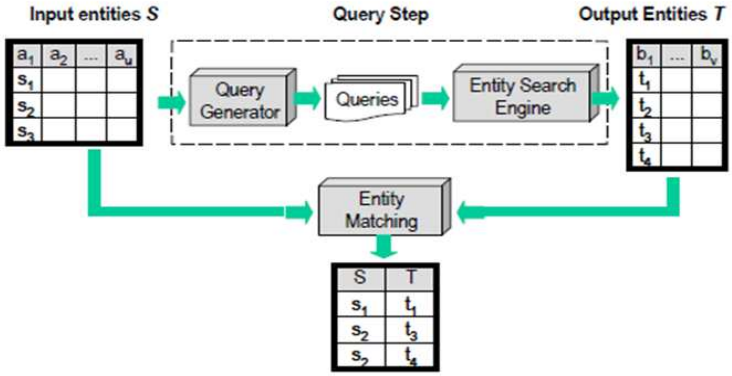

Figure 3.7: Approche de génération de requête [34]

c - Analyse des résultats d'un moteur de recherche

La recherche d'entités est également utilisée pour l'analyse des résultats des moteurs de recherche. L'analyse des résultats s'est avérée utile pour les utilisateurs dans de différents travaux présentant différentes approches.

– Dans [72], l'auteur a démontré que l'identification des entités dans les résultats et leur catégorisation et regroupement en types d'entités (personne, pays, etc.) sont efficacement exploités par les utilisateurs.

– Les auteurs [73] proposent de chercher les entités dans les snippets [8] des résultats, un snippet étant un texte de 10 à 20 mots. Ce travail propose d'enrichir les systèmes de recherche classique par les résultats de la recherche des entités nommées. Cet enrichissement sera effectué au moment de traitement de la requête, aucun prétraitemet n'est fait au préalable. L'inconvénient de cette approche est que les snippets ne contiennent pas toutes les entités nommées.

– Dans un autre travail, les auteurs [74] étendent la recherche des entités du travail précédent en considérant les textes complets des résultats. L'inconvénient de cette

8. Un snippet est la description ou un extrait d'une page web, qui suit le titre et précède l'URL et le lien cache. http ://searchengineland.com/anatomy-of-a-google-snippet-38357.

approche est le traitement du grand volume de données et la consommation considérable des ressources, aucun prétaitement n'est effectué également. Le modèle de calcul distribué du MapReduce [75, 76] a été utilisé comme solution à ce problème. Les auteurs de [73, 74] considèrent la recherche d'entités comme un méta-service fournit avec les systèmes de recherche par mots clés.

– Le graphe de connaissance "Google Knowledge Graph" [77], mis en place récemment par Google comprend les entités du monde réel et leurs relations et améliore la recherche de Google en retournant à l'utilisateur une description de l'entité qu'il cherche si la requête est composée d'une seule entité seulement.

3.3 Etat de l'art sur la diversité des résultats

La diversité a été étudiée dans de nombreux travaux. Son but n'est pas seulement de sélectionner les documents les plus pertinents mais aussi de présenter les documents sous tous les angles. Afin de permettre l'exploration efficace, les nouvelles approches doivent permettre la présentation de tous les aspects d'un sujet donné, en utilisant un nombre de documents représentatifs.

Selon les auteurs de [11], la notion de la diversité a été étudié dans différents contextes. Ils classifient alors les travaux de la diversité selon leur contexte, à savoir, les systèmes de recommandation, les requêtes de bases de données et les moteurs de recherche. Dans ce qui suit, nous définissons les différents domaines où la diversité est utilisée.

3.3.1 Diversité dans les systèmes de recommandation

Avant de définir la diversité dans les systèmes de recommandation, nous commençons par donner un aperçu sur ces derniers.

Les systèmes de recommandation deviennent de plus en plus populaires et importants dans le web d'aujourd'hui. Leur but est d'aider les utilisateurs à trouver des informations pertinentes et intéressantes dans le web.

Approches de recommandation

Deux principales approches sont utilisées pour effectuer la recommandation :

– Recommandation qui se base sur le contenu [78]

Cette approche utilise les attributs des items pour recommander des items similaires. La diversification dans ce cas est facile à appliquer puisqu'il suffira des prendre des items qui ne sont pas similaires.

– Recommandation qui se base sur le filtrage collaboratif [79]

Cette approche se définie par une fonction de distance entre des paires d'éléments. Il s'agit de trouver les utilisateurs similaires ou connectés à un utilisateur donné pour identifier les éléments qui sont bien classés (notés) par ces utilisateurs.

– Il existe également des stratégies de fusion qui combine les deux approches précédentes [80].

Diversité des recommandations

Les chercheurs du domaine des systèmes de recommandation se sont vite rendu compte que si l'ensemble des recommandations proposé à l'utilisateur est homogène, cela pourrait réduire l'intérêt de l'utilisateur [80]. Ce problème peut être résolu par la diversification des recommandations [81]. Le but de la diversification des recommandation est d'identifier une liste d'items qui ne sont pas similaires mais sont pertinents pour les utilisateurs.

– Diversification qui se base sur les attributs (*attribute-based*)

Cette méthode consiste à identifier des items qui diffèrent les uns des autres en se basant sur les valeurs des attributs [78]. Le manque d'attributs (par exemple, le contenu social) et le coût supplémentaire de la recherche des attributs ont donné naissance à une nouvelle approche de diversification des recommandations qui se base sur les explications des recommandations [80, 82].

– Diversification qui se base sur les explications des recommandations (*explanations*) [81]

• Pour les stratégies qui se basent sur le contenu, l'explication d'un item recommandé

est l'ensemble des items similaires que l'utilisateur a aimé dans le passé.

• Pour les stratégies du filtrage collaboratif, l'explication d'un item recommandé est un ensemble d'utilisateurs qui aiment cet item et qui sont relatifs à l'utilisateur (i.e. les amis de l'utilisateur ou ceux qui partagent un comportement commun avec l'utilisateur).

L'idée de la diversification des explications se définit comme suit : étant donné deux différents items recommandés, plus l'ensemble de leurs explications est semblable, plus ils sont semblables l'un à l'autre. La diversification peut être accomplie en identifiant les items les mieux classés par les algorithmes de recommandation, mais qui ne partagent que quelques explications communes. Cette diversité est différente de la diversité qui se base sur les attributs des items [11], car dans le cas du filtrage collaboratif par exemple, aucun attribut n'est nécessaire.

3.3.2 Diversité des résultats structurés

Les auteurs de [11] présentent deux travaux où la diversité des résultats structurés est effectuée.

Dans [83], les auteurs proposent une notion de diversité des résultats structurés qui s'effectue en post-traitement. Le résultat est organisé dans un arbre de décision pour faciliter la navigation des utilisateurs.

Dans [84], les auteurs étudient le problème de calcul efficace des résultats divers dans les applications de vente en ligne. Ils introduisent une notion hiérarchique de diversité dans les bases de données. Par exemple, lors de l'interrogation d'une base de voitures, la diversité peut être appliquée sur une marque d'abord, puis sur un modèle.

3.3.3 Diversité dans les résultats des moteurs de recherche

La plupart des moteurs de recherche effectuent la diversité sur les documents non structurés comme une étape post-traitement [85, 86].

Dans [87], les auteurs ont développé une méthode de reformulation de la requête pour reclasser les Top-N résultats de la recherche, tels que les documents susceptibles

d'être préférés par l'utilisateur sont présentés plus haut. Cette méthode a été proposée du fait qu'il a été constaté qu'un un grand nombre d'utilisateurs modifient leurs requêtes de recherche pour trouver les différents résultats manquants selon eux. Différentes reformulations de la requête sont proposées en fonction des intérêts des utilisateurs. Ceci représente une manière de diversification des résultats des moteurs de recherche.

De ces différents contextes où la diversité est appliquée, nous pouvons remarquer que cette dernière peut se faire sur le sens (par rapport à l'utilisateur), sur le contenu des résultats (par rapport aux attributs des items ou à la redondance des résultats) ou même afin d'apporter de la nouveauté à l'utilisateur, d'où la catégorisation des types de diversité que nous allons présenter dans la section suivante. Nous citerons également quelques travaux importants pour chaque type.

3.3.4 Types de diversité

La diversité peut se faire sur le sens d'une requête (l'intention de l'utilisateur) ou sur le contenu des résultats retournés, elle peut se baser sur les deux aussi.

Diversification basée sur le sens

Ce type de diversité traite les différentes possibilités d'une requête de l'utilisateur, en ayant des probabilités sur toutes les désambiguïsations de la requête. Son but est de retourner les résultats les plus pertinents (proches au sens voulu par l'utilisateur).

Parmi les approches proposées pour ce type, le travail de [62] qui considère le problème des requêtes ambiguës du web. Le but de ce travail est de diversifier les résultats en minimisant le risque d'insatisfaction de l'utilisateur. La pertinence et la diversité ont été prises en compte. Les auteurs de ce travail ont défini une métrique qui prend en compte l'intention des utilisateurs en maximisant la probabilité que l'utilisateur trouvera des informations utiles parmi les résultats de la recherche.

Un autre travail [88] propose de couvrir tous les sens de la requête pour faire de la diversité. Les auteurs de ce travail localisent et mettent en évidence les concepts

des documents pour couvrir les différents aspects. Pour cela, un petit ensemble de phrases est sélectionné pour chaque document. Ceci revient à résoudre un problème de *summarization* de textes. Une nouvelle approche a été proposée pour résoudre ce problème en considérant la couverture et l'orthogonalité.

Diversification basée sur le contenu (similarité)

Le but dans ce type de diversité est de réduire la redondance d'informations dans les résultats. Ceci est réalisé en évitant de retourner les documents qui offrent à l'utilisateur peu d'informations en se basant sur une examination préalable des résultats.

Une approche de diversification du contenu est proposée par [9]. Les auteurs considèrent la similarité des résultats de la recherche Top-k. Ce travail garde les avantages des algorithmes Top-k en limitant la recherche sans explorer tous les résultats et en considèrant la diversité en même temps. Un autre travail de [10] propose un algorithme efficace à la diversité nommé "Divgen". Cet algorithme apporte des performances significatives et des améliorations par le biais de nouvelles primitives d'accès aux données. Dans un autre travail [11], la diversification se définit par une fonction de distance entre des paires d'éléments en se basant sur leurs explications (les éléments qui sont bien classés (notés) par des utilisateurs similaires sont homogènes). Une heuristique présentée dans [12] considère la redondance comme une propriété booléenne, il s'agit de supprimer les résultats redondants à partir d'un seuil.

Une étude bibliographique présentée dans [89] classifie les types de diversification en trois catégories selon la façon dont la diversité entre éléments est définie. Elle ajoute un nouvel aspect qui est la nouveauté (*novelty*) aux deux types précédents (le contenu "*content*" : similarité et le sens "*Coverage*" : couverture des interprétations des besoins de l'utilisateur).

La nouveauté

La nouveauté est très reliée à la diversité. Il s'agit de trouver les éléments qui contiennent de nouvelles informations en les comparant à ceux déjà trouvés. La distinction entre la nouveauté et la diversité est faite dans [90] en considérant que la

nouveauté nécessite d'éviter la redondance alors que la diversité est la nécessité de résoudre l'ambiguïté. Un autre travail [12] basé sur la nouveauté a pour objectif d'ajouter aux systèmes de filtrage adaptatif la capacité de distinguer les éléments redondants. Ces systèmes devraient identifier les documents qui sont similaires à ceux précédemment trouvés, et aussi ceux qui ne sont pas similaires et apportent de nouvelles informations. La redondance d'un document est mesurée en considérant sa similarité par rapport aux autres documents trouvés. Trois manières sont présentées pour mesurer la similarité [12] : la distance géométrique, la distance distributionnelle et la différence d'ensemble qui se base sur le nombre de nouveaux termes trouvés dans un document.

3.4 Discussion

Les approches de l'état de l'art sont motivées dans le contexte du web. Dans notre travail, nous nous intéressons à des domaines d'applications spécifiques où la plupart des documents sont écrits autour d'entités nommées et sont organisés par thèmes. Parmi ces domaines, nous pouvons trouver les forums de discussion, les articles de journaux, les wiki news, etc.

Dans notre travail, nous visons à offrir à l'utilisateur la possibilité de trouver des entités pertinentes aux différentes requêtes tout en augmentant la diversité des résultats.

Notre travail englobe les différents domaines d'utilisation de la recherche d'entités présentées dans les différents travaux de l'état de l'art, à savoir : le requêtage du contenu des documents, la conception d'un moteur de recherche et l'analyse des résultats de la recherche.

Notre définition de la diversité se base sur différentes caractéristiques des entités, à savoir, les types des entités (personne, pays...) et les catégories des documents qui contiennent les entités (politique, sport...) extraites par annotation. A notre connaissance, ce type de diversité n'a pas été proposé dans le contexte de la recherche d'entités. Nous introduisons alors la diversification comme une nouvelle contribution à ce type de recherche.

Dans les chapitres suivants nous présentons notre modèle de données ainsi qu'une définition formelle de notre problème et nous montrons comment ce modèle, couplé à des outils existants de prétraitements et de fouille, peut être utilisé pour répondre à notre problématique.

Deuxième partie

Diversification des résultats de la recherche d'entités

4

Contexte de notre travail

4.1 Introduction

Être capable de trouver des entités précises est une notion intéressante si intégrée dans les moteurs de recherche actuels et dans les systèmes de recherche d'information. Ceci permettra aux utilisateurs de trouver en plus des pages web et des documents, des informations précises, c'est-à-dire, des entités (des personnes, des numéros de téléphone, des livres, des marques de voitures, etc.).

Chercher les entités pertinentes dans une collection de documents n'est pas une tâche facile. Pour cela, un prétraitement est nécessaire pour identifier d'abord les entités dans les documents. Des traitements sont ensuite appliqués selon la tâche qu'on veut réaliser.

Ce chapitre est réservé à la présentation de notre contexte de travail et à la formalisation de notre problématique. Nous commençons par positionner le contexte de notre approche par rapport à la recherche d'entités définie dans l'état de l'art (Chapitre 2, Section 3.2), puis nous présentons un exemple de motivation sur lequel nous nous baserons pour présenter notre modèle de données. Nous détaillerons encore plus les différents types des requêtes considérés dans notre travail à travers cet exemple de motivation. Nous finirons ce chapitre par une définition formelle du problème et par une discussion qui résume l'apport de notre approche.

4.2 Positionnement de notre approche

Dans notre travail [4–6], nous nous intéressons à de nombreuses sources de données telles que les forums de discussion, les articles de journaux et les wikinews, ainsi qu'aux systèmes dédiés à la recherche documentaire. Notre but est de permettre l'exploration des différentes entités trouvées ainsi que leurs documents diversifiés, c'est-à-dire, les documents portant sur chaque entité.

Dans le contexte de la recherche de documents scientifiques par exemple, les utilisateurs pourraient être intéressés par découvrir les documents contenant les entités relatives à une maladie particulière (entités apparaissant dans son contexte, tels que :

les symptômes d'une maladie), ou encore, dans les sources de données historiques, les entités relatives à une guerre entre deux pays (noms de chefs d'états, dates, lieux, etc.). Les entités peuvent avoir plusieurs types (Washington : la ville, ou la personne 'George Washington', ou encore Hollande : pays ou François Hollande). Les entités peuvent également apparaître dans des documents ayant différentes catégories (Politique, Finance, etc.). Cela nous pose deux défis majeurs : associer les entités ou mots clés constituant une requête aux différents types d'entités, et utiliser les types d'entités identifiées ainsi que les catégories des documents les contenant pour présenter les résultats le mieux possible aux utilisateurs. Nous essayerons à travers ce chapitre de formaliser ces problématiques.

La définition de la recherche d'entités "*Entity Retrieval*" présentée dans la taxonomie des tâches de recherche d'entités établie dans un travail de recherche [56] (voir le chapitre précédent : Chapitre 3, Section 3.2.2) consiste à retourner pour une requête donnée un seul type d'entité, c'est-à-dire, la réponse exacte ayant le bon type. Dans notre travail, nous proposons pour une même requête différents types, afin de maximiser la diversité des résultats.

Le problème de la recherche d'entité défini dans notre travail est le suivant :

Étant donné une requête formée d'entités ou de mots clés, le résultat retourné doit être les entités pertinentes de différents types augmentées par le entités relatives "*Related Entities*" (voir le Chapitre 3, Section 3.2.2) que nous nommons entités contextuelles (les détails seront donnés dans le chapitre suivant, Chapitre 5).

Dans notre travail, nous ne nous limiterons pas uniquement à retourner les entités relatives ayant le même type de la requête ; mais aussi les entités ayant d'autres types pour augmenter la diversité. Nous proposons également de faciliter la navigation en regroupant les résultats par entité au lieu de les regrouper par types.

L'utilisateur dans notre approche n'aura pas besoin de saisir d'autres informations sur sa recherche comme dans le cas de la Complétion de la liste d'entité "*Entity List Completion*" (Chapitre 3, Section 3.2.2). Il a le choix de saisir sa requête librement, en cherchant des entités par mots clés ou en cherchant par entités.

Les aspects différents de notre approche par rapport aux travaux de la recherche

d'entités seront présentées dans un tableau récapitulatif à la fin de ce chapitre.

L'apport de notre approche quant à la diversité a été également discuté dans le chapitre précédent (Chapitre 3, Section 3.4). Un tableau récapitulatif sera donné à fin de chapitre pour montrer l'apport de notre diversité par rapport aux types de diversité existants.

4.3 Exemple de motivation

Dans le cadre de ce travail, nous supposons qu'un utilisateur recherche ses informations par deux principales méthodes.

– **Avec connaissance des entités recherchées :** les entités recherchées sont connues. L'utilisateur fait la saisie d'une ou de plusieurs entités et peut s'attendre à des informations pertinentes et complémentaires à sa recherche. Il recherche dans ce cas par entités (recherche par entités).
Ce type de requêtes peut être comparé aux requêtes navigationnelles (voir le Chapitre 2, Section 2.2.1) dans le contexte de la recherche dans le web (recherche exacte).

– **Sans connaissance des entités recherchées :** dans ce cas, les entités sont inconnues pour l'utilisateur, c'est-à-dire, il ignore le nom d'une entité donnée ou bien il effectue sa recherche d'une manière générale dans un but informationnel. L'utilisateur saisit dans ce cas des mots clés pour décrire l'entité qu'il veut trouver mais s'attend également à des informations pertinentes et complémentaires. Il recherche dans ce cas des entités (recherche d'entités).
Ce type de requêtes peut être comparé aux requêtes informationnelles (voir le Chapitre 2, Section 2.2.1) dans le contexte de la recherche dans le web.

Nous présentons dans ce qui suit un exemple de motivation. Cet exemple est tiré d'un contexte d'informations politiques "*news*".

Supposons qu'un utilisateur souhaite avoir des informations sur la politique internationale. L'utilisateur peut poser différentes requêtes (voir la Figure 4.1).

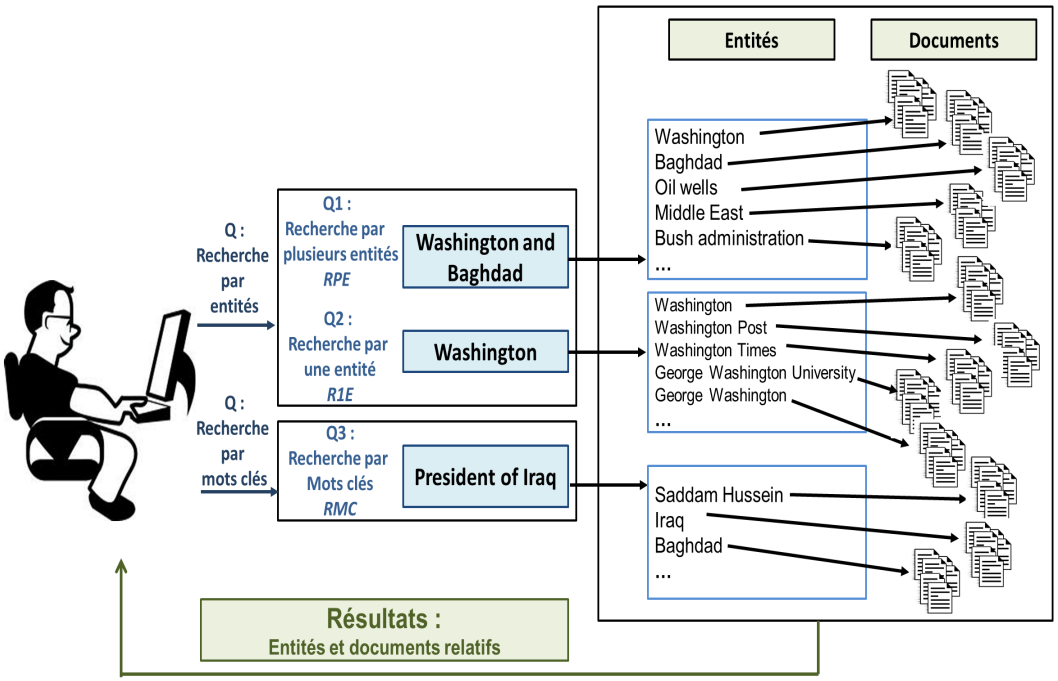

Figure 4.1: Exemple de motivation

`Requête 1.` Dans le premier scénario, l'utilisateur veut avoir des informations sur deux pays, il s'intéresse aux informations communes entre ces deux pays. Ce cas représente la recherche par plusieurs entités que nous avons nommé : RPE.

L'utilisateur saisit les entités (**connues**) : "*Washington and Baghdad*", il aura en résultat les entités relatives à sa recherche ainsi que leurs documents. Nous avons nommé les entités apparaissant dans le même contexte que la requête de l'utilisateur : entités contextuelles. L'utilisateur pourra explorer les documents relatifs aux entités trouvées.

`Requête 2.` Le deuxième scénario est un cas spécial de la recherche précédente RPE. Ce cas consiste en la recherche d'une seule entité, que nous avons nommé recherche d'une entité : R1E. L'utilisateur voudrait des informations sur une entité particulière. Pour cela, il saisit l'entité (**connue**) qu'il veut rechercher, par exemple : "*Washington*".

Il est intéressant de retourner à l'utilisateur, en plus de sa requête : "*Washington*", les entités composées par cette dernière (par exemple, *Washington Post*, *Washington Times*, *George Washington University*, etc.). Ceci augmentera la diversification des interprétations de la requête. L'utilisateur pourra alors explorer leurs documents relatifs.

Pour toutes les entités retournées, nous considérons les différents types d'une même entité. Par exemple, pour l'entité "*Washington*", l'utilisateur peut vouloir chercher des informations sur Washington la ville ("*Washington DC*"), ou bien, il peut vouloir chercher l'état de Washington ("*Washington State*") ou encore "*George Washington*" l'ancien président des Etats-Unis. Nous supposons que c'est plus intéressant de prendre en compte les différents types de l'entité et de retourner à l'utilisateur les documents les plus pertinents de chaque type. Nous avons nommé cette approche : diversification par types.

Nous supposons aussi que c'est intéressant de prendre en compte les différentes catégories des documents les plus pertinents pour une entité lorsqu'elle n'a qu'un seul type (par exemple, *Washington Post*). Nous avons nommé cette approche : diversification par catégories.

`Requête 3.` Dans le troisième scénario, l'utilisateur veut avoir des informations sur une requête formée de mots clés, par exemple : président de l'irak "*President of Iraq*".

Ce cas représente la recherche d'entités (**inconnues**) par mots clés : RMC. Les résultats sont les entités relatives (pertinentes et contextuelles) à cette requête et les documents répondant à chaque entité.

En résumé, nous considérons dans notre travail trois types de requêtes : recherche par une entité (R1E), par plusieurs entités (RPE) et recherche par mots clés (RMC).

Pour les types de requêtes RMC et RPE, les documents des différentes entités trouvées sont traitées comme dans le cas R1E, c'est-à-dire, deux approches sont considérées : diversification par types et diversification par catégories.

4.4 Modèle de données

Nous considèrons un ensemble de documents D, un ensemble d'entités E et un ensemble de mots clés K.

Chaque document contient plusieurs entités (Baghdad, Saddam Hussein, Washington Post, etc.) dans E et des mots clés (capital, president, newspaper, etc.) dans K.

Nous supposons un ensemble de catégories C et un ensemble de types T.

Un document a une ou plusieurs catégories (Politics, Technology_Internet, Sports, etc.) dans C et une entité a un ou plusieurs types (Person, Company, etc.) dans T.

DEFINITION 1. ***Une catégorie***

Une catégorie $c \in C$ a un identifiant *cid*, c'est-à-dire, c : [*cid*] / *cid* : un entier. Par exemple, Politics : [1], Sports : [3].

Comme nous avons dit précédemment, nous utilisons le mot "Catégorie" tout au long de cette thèse pour désigner le *topic* d'un document, c'est-à-dire, le thème.

DEFINITION 2. ***Un mot clé***

Un mot clé $k \in K$ a un identifiant *kid*, c'est-à-dire, k : [*kid*] / *kid* : un entier. Par exemple, journal : [154], président : [343].

DEFINITION 3. ***Une entité***

Une entité $e \in E$ a un identifiant *eid* et un ou plusieurs type(s) *e.types*, c'est-à-dire, e : [*eid*, {*e.types*}] / *eid* : un entier et *e.types* $\in T$.
Par exemple, Baghdad : [10, City], Washington : [25, {Person, City}].

DEFINITION 4. ***Un document***

Un document $d \in D$ a un identifiant *did*, un nom *name*, un ensemble d'entités *d.entities* $\in E$ ayant chacune un score *escore*, un ensemble de catégories *d.categories* $\in C$ ayant un score *cscore* et un ensemble de mots clé *d.keywords* $\in K$ ayant chacun un score *kscore*. Un document est défini comme suit :

d : [*did*, *name*, *d.entities*= $\{(e, count, escore)\}$, *d.categories*=$\{(c, cscore)\}$, *d.keywords*=$\{(k, count, kscore)\}$]. *did* : entier et *name* : String.

• $escore(e, d)$ est le score *escore* d'une entité e dans un document d. Une entité e apparait dans un document selon un *count* qui est le nombre d'apparition de cette

entité e dans le document d.

• $cscore(c, d)$ est le score $cscore$ d'une catégorie c dans un document d.

• $kscore(k, d)$ est le score $kscore$ d'un mot clé k dans un document d. Un mot clé k apparait dans un document selon un *count* qui est le nombre d'apparition de ce mot clé k dans le document d.

Exemple, d : [14, "guerre.xml", $d.entities$= {(Saddam Hussein, 4, 0.843), (Washington, 2, 0.657), (Washington Post, 2, 0.342)...}, $d.categories$ = {(Politics,1)}, $d.keywords$ = {(président, 5, 0.832), (capitale, 2, 0.567), (journal, 2, 0.432)... }].

Nous décrirons les scores plus en détail dans la Section 4.5.

Le diagramme UML de classe représenté par la Figure 4.2 synthétise notre modèle de données.

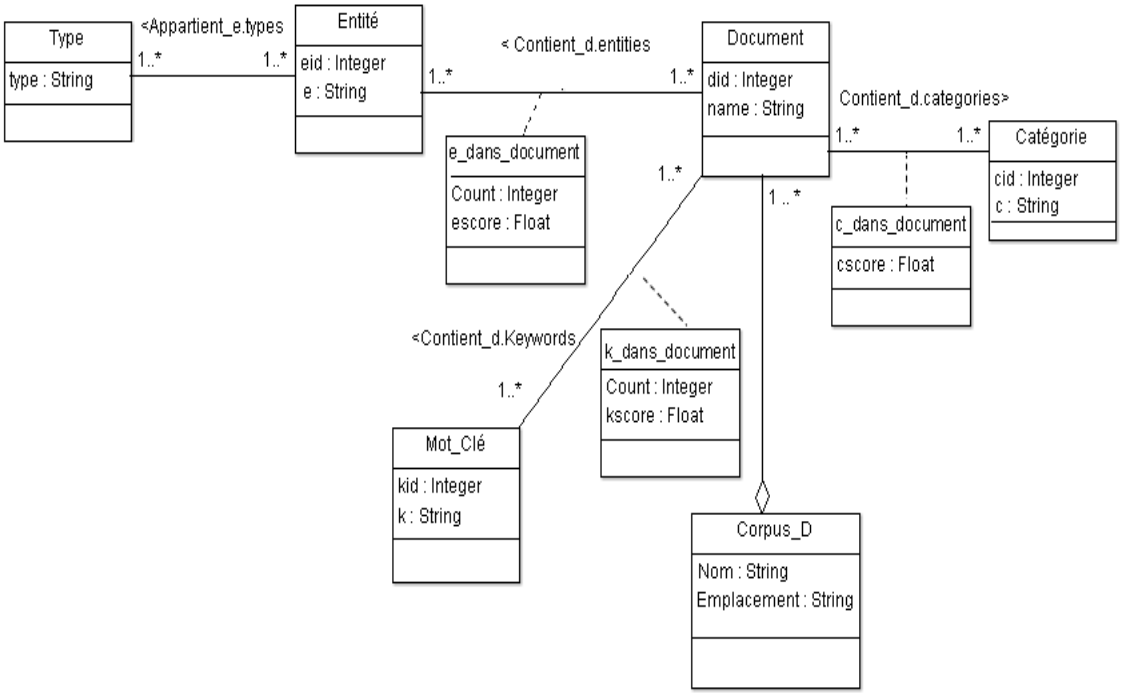

Figure 4.2: Diagramme de classe de notre modèle de données

Dans le diagramme de classe précédent de la Figure 4.2, les différentes composantes de notre modèle de données sont représentées par des classes et ces classes sont liées entre elles selon leurs relations expliquées précédemment.

4.5 Définition du problème

Considèrons une requête $Q = \{ t_1, ..., t_n \} / t_i \in E \cup K$, il s'agit de retourner des documents pertinents et organisés par entité. Nous expliquons d'abord comment les entités sont identifiées puis nous détaillons le calcul de score des documents qui incorpore pertinence et diversité.

4.5.1 Recherche d'entités

Étant donnée Q ci-dessus, il s'agit de trouver un ensemble d'entités $R_{R \subseteq E}$ relatives à la requête et pour chaque entité, de classer les documents qui lui sont associés.

Afin d'exprimer les trois types de requêtes décrits précédemment, nous définissons l'ensemble R comme suit :

$$R = \{CMPentities(Q) \cup Qentities(Q) \}$$

où $CMPentities(Q)$ sont les entités qui composent la requête Q si elle est constituée d'une seule entité e (cas R1E) et $Qentities(Q)$ sont les entités relatives (pertinentes et contextuelles) à la requête Q (cas RMC et RPE).

$$CMPentities(Q) = \begin{cases} \text{e} \cup comp(e) & \text{Si } Q = e \mid e \in E \\ \emptyset & \text{Sinon.} \end{cases}$$

où $comp(e)$=$\{ n / n \in E$ et $e.compose(n) = vrai \}$, l'ensemble des entités composées par e, c'est-à-dire, les entités commençant, finissant ou contenant l'entité de la requête. Donc, $e.compose(n)$ est vraie si une entité n contient e.

$Qentities(Q)$ est l'ensemble d'entités pertinentes (répondant à la requête) et contextuelles (apparaissant dans le contexte).

$$Qentities(Q) = \begin{cases} \text{entities\_of(TopK(Q))} & \text{If } \forall\ t_i \in Q,\ t_i \in K \\ \text{relatedEntities(Q)} & \text{Si } \forall\ t_i \in \text{Q},\ t_i \in E \end{cases}$$

$entities\_of(TopK(Q))$ sont les entités qui apparaissent dans les Top K documents qui répondent à la requête Q, dans le cas RMC.

$relatedEntities(Q)$ sont les entités qui existent dans les meilleurs documents qui répondent à Q lorsqu'elle est constituée de plusieurs entités, c'est-à-dire, dans le cas RPE.

Remarque : si la requête est un mélange d'entités et de mot clés, elle est alors considérée comme une requête de mots clés (RMC), car nous supposons que lorsque l'utilisateur forme une requête d'entités (RPE), c'est qu'il cherche un lien ou veut faire une comparaison (ex., Renault ou peugeot, infection et tumeur, etc.).

Résumé sur la construction de R

En résumé, l'ensemble d'entités R est construit suivant ces conditions :

– Si la requête Q est constituée d'une seule entité e (R1E) : l'ensemble R sera égal à $CMPentities(e)$, c'est-à-dire, les entités composées par e.

– Si la requête Q est constituée par plusieurs entités (RPE) : l'ensemble R sera égal à $RelatedEntities(Q)$, c'est-à-dire, les entités apparaissant dans les meilleurs documents qui répondent à Q.

– Si la requête Q est constituée de mots clés (RMC) : l'ensemble R sera égal à $entities\_of(TopK(Q))$, c'est-à-dire, les entités qui appartiennent aux documents Top K qui répondent à la requête Q.

Cette définition de l'ensemble R des entités permettra d'avoir une diversité des interprétations des différentes requêtes.

Classement des entités (*Ranking*)

Un traitement est appliqué à l'ensemble R pour qu'il soit retourné à l'utilisateur, ce traitement est le classement des entités (*Ranking*). Les entités de l'ensemble R suivent la condition suivante :

$\forall\ e \in R$, $entité\_représentative(e) = vrai$.

$entité\_représentative(e)$ retourne vrai lorsque l'entité représente bien un document d, c'est-à-dire, elle est importante dans le document donc elle apparait fréquemment (nombre d'apparition élevé).

Entité représentative

Contrairement aux mots clés pour lesquels la fréquence d'apparition est rarement utilisée seule et doit être combinée avec d'autre facteurs pour décider de l'importance de ce terme dans un document, la fréquence d'apparition d'une entité est suffisante pour décider de son importance. Plus la fréquence d'apparition est élevée, plus le document parle de cette entité, c'est-à-dire, le document est écrit autour de cette entité (puisque l'entité est une personne ou un pays par exemple). Nous aurons alors :

$entité\_représentative(e)$ est vraie, si $count(e) > threshold$. $threshold$ est un seuil à fixer pour décider de la fréquence d'apparition des entités à considérer. Par exemple, les entités apparaissant plus d'une fois. Nous aurons alors :

$\forall\ e_i \in R$, $count(e_i) > 1$.

Nous aurons aussi :

Pour un document d, $\forall\ e_i \in R$, $\nexists\ e_j \in d.entities /\ count(e_j) > count(e_i)$ et $e_j \notin R$, R contient les entités représentatives de d.

Un cas spécial :

Si toutes les entités d'un document apparaissent une seule fois uniquement, nous les considérons dans le résultat :

Si $\nexists\ e_i \in d.entities\ /\ count(e_i) > 1$, R reçoit $d.entities$, plus précisément :

$\forall\ e_i \in R$, $\exists\ e_k$, $k : 1..m\ /\ count(e_k) \geqslant 1$ et $m = size(d.entities)$.

Pour être retourné à l'utilisateur, l'ensemble R recevra alors les entités représentatives ordonnées selon la pertinence des documents qui répondent à la requête Q et dans lesquels elles apparaissent.

Le traitement est donc le suivant :

$\forall\ e \in R$, $Ordonner(DS,\ entité\_représentative(e))$. Nous nommons DS les documents qui répondent à la requête Q. Ceci consiste à suivre l'ordre d'importance des documents dans lesquels les entités apparaissent.

4.5.2 Recherche de documents

Les résultats sont généralement présentés à l'utilisateur sous forme de listes de documents avec leurs informations, par exemple, le titre, un extrait du document et une adresse (dans le cas des moteurs de recherche). Une autre méthode de présentation des résultats consiste à regrouper les résultats par catégories calculées dynamiquement (*clustering*) ou statiquement (selon des catégories existantes au départ). Dans notre travail nous proposons d'agréger les résultats par entités.

Pour chaque entité $e \in R$, il s'agit d'identifier un ensemble $S = \{d_1, ..., d_m\}$ de documents divers à retourner.

Le problème de sélection de divers items est définit généralement comme suit [11] :

Sélection de divers items

Soit un ensemble X de n items et une restriction k sur le nombre de résultats voulus. Le but est de sélectionner un sous-ensemble S de k items parmi les n items, telle que la diversité entre les items de S soit maximisée.

Dans notre cas, les items sont les documents. Les différents types d'une même entité (ex., ville, personne, organisation, etc.) ainsi que les catégories des documents les contenant (ex., Politique, Médecine, Sports, etc.), peuvent être exploitées pour diversifier les documents à retourner tout en considérant la pertinence de ces documents.

Nous définissons le problème de sélection des documents comme suit :

Sélection de documents diversifiés

Pour sélectionner l'ensemble S de documents, la diversité est maximisée et une restriction est mise sur la pertinence. Nous définissons alors une fonction nommée *div_type_cat* et une autre fonction nommée *pertinence*.

div_type_cat concerne la diversité de l'ensemble des résultats à retourner. Elle vérifie si un type ou une catégorie n'existe pas déjà dans les résultats.

pertinence concerne la pertinence de l'ensemble des résultats à retourner. Elle vérifie l'importance du document, c'est-à-dire, son score pour qu'il soit pris dans le résultat.

Un document $d \in D$ sera pris dans l'ensemble de documents S et retourné avec l'entité e si la fonction *div_type_cat* est vraie ou si la fonction *pertinence* est vraie.

- *div_type_cat* est vraie selon plusieurs cas :

• Si l'entité e a plusieurs types, *div_type_cat* est vraie si un type de l'entité e appartenant au document d n'existe pas dans un ensemble nommé $groupe\_type \subseteq e.types$. *groupe_type* est mis à jour par les types trouvés pour l'entité e dans les documents du résultat (cas de diversification par types).

• Si l'entité n'a qu'un seul type, *div_type_cat* est vraie lorsque la catégorie du document d n'existe pas dans un ensemble nommé $groupe\_cat \subseteq C$ qui est mis à jour par les catégories des documents trouvés pour l'entité e (cas de diversification par catégories).

• Si les ensembles *groupe_type* et *groupe_cat* sont vides (en début de traitement), *div_type_cat* est vraie aussi.

Le document d sera pris alors dans S et soit l'ensemble *groupe_type* est mis à jour par le type de l'entité trouvé dans d, soit c'est l'ensemble *groupe_cat* qui est mis à jour

par la catégorie du document (selon la multitude ou non des types de l'entité e).

Formellement :

div_type_cat est vraie si :

$$\begin{cases} \text{- Pour } e.types > 1 \mid e \in R : \\ \nexists\ type \in groupe\_type \mid type \in e.types \wedge (e, type) \in d.entities. \\ \text{groupe\_type} \subseteq T \\ \text{ou :} \\ -\text{Pour } e.types = 1 \mid e \in R : \\ \nexists\ c \in C \mid c \in groupe\_cat,\ c \in d.categories. \\ \text{groupe\_cat} \subseteq C \end{cases}$$

- *pertinence* est vraie si :

1. Le document d répond à la requête.

2. Le score du document d dépasse le seuil de pertinence.

Formellement :

1. $\exists\ t_i \in Q \mid t_i \in \{d.keywords \cup d.entities\} \wedge \exists\ e \in d.entities \mid e \in \text{R}$

2. $score(d, Q) > \theta$ où θ est un seuil de pertinence minimale.

$$score(d, Q) = \begin{cases} \sum escore(d, e) \in d.entities & \text{Si } e.types > 1 \mid e \in R \\ cscore(d, c) \in d.categories & \text{Si } e.types = 1 \mid e \in R \end{cases}$$

Le score du document d par rapport à la requête Q, $score(d, Q)$ se base sur le score de l'entité $escore(d, e)$ lorsque l'entité a plusieurs types (diversification par type), ou sur le score de la catégorie du document $cscore(d, c)$ lorsque l'entité n'a qu'un seul type (diversification par catégorie).

Le *escore* est score de e par rapport à un type dans le document d. Le *escore* est calculé par le système d'annotation automatique. Le *cscore* est le score d'une catégorie c dans un document d. Il est calculé aussi par le système d'annotation automatique.

Explication du choix d'un document

Le choix d'un document d pour qu'il soit pris dans S est fait selon *div_type_cat*. Ceci permet d'avoir tous les types possibles d'une entité ou toutes les catégories des documents, même si le score du document correspondant à l'entité du type ou à la catégorie ne dépasse pas le seuil de pertinence. En d'autres termes, un document unique par le type ou par la catégorie est toujours pris dans le résultat pour maximiser la diversité.

Le choix est fait aussi selon *pertinence* pour avoir les documents les plus pertinents (dépassant le seuil) portant sur un même type de l'entité e ou sur la même catégorie c.

Cela veut dire que si un document d est choisi en vérifiant *div_type_cat*, c'est que soit son type est nouveau dans la collection à retourner à l'utilisateur, soit sa catégorie est nouvelle.

Si un document d est choisi en vérifiant *pertinence* (pertinence est vraie), c'est qu'il est pertinent par rapport au type ou à la catégorie, c'est-à-dire par rapport aux documents ayant le même type de l'entité ou la même catégorie.

4.6 Discussion

Dans ce chapitre, nous avons présenté une définition formelle de notre problématique qui consiste à retrouver les diverses entités qui répondent à nos trois types de requêtes, de retrouver les résultats de chaque entité et leur appliquer un traitement qui permet de les diversifier.

La définition présentée permet d'assurer un maximum de diversité de types ou catégories avec une pertinence des documents choisis si les documents $d \in D$ sont classés préalablement selon pertinence.

Les caractéristiques de notre approche sont résumées dans le Tableau 4.1 et sont comparées à celles des approches de l'état de l'art sur la recherche d'entités (Chapitre 3). Il est à noter que les travaux de l'état de l'art motivent globalement leurs approches de la recherche d'entités dans le contexte du web.

Approches de l'état de l'art	Caractéristiques des autres approches	Caractéristiques de notre approche
Projet WISDM [1–3, 19, 69]	Les requêtes doivent utiliser des notations spéciales ou le langage CQL.	Les requêtes sont exprimées en langage naturel et trois types sont considérés : R1E, RPE, RMC.
(Bautin, Skiena, 2009) [71]	Les entités retournées sont classées sans diversification d'entités ou de documents.	Les entités et les documents sont diversifiés.
(Kaki, 2005) [72]	Les entités sont identifiées dans les résultats. Aucun lien inverse vers les documents n'est créé.	Chaque entité retournée pointe ses documents relatifs (exploration par entité).
(Fafalios et al., 2012) ; (Kitsos et al., 2013) [73] ; [74]	La recherche d'entités est un méta-service, le but n'est pas d'interpréter les requêtes par des entités.	Notre méthode de recherche sert à interpréter les sources par leurs entités.
Singhal, 2012 [77]	Les résultats sont retournés uniquement lorsque la requête représente une seule entité.	Les résultats sont retournés pour trois types de requêtes.

Table 4.1: Caractéristiques des autres travaux et Apports de notre approche

Dans notre travail, les différents objectifs de la recherche d'entités sont exploités. Nous faisons d'abord le requêtage du contenu lorsque nous appliquons le système d'annotation automatique sur un corpus de fichiers pour extraire les informations enfouies.

Nous proposons ensuite la conception d'un moteur de recherche qui considère différents types de requêtes et retourne des entités et des documents.

Dans notre travail, les résultats retournés sont analysés en utilisant les entités pertinentes et contextuelles et en organisant les documents diversifiés de chaque entité pour faciliter leur exploration.

Le Tableau 4.2 présente l'apport de notre approche de diversité par rapport aux caractéristiques des différents types de diversité.

Type de diversité	Caractéristique du type	Apport de notre approche
Diversité basée sur le sens	Calculer des probabilités sur les différentes interprétations des requêtes.	Les différentes interprétations des requêtes sont couvertes par les entités contextuelles et composées qui sont retournées à l'utilisateur.
Diversité basée sur le contenu	Comparer les similarités entre documents pour éviter la redondance.	Le contenu est diversifié en exploitant les différents types des entités des sources et les différentes catégories des documents.
Nouveauté	Considérer les résultats déjà trouvés en se basant sur de différentes mesures.	La nouveauté est assurée par les différents types et catégories retournés à l'utilisateur ainsi que les différentes entités composées et contextuelles.

Table 4.2: Caractéristiques des types de la diversité

Les approches de l'état de la l'art ont en général proposé des solutions plus ou moins complexes pour effectuer la diversité. Ces solutions retournent une liste diversifiée de documents dans les moteurs de recherche ou d'items dans les systèmes de recommandation. Dans notre travail, la diversité est définie différemment. Nous proposons de faire une double diversification à de différents types de requêtes dans le contexte de la recherche d'entités. Dans le chapitre suivant, nous allons détailler les étapes et les algorithmes de notre approche de diversification des résultats de la recherche d'entités.

5

Approche de diversification des résultats de la recherche d'entités

5.1 Introduction

Dans ce chapitre, nous allons résoudre les problèmes définis auparavant que nous résumons comme suit :

– Identification des entités : trouver les entités afin d'interpréter les sources de données par les entités qu'elles contiennent.

– Choix des entités à retourner : proposer à l'utilisateur en plus de entités pertinentes les entités composées ou contextuelles (relatives).

– Ranking : le nombre d'entités peut être important et toutes les entités ne sont pas toutes pertinentes. Il est alors nécessaire de choisir et classer les entités à retourner.

– Choix des documents à retourner : diversifier les documents les plus pertinents par types d'entités ou par catégories des documents.

Ce chapitre est organisé de la manière suivante. Dans la section suivante, nous proposons l'architecture conceptuelle de notre système et nous expliquons ensuite les étapes de notre approche d'une manière générale. Au cours de la Section 5.3, nous détaillons notre approche en utilisant des schémas et en présentant nos algorithmes. Nous concluons ce chapitre par une discussion sur l'approche proposée.

5.2 Architecture du système

La Figure 5.1 suivante présente l'architecture conceptuelle de notre système et résume notre approche.

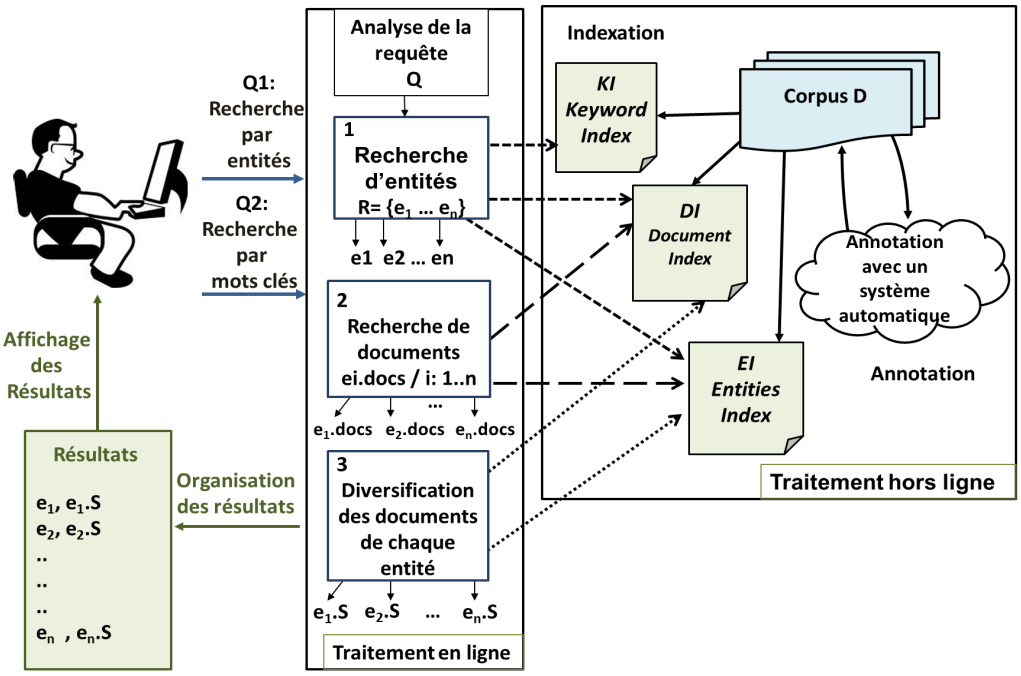

Figure 5.1: Architecture conceptuelle de notre système

D'après la Figure 5.1, nous remarquons les deux principaux modules de notre système : le traitement hors ligne et le traitement en ligne.

Le traitement hors ligne se compose de l'annotation du corpus et de son indexation qui crée trois différents index (KI, DI, EI, voir la Figure 5.1). Ces index seront définis dans la Section 5.4.2.

Le traitement en ligne est le traitement principal de notre approche. Il prend en charge des requêtes de recherche par entité(s) ou par mots clés et consiste à les analyser pour rechercher les entités (Recherche d'entités) en utilisant les trois index, puis à rechercher les documents relatifs à chaque entité (Recherche de documents) en utilisant les deux index EI et DI.

La dernière étape consiste à diversifier les documents par types ou catégories en se basant sur les deux index EI et DI.

Les résultats sont finalement retournés à l'utilisateur sous forme de liste d'entités avec les documents diversifiés correspondants. L'utilisateur pourra alors les explorer (exploration par entité).

5.3 Étapes de notre approche

D'une manière générale, notre approche est la suivante (voir le schéma de la Figure 5.2 suivante) :

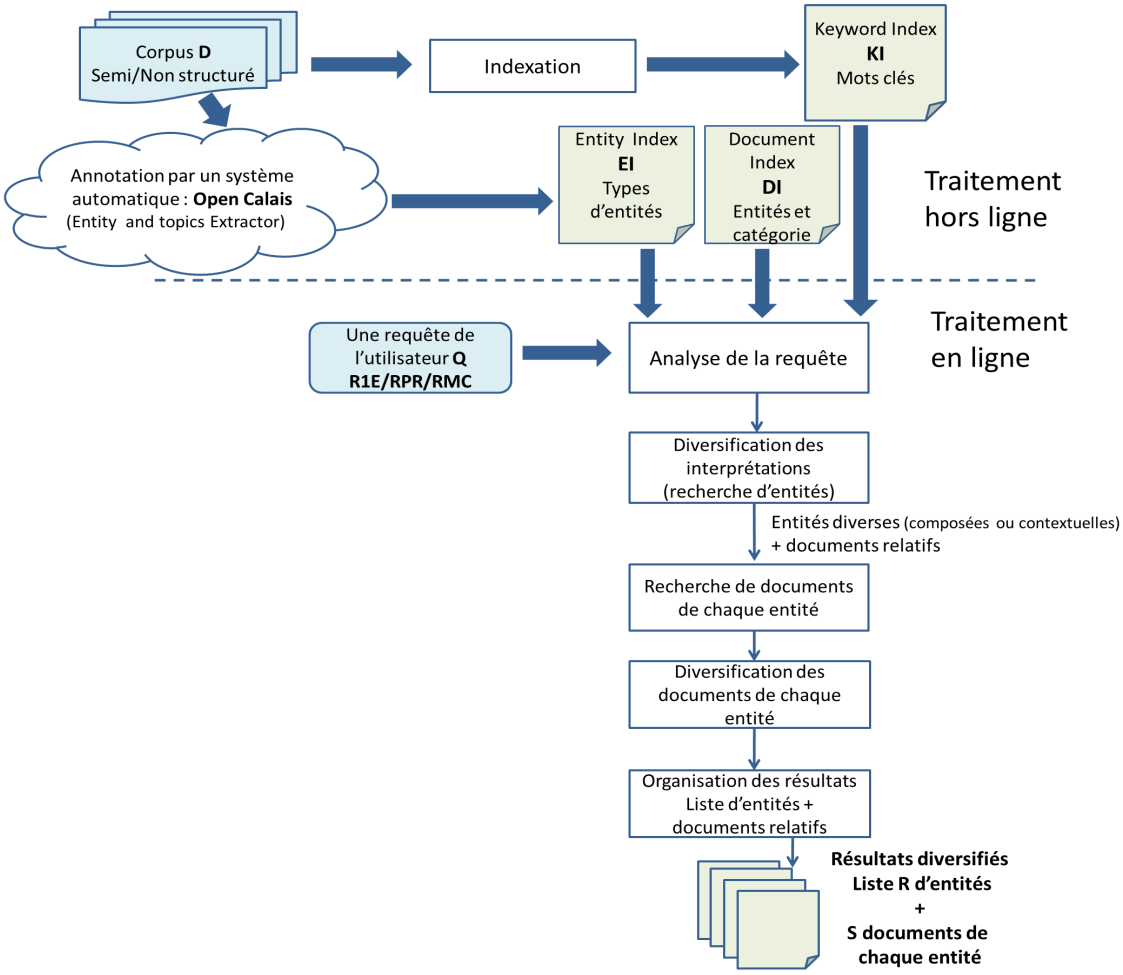

FIGURE 5.2: Étapes de notre approche

Nous considérons D, un corpus de documents semi ou non structurés (par exemple, newsgroups, wikinews, articles de journaux, etc.). Notre approche consiste à faire un prétraitement hors ligne pour préparer les informations au traitement en ligne selon la Figure 5.2 précédente :

– Nous commençons par annoter le corpus en utilisant un système d'annotation automatique tel qu'Open Calais, pour extraire les entités, leurs types et les catégories des documents avec les scores d'extraction (*relevance*).

– Nous créons différents index pour stocker les informations, i.e. les mots clés, les entités, les types, les catégories et les scores.
Les scores (définis précédemment dans la Section 4.5.2) sont : *kscore* du mot clé, le *escore* (extrait par calais) pour le type de l'entité et le *cscore* (extrait par Calais) pour la catégorie du document.
Trois index sont créés, à savoir : un index inversé classique pour les mots clés (KI, Keyword Index), un index inversé pour les types d'entités (EI, Entities Index), i.e. une entité apparait dans un document avec quel type (type de e dans d). Le troisième index est celui des entités et des catégories des documents (DI, Document Index), i.e. quelles sont les entités d'un document d donné et quelle est sa catégorie. Ces index seront détaillés dans la Section suivante 5.4.2.

Dans le traitement en ligne, nous utilisons nos index créés dans la partie hors ligne pour le traitement des requêtes.

Nous commençons par analyser la requête Q pour connaitre son type, selon les trois types définis précédemment : recherche par mots clés ou recherche par entités (RMC, RPE, R1E).

L'idée est de faire une diversification des interprétations de la requête pour retourner un ensemble d'entités pertinentes et diverses R, défini comme suit :

– Si la requête est de type RMC (l'index KI est utilisé pour le traitement des mots clés), R sont les entités pertinentes plus les entités relatives au contexte de la requête que nous nommons : entités contextuelles.

– Si la requête est de type RPE (les index EI et DI sont utilisés), R sont les entités pertinentes et les entités contextuelles, c'est-à-dire, qui apparaissent dans les documents communs entre les entités de la requête.

– Si la requête n'est composée que d'une seule entité (cas R1E), R sera égal à l'entité elle même étendue par les entités composées par cette dernière, c'est-à-dire, les entités qui commencent, finissent ou contiennent l'entité de la requête.

Après avoir trouver l'ensemble R, les documents sont recherchés pour leur appliquer une autre diversification.

Pour chaque entité e ∈ R, il s'agit d'identifier un ensemble S de documents à retourner avec l'entité e à l'utilisateur. Le traitement est le suivant (les index EI et DI sont utilisés) :

– Si l'entité a plusieurs types : les documents de l'entité sont classés dans des groupes selon leurs types. La diversification des documents est faite selon les types de l'entité, "au moins un document" par type (meilleur document, selon le plus grand score) doit être retourné à l'utilisateur pour garantir un maximum de diversité. Les autres documents seront sélectionnés selon pertinence, c'est-à-dire, leurs scores doivent dépasser un seuil.

– Si l'entité a un seul type : les documents de l'entité sont classés dans des groupes selon leurs catégories. La diversification des documents est faite selon les catégories des documents relatifs à l'entité, "au moins un document" par catégorie (meilleur document, selon le plus grand score) doit être retourné à l'utilisateur pour garantir un maximum de diversité. Les autres documents sont sélectionnés selon pertinence, c'est-à-dire, leurs scores doivent dépasser un seuil.

Dans ce qui suit, nous présentons en détail les deux phases de traitements : hors ligne et en ligne.

5.4 Phase de traitements hors ligne

Pour trouver des résultats en réponse à une requête de l'utilisateur, plusieurs traitements sont appliqués au corpus des fichiers afin d'adapter ce dernier au processus de la recherche d'entités et documents. Le traitement et l'annotation du corpus constituera une étape importante de notre approche. Son but est de préparer des index pour faciliter le traitement en ligne des différents types de requêtes.

Dans cette section, nous allons décrire les traitements effectués hors ligne et nous présentons les index cités précédemment.

5.4.1 Annotation

Les systèmes d'annotation automatique de documents apparus récemment (présentés précédemment, voir Section 2.4.2), rattachent automatiquement des méta-données sémantiquement riches à des documents, en catégorisant et en liant ces documents à des entités "*Entity Linking*".

Dans ce travail, nous avons utilisé Open Calais (présenté dans la Section 2.4.2) pour l'annotation des corpus semi ou non structurés. L'utilisation de ce système permet d'annoter automatiquement le corpus et extraire les entités nommées existantes, les catégories des documents ainsi que d'autres informations. Un autre outil qui permet d'extraire les entités nommées avec un type et un score est JRCNames [1]. Cet outil sert aussi à détecter les variantes d'une même entité (Cajkovskij et Tchaikovsky, NATO et OTAN) ou encore à résoudre les ambiguités des acronymes. Cet outil identifie les entités de types : personne, pays, oranisation. Nous l'utilisons pour extraire les entités non reconnues par Open Calais.

Si jamais une entité n'est reconnue ni par Open Calais ni par JRCNames, elle sera après indexée par notre approche comme un mot clé (elle ne sera donc pas ignorée).

Remarque importante :

Il est important de noter que l'annotation dépend de la performance du système automatique mais nous rappelons que le but de notre travail n'est pas d'apporter une contribution à ce niveau mais d'utiliser l'existant afin d'exploiter les différences entre les entités (dans notre cas les types) et entre les documents (dans notre travail les catégories) pour proposer une nouvelle approche de diversification des résultats dans le contexte de la recherche d'entités, et ce quelle que soit la manière d'extraire ces différences.

L'identification des entités se fait avec une certaine incertitude, l'extracteur retourne alors une probabilité de fiabilité de l'extraction (*confidence*). Cette probabilité sera utilisée dans le calcul du score des entités.

1. https ://ec.europa.eu/jrc/en/language-technologies/jrc-names

Pourquoi Open Calais ?

Open Calais peut être choisi pour de différentes raisons [2]. Nous l'avons choisi pour les raisons suivantes :

– D'abord, Open Calais est fourni par Thomson Reuters, l'une des plus grandes organisations de l'information dans le monde. La construction d'une nouvelle application au dessus du service Open Calais est sûre, du fait que ce service sera toujours opérationnel.

– La fiabilité : Open Calais est hébergé dans des *data centers* sûrs. Il est surveillé et disponible.

– La précision : plusieurs outils ont été fournis par Open Calais depuis plusieurs années. Ils ont été utilisés par des centaines d'organisations et par des milliers d'utilisateurs. Bien que les systèmes parfaits sont rares, les développeurs d'Open Calais sont convaincus que leurs outils sont parmi les meilleurs et ils visent à augmenter la précision.

– L'objectif : l'objectif d'Open Calais est de fournir l'infrastructure, l'architecture de la solution est réalisée par l'utilisateur.

– Les langues : la langue anglaise est maitrisée par les services d'Open Calais, la langue française un peu moins mais des améliorations sont prévues pour le Français.

– La performance : Open Calais excelle dans certains domaines comme les médias et le sport, ce qui nous arrange pour le contexte de travail de cette thèse.

2. http ://www.opencalais.com/blogs/tom/why-opencalais

Le diagramme UML de classe représenté par la Figure 5.3 présente les différents types extraits par Open Calais. Pour l'exemple de la figure, Open Calais a été appliqué sur un petit corpus de Wikinews [3].

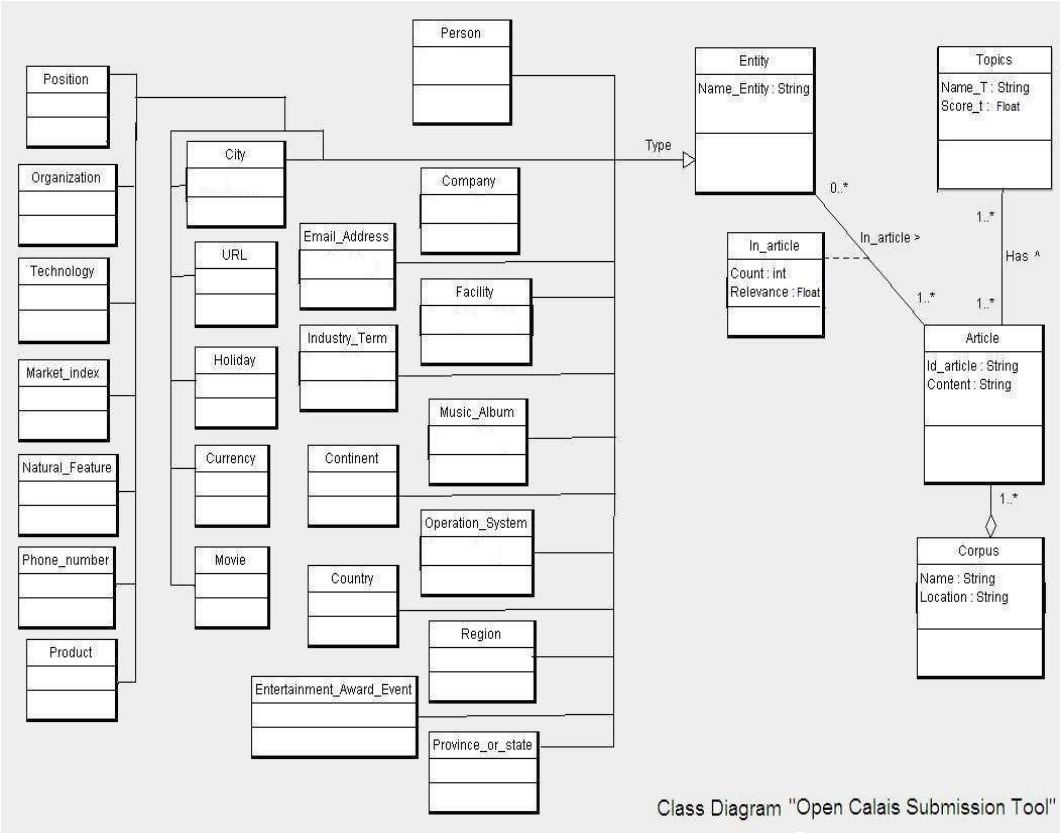

Figure 5.3: Types extraits par Open Calais

Nous remarquons d'après la Figure 5.3, qu'Open Calais a extrait 24 types malgré la taille du corpus (333 fichiers environ), ce qui représente une variété qui va nous aider dans la diversification.

3. http ://www.rapidlibrary.com/files/wikinews-11mar1-zip_ulcqfvecqyi89on.html

5.4.2 Indexation

En plus de l'annotation, un autre traitement hors ligne est nécessaire, ce traitement est l'indexation. L'indexation consiste à traiter le corpus de documents D pour créer un index des termes importants et représentatifs (index classique de mots clés). L'indexation consiste aussi à stocker les informations extraites par l'annotation dans d'autres index. D'autres tâches sont exécutées en hors ligne aussi telles que le calcul des fréquences d'apparition des mots clés localement (dans le document) et globalement (dans tout le corpus) ainsi que la fréquence d'occurrence des entités dans un document (extraites par l'annotateur).

Comme c'est dit dans les étapes de l'approche, nous aurons besoin de trois index (deux index inversé et un index des documents). Ces trois index sont détaillés dans ce qui suit :

– **KI** (Keyword Index) : est un index inversé qui fait correspondre à chaque mot clé k les documents le contenant avec un score $kscore$ (score de k par rapport à un document d).
Nous aurons, k : { (did, $tf.idf(k,d)$) }, $tf.idf(k,d) = kscore(k,did)$.
Par exemple, president : { (59436, 0.18), (59517, 0.09), (75967, 0.03) }.

$tf.idf$ est un poids obtenu en multipliant le tf (*term frequency*) qui est la fréquence d'apparition du terme par le idf (*inverse document frequency*) qui est la fréquence inverse du document qui mesure l'importance du terme dans le corpus. La formule du tf*idf est calculée comme suit [4] :

$tf = n_{ij} \ / \ \sum n_{kj}$, $k : 1 \ldots N$.
i : le terme, j : le document j, n_{ij} : nombre d'apparition du terme i dans le document j, $\sum n_{kj}$: la somme des termes du document j, N : nombre de documents du corpus.

4. http ://fr.wikipedia.org/wiki/TF-IDF

L'idf est le logarithme de l'inverse de la proportion de documents du corpus qui contiennent le terme :
$idf(t, D) = log\ N\ /\ |\ \{\ d \in D : t \in d\ \}\ |$.
N : nombre de documents du corpus, $|\ \{\ d \in D : t \in d\ \}\ |$: nombre de documents ou le terme t apparait. Pour éviter la division par 0 (si le terme n'apparait pas dans le corpus), cette formule est ajustée [5] à $1+|\ \{\ d \in D : t \in d\ \}\ |$.

L'identifiant *did* est utilisé pour stocker un document d car c'est un numéro unique. Cet index est nécessaire pour trouver les documents dans le traitement des requêtes de mots clés (RMC).

La Figure 5.4 suivante présente un exemple de création de l'index KI.

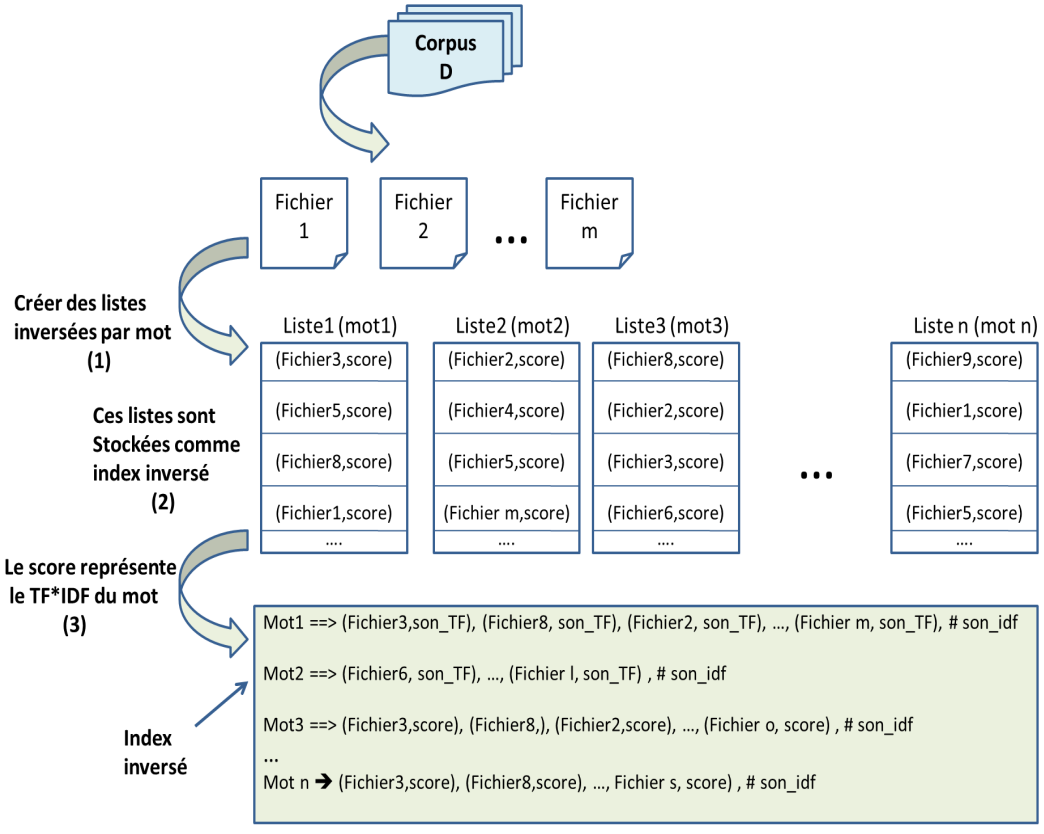

Figure 5.4: Création de l'index de mots clés "KI"

5. http ://en.wikipedia.org/wiki/Tf%E2%80%93idf

L'algorithme (*Algorithm 1*) suivant décrit les étapes de création de l'index de mots clés "KI" selon la Figure 5.4.

Algorithme 1 : Création KI

Input : D ; Corpus de documents
Output: KI /*Keyword Index*/

```
1.1  begin
1.2      créer_fichier(KI) ;
1.3      /*Créer le fichier Keyword_Index*/
1.4      foreach d in D do
1.5          extraire_termes() ;
1.6          /*Extraire les termes k (Keywords) du document*/
1.7          foreach k in d do
1.8              if (k.existe(KI)) then
1.9                  /*Si le terme k a été déjà trouvé dans un autre document (existe dans KI)*/
1.10                 ajouter(did, tf, KI) ;
1.11                 /*Ajouter le document contenant le terme et son score à k*/
1.12                 /* Seul le tf est stocké puisque idf peut être calculé une seule fois car il est
                     inchangeable pour un terme.*/
1.13             else
1.14                 écrire(k, did, tf, KI) ;
1.15                 /*Ecrire le terme k, son document, son tf dans KI*/
1.16             EndIf
1.17         EndFor
1.18     EndFor
1.19 End
```

– **EI** (Entities Index) : les entités sont indexées de la même manière que les mots clés. Un index inversé (EI) est construit pour les entités en plus de l'index inversé traditionnel des mots clés (KI). L'index EI retournera une liste qui contient les informations des entités.
Open Calais est utilisé pour l'extraction des entités avec leurs types et leurs scores, il est renforcé par l'outil JRCNames pour l'extraction des entités non reconnues.

Cet index inversé stocke pour une entité donnée e les types de l'entité avec les documents les contenant selon un score '*esore*' (score du type de l'entité e dans un document d) et un count (nombre d'apparition de l'entité dans le document) calculés en se basant sur l'annotateur.
Nous aurons, e : { (did, $type$, $escore(did,e)$, $count$) }
Example : Saddam Hussein : { (575, person, 0.332, 3), (810, person, 0.341, 3), (856, person, 0.315, 2), (881, person, 0.331, 2) }.

L'index EI est utilisé pour trouver les documents relatifs aux entités dans le cas de la recherche par entités, il est utilisé également pour trouver les documents relatifs aux entités une fois l'ensemble R construit. Il est utilisé aussi dans le cas la diversification par types.

– **DI** (Document Index) : Cet index contient les entités d'un document et sa catégorie. Open Calais est utilisé pour l'extraction des catégories des documents avec leurs scores. Open Calais extrait une ou plusieurs catégories mais dans notre travail nous considérons la catégorie qui a le score le plus élevé.
Cet index fait correspondre à chaque document $d \in D$, les entités $d.entities$ qui apparaissent dans ce document ainsi que sa catégorie c avec son score $cscore$.
Nous aurons, d : { ($d.entities$, c, $cscore(did,c)$) }
Par exemple, 575 : { { Washington, white house... }, (politics, 0.91)}.
L'index DI sera utilisé pour la diversification par catégories et pour trouver les entités des documents.

La Figure 5.5 suivante présente l'étape de création des index EI et DI.

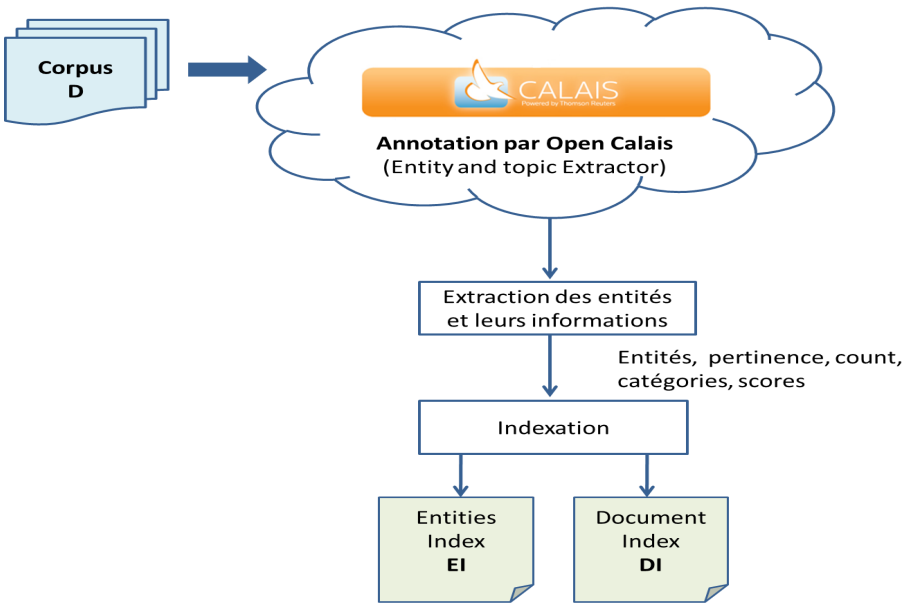

Figure 5.5: Création d'index d'entités "EI" et de documents "DI"

L'algorithme (*Algorithm 2*) suivant représente la création de l'index des entités "EI" et l'index de documents "DI" selon les étapes présentées dans la Figure 5.5.

Algorithme 2 : Création EI et DI

Input : D ; Corpus de documents
Output: EI (Entities Index), DI (Document Index)

```
2.1  begin
2.2      /*1ère étape : création de DI*/
2.3      créer_fichier(DI) ;
2.4      foreach d in D do
2.5          /*Faire pour tous les documents lors du lancement d'Open Calais et JRCNames avec
             le corpus*/
2.6          écrire(did, c, cscore, DI) ;
2.7          /*Ecrire dans DI le nom du document avec sa catégorie c*/
2.8      EndFor
2.9      /*2ème étape : création de EI*/
2.10     créer_fichier(EI) ;
2.11     /*Créer le fichier Entity_Index*/
2.12     foreach e in E do
2.13         /*Pour toutes les entités extraites lors du lancement d'Open Calais et JRCNames avec
             les documents du corpus*/
2.14         if (e.existe(EI)) then
2.15             /*Si l'entité e a été déjà trouvée (existe dans EI)*/
2.16             ajouter(did, type, escore, count, EI) ;
2.17             /*Ajouter le document contenant l'entité et le score et le nombre d'apparition
                 count extrait par le système à l'entité déjà existante dans EI*/
2.18         else
2.19             écrire(e, did, type, escore, count, EI) ;
2.20             /*Ecrire l'entité, son document, type, score et count dans EI*/
2.21         EndIf
2.22     EndFor
2.23 End
```

La création des deux index EI et DI est réalisée d'une manière parallèle (lors de la phase d'annotation) puisque c'est l'annotateur qui extrait les différentes informations de ces deux index.

5.5 Phase de traitements en ligne

Le traitement en ligne consiste à traiter la requête et retourner des résultats aux utilisateurs. Il est décrit dans la Figure 5.6.

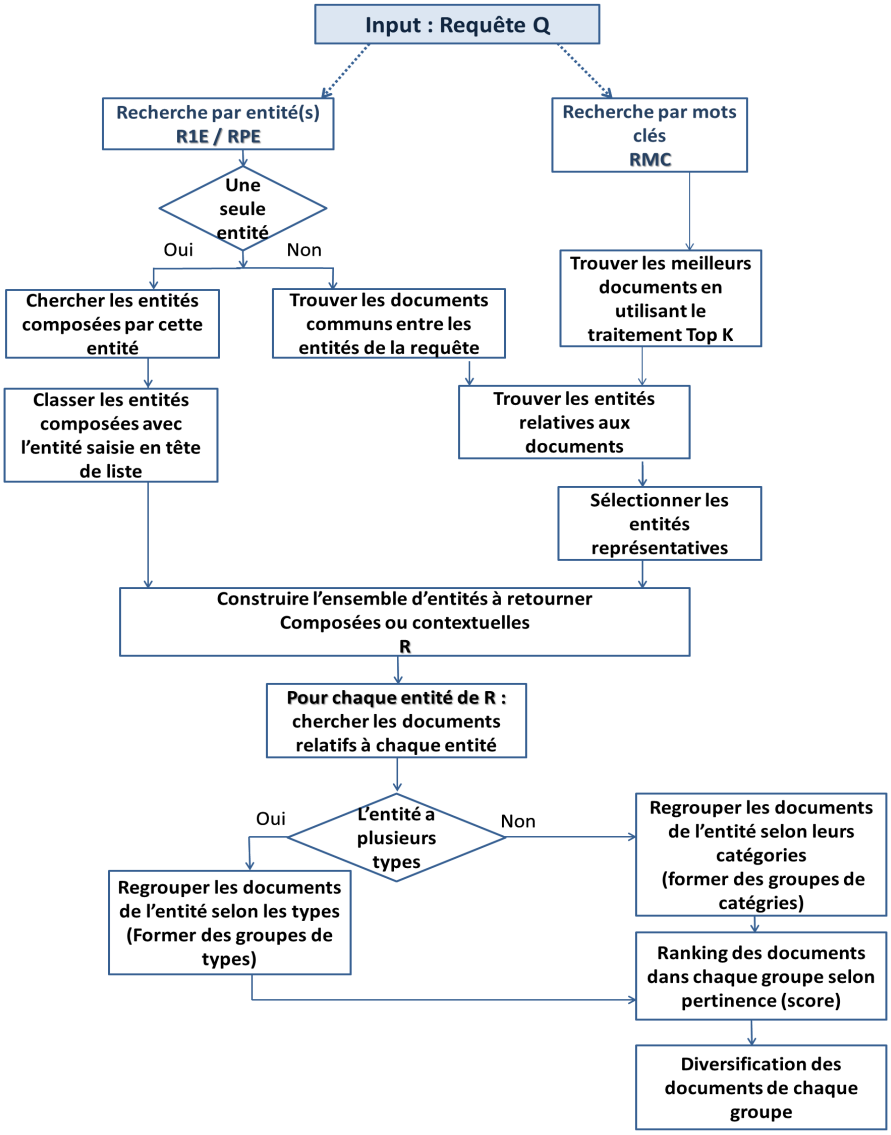

Figure 5.6: Traitement en ligne

D'après la Figure 5.6, le traitement de la requête en ligne est le suivant :

– La requête peut être constituée d'entités ou de mots clés.

– Pour une requête d'entités, s'il s'agit d'une seule entité (R1E), le traitement à

faire est de trouver les entités composées et de construire l'ensemble R.
L'ensemble R sera formé par les entités composées (contenant l'entité de la requête) ainsi que l'entité de la requête en tête de liste.

– Pour une requête d'entités, s'il s'agit de plusieurs entités (RPE), le traitement à faire est de trouver les documents communs entre les entités de la requêtes.
La construction de l'ensemble R est commencée en cherchant les entités relatives aux documents communs. Ces entités sont filtrées pour garder les entités représentatives.
L'ensemble R sera formé par les entités pertinentes (entités de la requêtes et les entités qui répondent à la requête) ainsi que les entités contextuelles (apparaissant dans le contexte de la requête).

– Pour une requête formée de mots clés, la première étape consiste à trouver les meilleurs documents qui répondent à la requête en utilisant le traitement Top K (ce traitement sera détaillé dans la section suivante). Il s'agit ensuite de sélectionner les entités relatives aux documents trouvés. Parmi ces entités, seules les entités représentatives sont sélectionnées pour construire l'ensemble R.
L'ensemble R sera constitué d'entités pertinentes et contextuelles.

– Pour chaque entité de R (quel que soit le type de la requête), la première étape consiste à accéder à l'index EI pour récupérer les documents contenant cette entité.
– Si l'entité a plusieurs types, les documents ayant un même type sont regroupés ensemble (former des groupes de types).
– Si l'entité a un seul type, les documents ayant une même catégorie sont regroupés ensemble (former des groupes de catégories).
– Classer les documents de chaque groupe selon le score ($score(d, Q)$).
– Diversifier les résultats en sélectionnant des documents de chaque groupe afin de maximiser la diversité tout en considérant la pertinence.
– Le résultat est une liste d'entité R avec un ensemble de documents pour chaque entité.

5.5.1 Algorithmes des traitements en ligne

Dans ce qui suit, nous présentons nos algorithmes de traitement en ligne.

L'algorithme Traitement_requête (Algorithme 3) est l'algorithme principal. Il a en entrée la requête de l'utilisateur Q et donne en sortie l'ensemble d'entités R avec la liste de documents diversifiés S de chaque entité.

Algorithme 3 : Traitement_requête

```
   Input  : Q /*La requête*/
   Output: R[ ] /*Ensemble d'entités*/,
           S[r][ ] /*Les listes des documents diversifiés des entités, r est la taille de l'ensemble
   R*/
3.1 begin
3.2     /*R[ ] La liste des entités relatives à trouver*/
3.3     R[ ] ← Recherche_entités(Q) ; /*Algorithme 4*/
3.4     foreach e in R do
3.5         e.docs[ ] ← EI.Accès(e) ;
3.6         /*Récupérer les documents relatifs à e*/
3.7         S[eid][ ] ← Recherche_de_documents(e,e.docs[ ]) ; /*Algorithme 5*/
3.8     EndFor
3.9 End
```

Pour trouver la liste d'entités R (pertinentes, composées ou contextuelles), un premier traitement est appliqué à la requête (ligne 3.3 : recherche d'entités, appel de l'Algorithme 4).

Les détails du traitement sont donnés dans l'*Algorithme 4*.

Après récupération de la liste des entités R, le même traitement sera appliqué à toutes les entités trouvées pour que l'utilisateur puisse explorer les résultats de n'importe quelle entité de R (de la ligne 3.4 à la ligne 3.7).

Après récupération des documents portant sur une entité en accédant à l'index des entités EI (ligne 3.5), l'*Algorithm 5* est appelé pour trouver les résultats, c'est-à-dire, les documents S diversifiés et pertinents de chaque entité (ligne 3.7).

L'algorithme Recherche_entités (Algorithme 4) est le suivant. Il est appelé par l'algorithme principal (Algorithme 3) et a en entrée la requête Q. La liste d'entités R est retournée en sortie et est trouvée selon le type de la requête.

Algorithme 4 : Recherche_entités

```
Input  : Q /*Les termes de la requête*/
Output: R[] /*La liste des entités relatives trouvées*/

4.1  begin
4.2      if (Q.requête_d'entités() == Vrai) then
4.3          if (Q.nbr_entités() == 1) then
4.4              R ← CMPentities(Q) ; /*Chercher les entités composées par l'entité*/
4.5              R ← Limiter(R, seuil) ; /*R est limité et contient Q en tête de liste*/
4.6          else
4.7              /*Q est requête RPE. Nous utilisons une liste DS[ ] pour les documents*/
4.8              DS ← CHERCHER_DOCS_COMMUNS(Q, EI) ;
4.9              /*Trouver les documents DS répondant à la requête en utilisant l'index EI*/
4.10             relatedEntities(Q) ← SÉLECTION_REPRÉSENTATIVES (DS, DI, EI) ;
4.11             /*Trouver les entités représentatives de DS en utilisant DI et EI*/
4.12             R ← Ordonner(DS, relatedEntities(Q)) ; /*Ordonner selon l'ordre de DS*/
4.13         EndIf
4.14     else
4.15         /*Q est une requête de mots clés*/
4.16         DS ← TopK_docs(Q, KI) ; /*Threshold Algorithm*/
4.17         /*Trouver les meilleurs documents qui répondent à la requête en utilisant l'index KI*/
4.18         entities ofTopK(Q) ← SÉLECTION_REPRÉSENTATIVES (DS, DI, EI) ;
4.19         /*Trouver les entités représentatives des Top K en utilisant DI et EI*/
4.20         R ← Ordonner(DS, entities ofTopK(Q)) ; /*Ordonner selon l'ordre de DS*/
4.21     EndIF
4.22     Retourner (R) ;
4.23 End
     procedure CHERCHER_DOCS_COMMUNS(Q, EI)
     foreach e in Q do
         docs[e] ← EI.Accès(e) ; /*Récupérer docs les documents de l'entité e*/
     EndFor
     DS ← intersect(docs) ;
     /*DS contient les documents communs à toutes les entités de Q obtenus par intersection*/
     if (DS.vide( ) == Vrai) then
         DS ← Relaxer_intersect(docs) ; /*Si DS est vide, relaxer pour retourner des résultats*/
     EndIf
     Retourner(DS) ;

     end procedure

     procedure Sélection_représentatives(DS, DI, EI)
     foreach d in DS do
         foreach e in DI.Accès(d) do
             if (EI.Accès(e).count > 1) then
                 Entités[d] ← e ;
                 /*Si une entité e apparait plus d'une fois, elle est prise*/
             EndIf
         EndFor
         if (Entités[d].vide( ) == Vrai) then
             Entités[d] ← Relaxer(DI.(d.entities)) ; /*Si count == 1, prendre les entités d.entities
             de d en résultat*/
         EndIf
         R ← Entités[d] ;
     EndFor
     Retourner(R) ;

     end procedure
```

Pour une requête d'entités, si l'utilisateur cherche par une seule entité (R1E), les entités composées par cette entité sont recherchées (ligne 4.4).

Les entités sont retournées selon l'emplacement de l'entité de la requête. Nous supposons que l'ordre : entités commençant, contenant l'entité de la requête au milieu puis finissant par l'entité de la requête, peut être le plus intéressant, afin de permettre à l'utilisateur de mieux explorer les résultats. Pour être retournées dans R, les entités composées sont limitées par un seuil (ligne 4.5). Les entités composées sont trouvées en utilisant l'index des entités EI.

Si l'utilisateur cherche par plusieurs entités (RPE), les documents qui répondent à la requête seront trouvés (ligne 4.8), ce sont les documents portant sur toutes les entités de la requête (voir la procédure "Chercher_Docs_Communs").

Les entités R relatives à la requête sont ensuite recherchées (ligne 4.10). R est formé dans ce cas par les entités représentatives de $relatedEntities(Q)$ (voir la procédure Sélection_Représentatives), les entités qui apparaissent dans l'ensemble de documents DS qui répondent aux entités de la requête Q (l'index DI et EI sont utilisés). Les entités sont ensuite ordonnées selon l'ordre des documents dans lesquels elles apparaissent (ligne 4.12).

Pour une requête de mots clés (RMC), les entités relatives sont trouvées en appliquant le traitement Top K avec les mots clés de la requête (ligne 4.16). Nous définissons alors le problème de trouver les top-k documents qui satisfont la requête. L'index KI est utilisé.

Pour le traitement des requêtes top-k, nous avons choisi d'utiliser l'algorithme TA (*Threshold Algorithm*) [59]. Cet algorithme défini par Fagin et al. est l'un des meilleurs algorithmes de traitements dans le contexte centralisé et le plus efficace pour résoudre les requêtes Top-k sur des listes ordonnées [61]. L'algorithme TA est donné dans l'annexe (Section A.2).

Quand la requête est un ensemble de mots-clés, nous utilisons le tf*idf pour le *scoring* des documents. Et nous définissons :

$score(d, Q)$ tels que d un document et Q une requête est la somme des $tf * idf(d, t)$ des termes t pour un document d où t est un terme dans la requête Q.

Les entités de R seront les entités représentatives des Top K meilleurs documents $entitiesofTopK(Q)$. L'index DI et EI sont utilisés pour trouver ces entités (voir la procédure Sélection_Représentatives).

Les entités sont ordonnées selon l'ordre des documents dans lesquels elles apparaissent (ligne 4.20). Pour les deux types de requêtes (RPE et RMC), nous définissons la sélections des entités représentatives comme suit :

Parmi les documents les plus pertinents (Top K dans le cas de recherche par mots clés 'RMC' et documents communs dans la recherche par entités 'RPE'), les entités représentatives sont celle ayant le plus grands 'count', c'est-à-dire, qu'elles apparaissent fréquemment dans un document (le représentent bien).

Pour un ensemble d'entités e_1, e_2, ..., e_n, les entités représentatives sont choisies selon le *count*. Nous avons considéré les entités apparaissant plus d'une fois dans un document.

Les entités R sont à la fin retournées (ligne 4.22). Pour ces entités, le bon ordre de pertinence (ranking) est généré automatiquement, vu que ces entités sont extraites par $entities\_of(TopK(Q))$, c'est-à-dire, les entités représentatives des meilleurs documents dans le cas des requêtes RMC ou elles sont extraites par $relatedEntities(Q)$, c'est-à-dire, les entités des documents communs (choisis également selon pertinence) dans le cas des requêtes RPE.

L'algorithme Documents_résultats (Algorithme 5) est appelé pour trouver les documents relatifs aux entités. Cet algorithme est le suivant :

Algorithme 5 : Documents_résultats

```
Input  : e /*Entité*/, e.docs[ ]/*Documents de e*/
Output: S[eid][ ] /*Les documents diversifiés et pertinents de l'entité e, eid est l'identifiant de
        l'entité e*/

5.1  begin
5.2      if (e.hastypes() == Vrai) then
5.3          annotations ← e.types(EI) ;/*e a plusieurs types*/
5.4          liste_groupe ← Null ;
5.5          /*La liste liste_groupe est dans ce cas groupe_type (groupe de types) ;
5.6      else
5.7          /*e.hastypes() est Faux*/
5.8          annotations ← DI.categories(e.docs) ;
5.9          liste_groupe ← Null ;
5.10         /*La liste liste_groupe est dans ce cas groupe_cat (groupe de catégories) ;
5.11     EndIf
5.12     Classer(e.docs[ ], score) ; /*score = escore pour les types et cscore pour catégories*/ ;
5.13     S[eid][ ] ← GROUPES_DIV(d.e, annotations, liste_groupe) ;
5.14 End
     procedure GROUPES_DIV(d.e, annotations, liste_groupe)
     foreach d in e.docs[ ] do
         if (liste_groupe.has(d.e, annotations) == Faux) then
             liste_groupe = annotations[d.e] ;
             /*Mettre à jour liste_groupe par le nouveau type ou catgorie */
             S[eid][compteur] ← ajouter(d) ;
             compteur++ ;
         else
             /*liste_groupe.has(d.e, annotations) est Vraie*/
             /*Le type ou la catégorie existe déjà dans les résultats, vérifier la pertinence*/
             PERTINENCE(e, d) ;
         EndIf
     EndFor
     end procedure

     procedure PERTINENCE(e, d)
     if (d.score ≥ θ) then
         /*θ : seuil minimal, fixé par expériences à (0.4*count) pour la diversification par types et à
         0.7 pour la diversification par catégories*/
         S[eid][compteur] ← ajouter(d) ;
         compteur++ ;
     EndIf
     end procedure
```

L'algorithme 5 est commencé par la vérification du nombre de types de l'entité e pour savoir quel traitement entreprendre.

Si l'entité a plusieurs types (ligne 5.2), l'ensemble de types de l'entité e sera attribué à une liste *annotations* en accédant à l'index EI et une nouvelle liste nommée *liste_groupe* est initialisé, elle contiendra dans ce cas la liste des types *groupe_type* (de la ligne 5.2 à la ligne 5.5).

Si l'entité n'a qu'un seul type (ligne 5.6), l'ensemble des catégories des documents de l'entité e sera attribué à la liste *annotations* en accédant à l'index DI et la liste *liste_groupe* est initialisée pour recevoir après les catégories *groupe_cat*. Une même liste (*liste_groupe*) est utilisée pour les deux cas (*groupe_type* et *groupe_cat*) pour unifier le traitement.

Un classement des documents de l'entité e est ensuite effectué (ligne 5.12). Ceci facilite la sélection des documents pertinents. Cette méthode utilise les scores définis précédemment (Section 4.5.2).

La diversification est appliquée par la suite pour retourner pour une entité e un ensemble de documents S. La diversification est effectué en appelant la procédure GROUPES_DIV (ligne 5.13).

La procédure GROUPES_DIV vérifie pour chaque document d s'il est unique pour le type de l'entité ou la catégorie. Si c'est le cas, il sera retourné en résultat car ceci augmente la diversité. Sinon sa pertinence est vérifiée.

Ce traitement est effectué en vérifiant si *liste_groupe* ne contient pas le type ou la catégorie du document d en cours qui sont stockés dans *annotations* (traitement effectué précédemment, ligne 5.3 et 5.8).

Si ceci est vérifié, *liste_groupe* sera mise à jour par le nouveau type ou par la nouvelle catégorie et le document d sera ajouté à documents S de l'entité e (pour maximiser le diversité).

Sinon, si *liste_groupe* contient déjà le type ou la catégorie du document d en cours, la pertinence du document est considérée en faisant appel à la procédure PERTINENCE.

La procédure PERTINENCE ajoute le document d à S si son score dépasse un seuil. Un seuil minimal a été fixé en consultant les résultats de l'annotation. Le *escore* (types) est fixé à 0,4 et le *cscore* (catégories) est fixé à 0,7.

5.6 Discussion

Dans notre approche, nous nous basons principalement sur l'annotateur Open Calais qui identifie les entités existantes dans le corpus afin de les stocker dans des index, pour ensuite diversifier les entités qui répondent aux différents types de requêtes selon les interprétations possibles (entités composées ou entités contextuelles).

Nous nous basons aussi sur l'annotateur dans un deuxième temps pour diversifier les documents des entités trouvées selon les types qu'elles peuvent avoir ou selon les catégories des documents.

Notre définition de diversification se base sur la formation de groupes de documents selon les types et les catégories. En formant ces groupes de documents, l'appartenance de documents à chaque groupe suivant le type ou la catégorie peut être utilisée pour faire la distinction entre les documents et appliquer la diversité sans calculer les similarités ou traiter le contenu des documents.

La diversification est donc réalisée et est maximisée en construisant *groupe_type* et *groupe_cat* qui permettent de considérer tous les types et les catégories existantes. Les documents les plus pertinents (dépassant un seuil) de chaque groupe sont pris comme résultat pour assurer la pertinence.

Nous pouvons dire que notre double diversification se base sur les annotations et sur les index ce qui résout le problème du choix à faire pour diversifier.

Au moins un document de chaque groupe est retourné pour maximiser la diversité et les documents les plus pertinents de chaque groupe sont pris dans le résultat.

Notre approche présente néanmoins certains challenges : son efficacité dépend de certains paramètres, a savoir :

– Pour les requêtes de mots clés (RMC) :

Le seuil du nombre de documents pertinents (le k des Top documents).

– Pour les requêtes d'entités (RPE) :

Le seuil des documents communs si le corpus est très grand. Si le corpus est petit, le nombre de documents communs est restreint, il n'est donc pas nécessaire de définir un seuil.

– Pour les requêtes d'entités (RPE) et les requêtes de mots clés (RMC) :

Un seuil pour les entités représentatives à prendre dans R.

– Pour les requêtes formées d'une seule entité (R1E) :

Le seuil du nombre d'entités composées à retourner.

– Pour les trois types de requêtes (R1E, RPE, RMC) :

Un seuil pour la diversification par types et par catégories. Ces deux seuils ont pu être choisis et fixés en consultant les résultats de l'annotation. Nous avons remarqué que lorsque le score d'extraction du type dépasse 0.4, il est généralement juste. Nous avons remarqué aussi que lorsque le score d'extraction de la catégorie dépasse 0.7, il s'agit de la bonne catégorie du document. Ces deux seuils ne présentent donc pas de très grands inconvénients.

Le choix des seuils est dans la plupart des cas arbitraire. Néanmoins, les seuils à fixer sont au pire deux seuils pour un type de requête.

6

Expérimentations

6.1 Introduction

Nous avons implémenté un prototype reflétant le fonctionnement de notre système. Ce prototype permet de traiter les requêtes des utilisateurs. Il offre à l'utilisateur le choix entre une requête de recherche par entités ou une requête de recherche par mots clés. Les résultats retournés sont des entités diverses et des documents divers et pertinents relatifs aux entités trouvées.

Nous avons effectué une série d'expériences afin d'évaluer les performances de notre approche et la pertinence des résultats.

Nous commençons d'abord par décrire l'environnement dans lequel nous avons réalisé notre prototype puis nous discutons et analysons les expériences réalisées.

6.2 Environnement de développement

Le prototype a été implémenté en java et les tests ont été réalisés sur une machine Intel Core i5, avec une vitesse de 2.5 GHZ, dotée d'une capacité mémoire de 4GB de RAM sous Windows 7.

6.2.1 Corpus utilisés

L'évaluation d'un système est réalisée en utilisant une collection de test : typiquement, un corpus de plusieurs milliers de documents et quelques dizaines de requêtes, auxquelles sont associés des jugements de pertinence utilisateur, établis par des experts [91].

Deux collections de données réelles ont été utilisées pour les expériences, le corpus 20NewsGroups [1] et une collection extraite du corpus Reuters-21578 [2].

1. http ://qwone.com/ jason/20Newsgroups/
2. http ://www-igm.univ-mlv.fr/ mconstan/enseignement/m2pro/tal/tal-td2/node1.html

Corpus 20 NewsGroups

Ce corpus [3] est une collection d'environ 20 000 messages, collectés de 20 différents newsgroups (environ 1000 messages par groupe). Il a été choisi pour sa richesse en entités et la variété de thèmes (catégories) de ses documents.

Collection de Reuters

Ce corpus est une collection de documents textuels téléchargée du site de l'Institut d'informatique et d'électronique Gaspard-Monge [4]. Cette collection de textes a été extraite automatiquement de la collection Reuters-21578 [5]. Chaque texte est classé dans une ou plusieurs des 135 catégories utilisées dans la collection. Cette catégorisation est donnée dans un fichier nommé categorisation.txt où chaque ligne correspond à la catégorisation d'un texte. Ce corpus contient environ 20 000 fichiers et a été choisi pour sa richesse et la variété des catégories aussi.

6.2.2 Indexation et annotation

L'architecture du système est composée de deux parties, une partie hors ligne et une partie en ligne. Le prototype réalisé permet dans un premier temps d'indexer le corpus de documents et de l'annoter afin de créer les index présentés préalablement (KI, EI, DI).

Pour l'indexation, nous avons utilisé un logiciel libre de recherche d'information Lucene [6]. Pour l'annotatin nous utilisons Open Calais API.

Indexation par Lucene

Lucene est une bibliothèque de logiciels libres (*Open Source*) largement utilisée dans la recherche d'information. Elle est très performante et peut servir de base pour développer tout type de moteur de recherche.

3. http ://qwone.com/ jason/20Newsgroups/
4. http ://www-igm.univ-mlv.fr/ mconstan/enseignement/m2pro/tal/tal-td2/node1.html
5. http ://www.daviddlewis.com/resources/testcollections/reuters21578/
6. Apache Software Foundation, "Lucene," http ://lucene.apache.org/.

Lucene n'est pas une application complète mais un ensemble de classes java (packages) disponible en ligne [7]. Cette bibliothèque offre la possibilité d'indexer et de rechercher le texte [92].

Indexation par Hadoop

Avec l'explosion du volume des données et la nécessité de la présentation rapide des résultats, une parallélisation des traitement améliorera sans doute les performances. Pour cette raison, nous avons pensé à introduire le modèle MapReduce dans nos algorithmes pour effectuer certaines tâches d'une manière parallèle, plus d'informations sont fournies en annexe (Section A.1).

Plusieurs implémentations du modèle MapReduce existent, par exemple, le framework libre *Hadoop* [8].

Nous avons commencé par effectuer l'indexation en utilisant Hadoop. Nous avons ensuite calculé le temps nécessaire à l'indexation par Hadoop et le temps de l'indexation par Lucene.

Les résultats sont présentés dans la Figure 6.1. Nous avons varié la taille du corpus (de 500 à 20 000 fichiers) et calculé le temps d'indexation pour chaque cas.

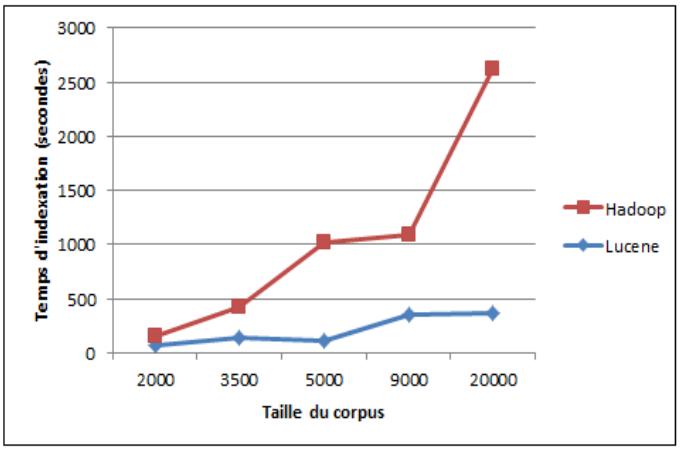

Figure 6.1: Variations du temps d'indexation en fonction de la taille du corpus

D'après la Figure 6.1, il est clair que Lucene prend beaucoup moins de temps à

7. https ://code.google.com/p/luke/source/checkout
8. http ://searchcloudcomputing.techtarget.com/definition/Hadoop

indexer les corpus utilisés, ceci nous a permis de déduire que l'utilisation de Hadoop pour un corpus de taille raisonnable n'est pas pertinent.

Quelques posts récents [9] [10] [11] publiés dans des blogs de développeurs connus critiquent l'utilisation inappropriée de Hadoop.

L'un des post présente des cas où Hadoop peut être utilisé et d'autres cas où il n'est pas intéressant d'utiliser Hadoop [93]. Nous citons dans ce qui suit les cas les plus importants [93] :

Cas où Hadoop peut être utilisé :

– Les ensembles de données sont vraiment grands : des téra ou des pétabytes (ne pas penser à Hadoop si les données sont de l'ordre de méga ou gigabytes).

– Les données sont de natures diverses : données structurées, non structurée, etc.

– Constuire une "*Enterprise Data Hub*" : un nouveau modèle de gestion de données pour le contexte Big Data.

– Si des donnés utiles sont jetées : Hadoop a la capacité de stocker des pétaoctets de données. Si des données potentiellement utiles sont jetées faute du coût d'archivage, la mise en place d'un cluster Hadoop permettra de conserver ces données.

Cas où Hadoop ne doit pas être utilisé :

– Les réponses doivent être retournées rapidement.

– Les requêtes sont complexes.

– La nécessité de l'accès aléatoire aux données et la nécessité de l'intéractivité.

– Le stockage de données sensibles.

9. When to use Hadoop (and when not to) : http ://www.citeworld.com/article/2462886/big-data-analytics/when-to-use-hadoop-and-when-not-to.html

10. To Hadoop or Not to Hadoop ? :
http ://www.thoughtworks.com/insights/blog/hadoop-or-not-hadoop

11. Don't use Hadoop - your data isn't that big :
https ://www.chrisstucchio.com/blog/2013/hadoop_hatred.html

Annotation

Comme nous l'avons dit précédemment, dans ce travail nous utilisons l'API d'Open Calais [12] ; "j-calais" (*Java interface to the OpenCalais API*) .

L'API Calais a été renforcée par l'outil d'extraction d'entités nommées et leurs variantes : JRCNames [13]. Cet outil contient une collection de noms propres qu'on trouve fréquemment plus des noms téléchargés de Wikipedia [14] ainsi que des noms vérifiés manuellement [94]. Cette liste est accompagnée par un logiciel qui trouve ces noms dans le texte et retourne leurs IDs, les variantes (exemple, NATO et OTAN) ainsi que le type [15] (les noms et types extraits sont nécessairement justes puisque l'extraction se fait en se basant sur la liste déjà construite).

6.3 Evaluation de la pertinence

Pour évaluer la pertinence, nous avons effectué trois tests utilisateurs, avec l'aide d'un groupe d'étudiants (15 étudiants) de différentes spécialités : informatique, médecine, architecture, etc.

Les réponses de ces testeurs sont obtenues à partir de formulaires que nous avons mis en ligne en utilisant l'outil de création de sondages "*SurveyMonkey*" [16]. Des captures d'écran de ces formulaires sont présentés avec les tests dans cette section.

La pertinence est calculée à partir des résultats des formulaires pour :

1. Les entités retournées (section 6.3.1) : pour chaque requête, calculer le nombre des entités très pertinentes, le nombre d'entités qui peuvent être pertinentes (entités contextuelles ou composées) et le nombre d'entités qui ne sont pas du tout pertinentes et ensuite calculer la moyenne et le pourcentage par rapport au nombre total des entités retournées.

12. http ://www.opencalais.com/applications/j-calais-java-interface-opencalais-api
13. https ://ec.europa.eu/jrc/en/language-technologies/jrc-names
14. http ://www.wikipedia.org/
15. Cet outil contient les noms de plus de 205 000 entitiés distinctes : personnes, organisations...
16. https ://fr.surveymonkey.com/

2. La diversification des documents (section 6.3.2) : les types et les catégories trouvées pour les documents. Pour une entité retournée, si les documents sont diversifiés par types, les types retournés sont-ils pertinents (combien le sont, combien sont moyens et combien sont mauvais). Si les documents sont diversifiés par catégories (combien de catégories sont bonnes, combien sont moyennes, combien sont mauvaises). La moyenne est calculée et transformée en pourcentage.

Les résultats d'une recherche d'informations sont généralement jugés bons ou mauvais par rapport à un objectif. Les objectifs précisés aux testeurs sont les suivants :

1. Pertinence des entités : l'objectif est de mesurer la pertinence des entités retournées pour les trois types de requêtes (R1E, RPE, RMC).
2. Pertinence des documents diversifiés : l'objectif est de savoir si les types et les catégories des documents retournés sont pertinents pour avoir de la diversité.

6.3.1 Pertinence des entités retournées pour R1E, RPE et RMC

Comme c'est mentionné précédemment, ce test sert à mesurer la qualité des résultats obtenus, pour vérifier si les résultats satisfont l'utilisateur du point de vue pertinence des entités retournées (pour les trois types de requêtes R1E, RPE et RMC), c'est-à-dire, de mesurer la pertinence des entités retournées (l'ensemble R).

Nous avons demandé aux testeurs de numéroter de 1 à 3 les réponses obtenues par notre prototype pour 5 différentes requêtes en utilisant les deux corpus 20NewsGroups et Reuters [17].

La signification de la numérotation est la suivante : (1) très pertinent, (2) peut être pertinent et (3) pas du tout pertinent, et ce pour les trois types de requêtes R1E, RPE, RMC. Les réponses ont été agrégées et la moyenne a été calculée et transformée en pourcentage pour chaque requête.

17. Les requêtes et entités choisies pour faire les tests ont des réponses dans les deux corpus 20NewsGroups et Reuters.

`Requête R1E :` Pour le premier type R1E, les requêtes posées sont présentées dans le tableau 6.1.

Les figures 6.3, 6.4 présentent le pourcentage de pertinence par rapport au nombre d'entités composées trouvées dans le corpus 20NewsGroups et Reuters respectivement.

Le nombre d'entités composées trouvées pour les requêtes testées (l'ensemble R) est présenté également dans le tableau 6.1 pour les deux corpus.

La Figure 6.2 suivante présente un exemple de formulaire pour ce type R1E (avec résultats du corpus 20NewsGroups). Ce formulaire est disponible en ligne à l'adresse : https ://fr.surveymonkey.com/s/BQGSNXT

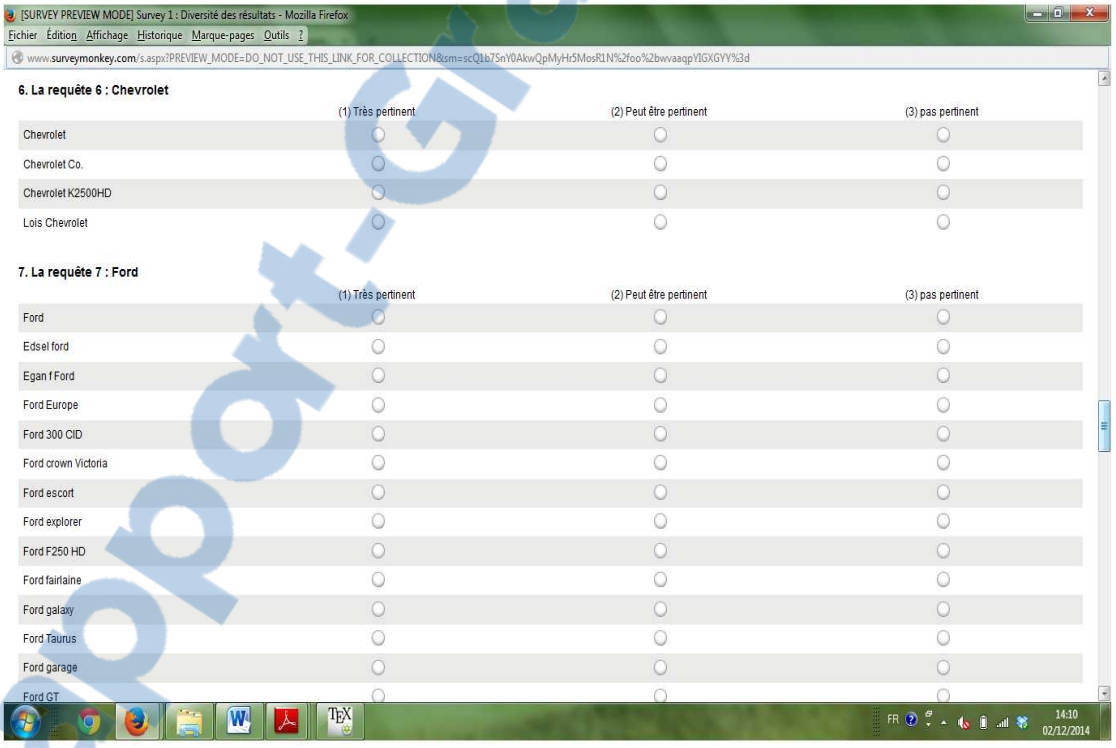

Figure 6.2: Survey 1. Exemple de formulaire des requêtes R1E

Requêtes	Corpus 20NewsGroups Nombre d'entités composées trouvées	Corpus Reuters Nombre d'entités composées trouvées
Chevrolet	4	6
Lincoln	11	15
America	19	10
Ford	25	25
Cancer	29	74

Table 6.1: Requêtes R1E posées

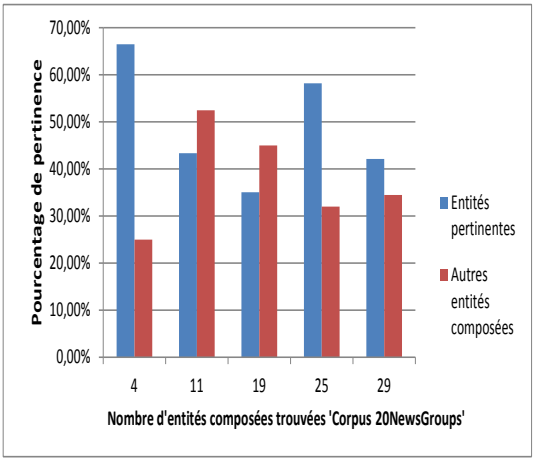

Figure 6.3: "Corpus 20NewsGroups" Pourcentage de pertinence des entités composées trouvées (R1E)

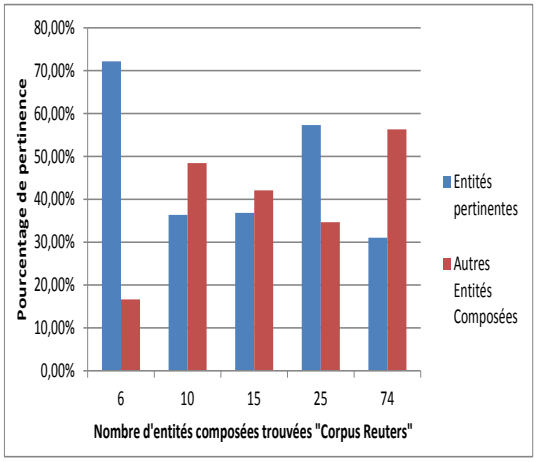

Figure 6.4: "Corpus Reuters" Pourcentage de pertinence des entités composées trouvées (R1E)

Nous remarquons d'après les figures 6.3 et 6.4 que dans les deux cas, le pourcentage des entités très pertinentes peut atteindre 70 % (requête 1 : Chevrolet) et peut dépasser 55 % (requête 3 : Ford). Mais même quand ce pourcentage diminue à environ 30 % (cas du Corpus Reuters, pour la requête "cancer"), un autre pourcentage qui est celui d'autres entités composées augmente (Voir Figure 6.4), ce qui revient à dire que lorsque l'utilisateur ne note pas toutes les entités comme pertinentes, ce qui est trouvé n'est pas faux.

Par exemple, pour la requête "cancer", une des entités trouvées est "Michigan Cancer foundation". Elle a été notée (2) (peut être pertinente) alors qu'une autre entité trouvée

”Skin cancer” est notée (1) (très pertinente). L'utilisateur cherchait alors la maladie, un autre utilisateur qui cherche les centres et fondations fera le contraire.

Donc pour les utilisateurs, certaines entités composées sont plus pertinentes que les autres, elles sont constituées par la même entité mais portent sur des éléments complètement diffèrents, ce qui explique les résultats obtenus dans les deux figures : lorsque certaines entités sont pertinentes, d'autres le sont moins (selon le besoin de l'utilisateur).

Une idée est d'affecter chaque entité retournée à l'un des trois groupes (au début : celles qui commencent par l'entité, au milieu : celle qui contiennent l'entité, à la fin : celles qui finissent par l'entité de la requête). L'utilisateur pourra ainsi explorer la catégorie qui l'interésse puisqu'il connait normalement l'emplacement de son entité. Cette solution facilite l'exploration lorsque le nombre d'entités est grand.

Nous remarquons aussi que le nombre d'entités retournées n'a pas d'influence sur la pertinence. Ce qui influe sur la pertinence c'est le sens de l'entité.

Dans la suite de ces expériences, les entités trouvées sont pertinentes si elles répondent directement à la requête, elles sont contextuelles si elles ont un lien avec le contexte de la requête ”peuvent être pertinentes”, sinon elles sont impertinentes (quand elles n'ont aucun lien avec la requête).

`Requête RPE :` Pour le deuxième type RPE, les requêtes posées sont présentées dans le tableau 6.2.

Les figures 6.6, 6.7 présentent le pourcentage de pertinence par rapport au nombre d'entités relatives trouvées dans le corpus 20NewsGroups et Reuters.

Le nombre d'entités relatives trouvées pour les requêtes testées (l'ensemble R) est présenté également dans le tableau 6.2 pour les deux corpus.

La Figure 6.5 suivante présente un exemple de formulaire pour le type RPE (avec résultats du corpus 20NewsGroups). Ce formulaire est disponible en ligne à l'adresse : https ://fr.surveymonkey.com/s/BGKYKS5

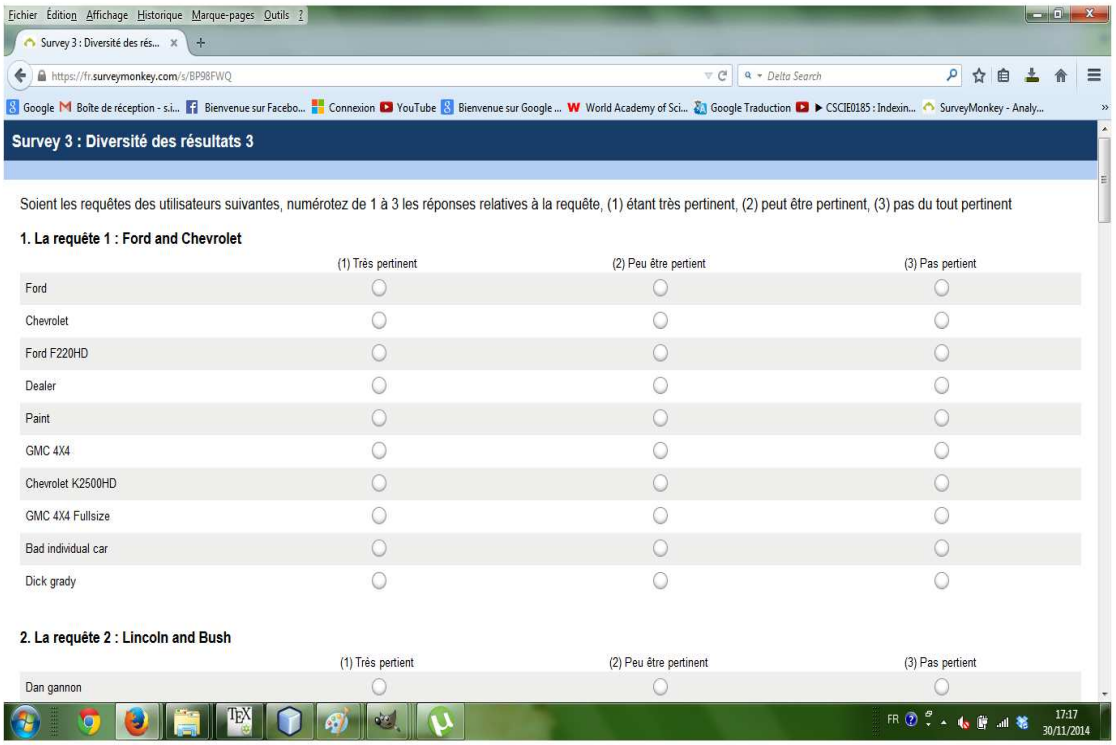

Figure 6.5: Survey 2. Exemple de formulaire des requêtes RPE

Requêtes	Corpus 20NewsGroups Nombre d'entités relatives trouvées	Corpus Reuters Nombre d'entités relatives trouvées
Ford and Chevrolet	10	8
Lincoln and Bush	11	2
Infection and Tumors	12	2
Volvo USA	14	26
Washington and Baghdad	31	47

Table 6.2: Requêtes RPE posées

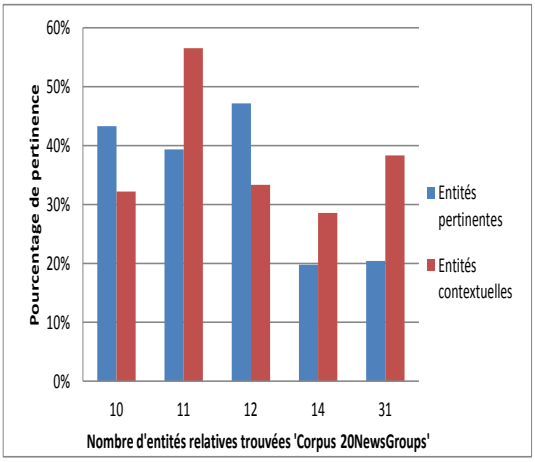

Figure 6.6: "Corpus 20NewsGroups" Pourcentage de pertinence des entités relatives trouvées (RPE)

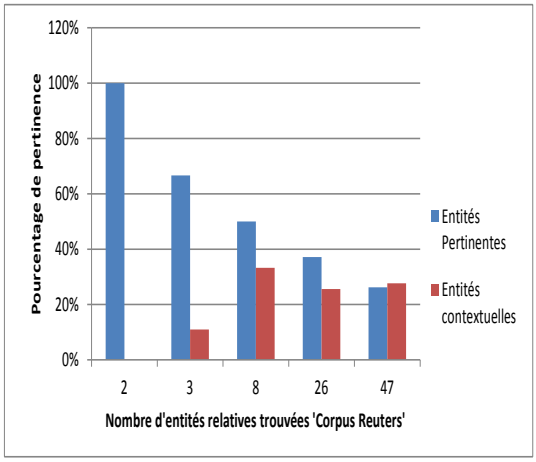

Figure 6.7: "Corpus Reuters" Pourcentage de pertinence des entités relatives trouvées (RPE)

Pour ce type de requêtes (RPE), les entités sont trouvées en cherchant d'abord les documents qui répondent à la requête.

Pour le corpus 20NewsGroups, les entités trouvées sont notées en moyenne comme étant pertinentes dans environ 35 % des cas et contextuelles dans 38 % des cas. Ces valeurs sont la moyenne des pourcentages de toutes les requêtes de la Figure 6.6.

Pour le corpus Reuters, en moyenne 55 % des cas sont notés comme très pertinents et 20 % d'entités contextuelles (Figure 6.7).

pour le corpus 20NewsGroups, environ 73 % d'entités peuvent être satisfaisantes aux utilisateurs (c'est-à-dire très pertinentes et peuvent être pertinentes) et pour le corpus Reuters, 75 % le sont. Les autres entités (le pourcentage restant) sont celles apparaissant dans les documents répondant à la requête mais qui ne sont pas pertinentes à la requête. Un meilleur filtrage est prévu dans les perspectives pour une meilleure qualité des résultats.

`Requête RMC` : Pour le troisième type RMC, les requêtes posées sont présentées dans le tableau 6.3.

Les figures 6.9, 6.10 présentent le pourcentage de pertinence par rapport au nombre d'entités relatives trouvées dans le corpus 20NewsGroups et Reuters.

Le nombre d'entités relatives trouvées pour les requêtes testées (l'ensemble R) est

présenté également dans le tableau 6.3 pour les deux corpus.

La Figure 6.8 suivante présente un exemple de formulaire pour le type RMC (avec résultats du corpus 20NewsGroups). Nous avons mis ce formulaire en ligne. Il est disponible à l'adresse : https ://fr.surveymonkey.com/s/S2C3X5N

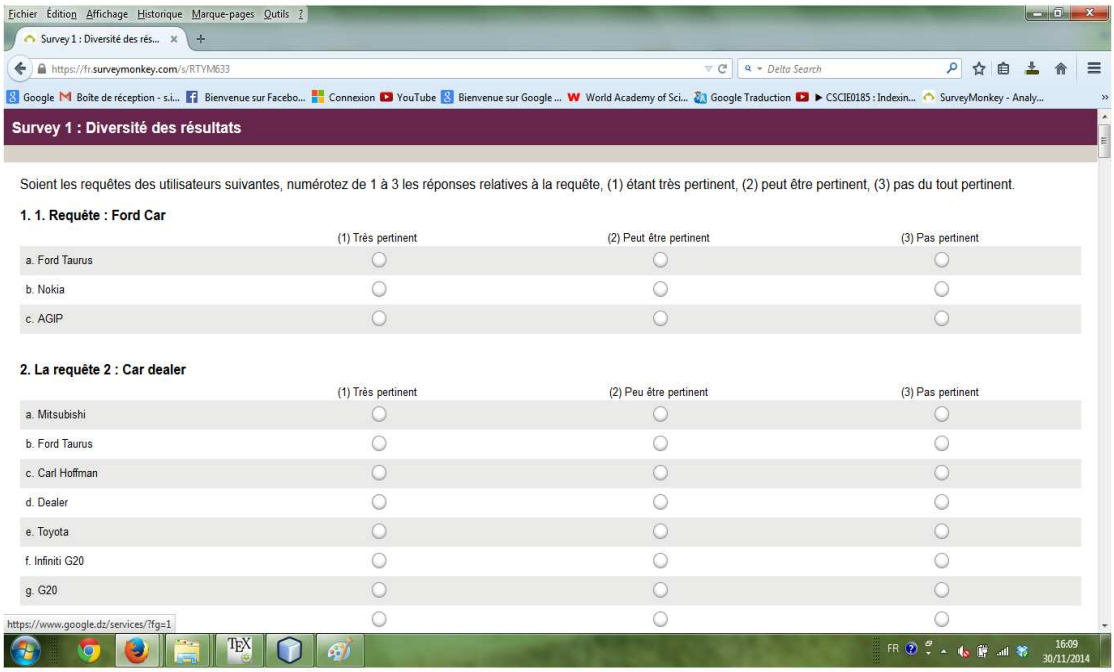

Figure 6.8: Survey 3. Exemple de formulaire des requêtes RMC

Requêtes	Corpus 20NewsGroups Nombre d'entités relatives trouvées	Corpus Reuters Nombre d'entités relatives trouvées
Ford Car	3	35
Patient disease	10	36
Car Dealer	13	50
Buy Ford	35	14
Prime Minister	42	80

Table 6.3: Requêtes RMC posées

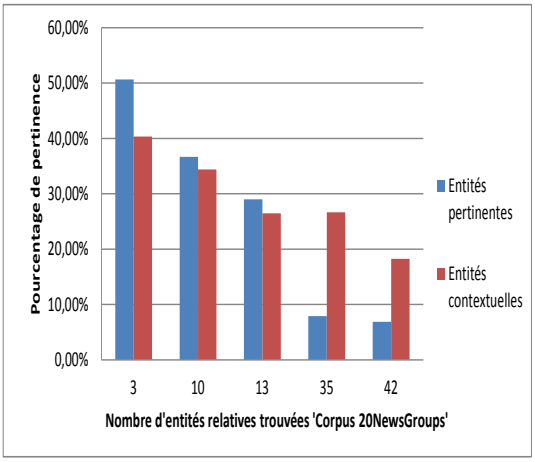

Figure 6.9: "Corpus 20NewsGroups" Pourcentage de pertinence des entités relatives trouvées (RMC)

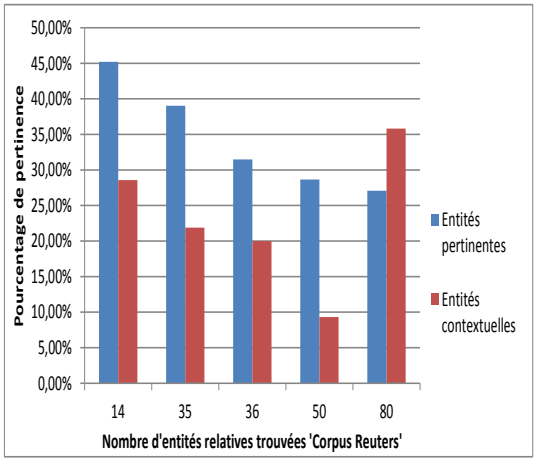

Figure 6.10: "Corpus Reuters" Pourcentage de pertinence des entités relatives trouvées (RMC)

Pour ce type de requêtes RMC et pour le corpus 20NewsGroups, les entités trouvées sont notées en moyenne très pertinentes dans environ 26 % des cas et comme contextuelles dans environ 30 %. Ces valeurs sont la moyenne des pourcentages de toutes les requêtes (voir la Figure 6.9).

Pour le corpus Reuters, en moyenne 35 % des cas sont notés comme très pertinents et 25 % d'entités contextuelles (Figure 6.10). Globalement 56 % d'entités sont satisfaisantes (très pertinentes, peuvent être pertinentes) à l'utilisateur pour le corpus 20NewsGroups et environ 60 % pour le corpus Reuters.

Les résultats sont inférieurs aux deux types de requêtes précédents car l'interprétation des mots clés est plus complexe que la recherche par entités, une amélioration de cette partie de l'algorithme sera envisagée ultérieurement.

Pour résumer, dans les figures précédentes de 6.3 à 6.10 (pertinence des entités trouvées pour les utilisateurs), les testeurs nous ont expliqué qu'ils ont défini la pertinence comme étant "quelque chose" qui les intéressent.

Lorsqu'ils ont noté les entités trouvées à (2) peut être pertinent et à (3) pas du tout pertinent, c'est soit dans les cas où d'autres entités les intéressent plus (par exemple, Ford Taurus noté (1) est plus intéressante que Gerald Ford noté (2)), soit dans les cas où ils ne connaissent pas l'entité trouvée (par exemple, dans la plupart des cas Witey

Ford, ancien joueur Américain de baseball a été noté (3)).

Dans d'autres cas, le type de l'entité recherché par l'utilisateur est inexistant dans les résultats ce qui peut rendre l'utilisateur insatisfait.

La diversité et la nouveauté dans les résultats ont été bien prises par la majorité des testeurs même si les préférences et la culture ont un impact important sur la satisfaction des requêtes. Nous pensons à enrichir les entités retournées par un résumé les décrivant pour faciliter la compréhension. Ceci pourra être fait dans un travail ultérieur.

6.3.2 Précision, Recall, MAP

Dans les systèmes d'information, la performance est calculée généralement par les mesures standard *Precision* et *Recall* [95], qui sont définis comme suit [18] :

$Precision = true\ positives\ /\ (true\ positives + false\ positives)$.

$Recall = true\ positives\ /\ (true\ positives + false\ negatives)$.

$True\ Negative$: un cas négatif identifié comme négatif.

$True\ Positive$: un cas positif identifié comme positif.

$False\ Negative$: un cas positif identifié comme négatif.

$False\ Positive$: un cas négatif identifié comme positif.

Precision indique la précision du système par rapport aux informations retournées, c'est-à-dire, le nombre d'informations correctes parmi celles extraites.

Recall indique le pourcentage des informations pertinentes que le système a retourné, c'est-à-dire, le nombre d'entités trouvées parmi celles qui sont censées être trouvées.

Si on applique la formule classique du calcul de la précsion et du recall sur nos résultats, nous aurons à calculer ce qui suit :

Précision = entités correctes trouvées / nombre total des entités trouvés.

Recall = entités correctes trouvées / nombre total des entités correctes.

18. Formules de la précision et du recall,
http ://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-unranked-retrieval-sets-1.html

Dans notre cas et en prenant les résultats de la Section 6.3.1, les entités sont considérées correctes si elles sont pertinentes pour l'utilisateur, elles sont considérées comme trouvées si elles sont retournées à l'utilisateur. Les formules de la précision et du recall deviennent alors :

Précision = entités pertinentes retournées / nombre total des entités retournées.

Recall = entités pertinentes retournées / nombre total des entités pertinentes.

Nous avons calculé la précision pour nos trois types de requêtes R1E, RPE, RMC en nous basant sur les résultats de la Section 6.3.1 précédente.

Le tableau 6.4 suivant présente la précision des résultats pour les entités pertinentes et les entités composées (cas R1E) et contextuelles (cas RPE et RMC).

Types de requêtes	Entités pertinentes	Entités composées ou contextuelles
Précision R1E	0.49	0.402
Précision RPE	0.34	0.376
Précision RMC	0.378	0.299

Table 6.4: Précision des entités

D'après les résultats du tableau 6.4, nous pouvons confirmer la bonne performance de notre système pour les trois types de requêtes.

La précision des requêtes R1E pour les entités pertinentes est égale à 0.49 et celle des entités composées est égale à 0.402. Si nous considérons que les entités composées sont aussi pertinentes, nous aurons une précision qui est égale à 0.892 ce qui représente une très bonne performance.

La précision des requêtes RPE pour les entités pertinentes est égale à 0.34 et celle des entités contextuelles est égale à 0.376. Si nous considérons que les entités contextuelles sont aussi pertinentes, nous aurons une précision qui est égale à 0.716.

La précision des requêtes RMC pour les entités pertinentes est égale à 0.378 et celle des entités contextuelles est égale à 0.402. De la même façon que les deux autres cas, si

nous considérons les entités contextuelles comme des entités pertinentes, nous aurons une précision qui est égale à 0.677 ce qui représente une bonne performance aussi.

Pour le calcul du recall nous aurons besoin de calculer le nombre total des entités pertinentes ($(true\ positives + false\ negatives)$). L'identification des $false\ negatives$, c'est-à-dire, entités correctes considérées comme incorrectes n'est pas une tâche facile puisque cette identification dépend de plusieurs facteurs, à savoir : la performance du système d'extraction (annotateur) pour l'extraction des entités ainsi que les différents seuils définis précédemment pour les trois types de requêtes.

Nous avons donc considéré un autre moyen qui résume et représente bien la courbe de la précision-recall en un seul nombre appelé : précision moyenne "*average precision, AP*" [96].

Average Precision (AP)

Comme son nom l'indique, la précision moyenne est la moyenne des valeurs de la précision pour un recall allant de 0 à 1 (toutes les valeurs du rappel entre 0 et 1). Nous avons alors :

$AP = \sum_{k=1}^{N} P(k)\ \Delta\ r(k)$

N est le nombre total des éléments, $P(k)$ est la précision de k éléments, delta $r(k)$ est le changement de rappel entre k-1 et k.

Pour définir l'AP d'une manière générale, nous présentons l'exemple suivant [19] : supposons une requête de Google qui a 10 résultats. Le meilleur cas est que les 10 résultats soient pertinents. Si seulement certains le sont, par exemple 5, le meilleur cas serait que les 5 plus pertinents sont retournés à l'utilisateur en premier. C'est moins intéressant pour l'utilisateur que les 5 moins pertinents résultats apparaissent avant les meilleurs résultats. Le score AP reflète cette notion.

19. Une introduction au MAP, http ://fastml.com/what-you-wanted-to-know-about-mean-average-precision/

Mean Average Precision (MAP)

Le "*Mean Average Precision*"[20] [13] est une métrique utilisée dans les systèmes de recherche d'information et dans les systèmes de recommandation. Elle est utilisée pour calculer la pertinence de l'ordre d'apparition des réponses retournées et des items recommandés.

Le MAP est une moyenne des APs (Average precision). Dans notre cas, elle représente la somme des APs de plusieurs utilisateurs divisée par le nombre d'utilisateurs.

Interprétation de la précision moyenne

L'interprétation des résultats de la précision moyenne est simple, elle peut être faite de la manière suivante [97] :

Un AP = 0.1 revient à dire qu'à chaque 10 résultats on rencontre une bonne réponse. Un système qui a un AP = 0.1 est clairement mauvais.

Un AP = 0.2 revient à dire qu'à chaque 5 résultats on rencontre une bonne réponse.

Dans un système qui a un AP = 0.4 la bonne réponse est rencontrée à la 2ème ou à la 3ème position, alors qu'un système qui a un AP = 0.5 retourne plus de bonnes réponses que de mauvaises réponses.

Nous nous basons sur ces interprétations pour évaluer les résultats que nous allons obtenir.

Dans ce qui suit, nous calculons la précision moyenne "AP" des entités retournées ainsi que le "MAP" pour les trois types de requêtes (R1E, RPE, RMC) et ce pour différents utilisateurs (testeurs), en nous basant sur les résultats de la Section 6.3.1.

Précision moyenne des entités de R1E

Le tableau 6.5 suivant présente la précision moyenne des entités pertinentes de type R1E (nous considérons les réponses de 3 testeurs choisis aléatoirement).

20. Une introduction au MAP, http ://fastml.com/what-you-wanted-to-know-about-mean-average-precision/

/	Testeur 1	Testeur 2	Testeur 3	**MAP**
Précision des entités pertinentes	0.730	0.213	0.565	**0.502**

Table 6.5: Précision moyenne des entités pertinentes "R1E"

Pour chaque testeur, et pour la liste des entités retournées, nous calculons la précision pour chaque entité pertinente trouvée. Nous calculons après la MAP, la moyenne entre les APs des testeurs (voir Tableau 6.5).

La MAP est dans ce cas supérieure à 0.5 ce qui représente une bonne performance. L'interprétation des résultats de la précision moyenne a été présentée précédemment.

Précision moyenne des entités de RPE

Le tableau 6.6 suivant présente la moyenne de précision des entités pertinentes de type RPE (nous considérons les réponses de 3 testeurs choisis aléatoirement).

/	Testeur 1	Testeur 2	Testeur 3	**MAP**
Précision des entités pertinentes	0.512	0.512	0.3	**0.441**

Table 6.6: Précision moyenne des entités pertinentes "RPE"

Pour chaque testeur, et pour la liste des entités retournées, nous calculons la précision pour chaque entité pertinente trouvée. Les précisions moyennes des 3 testeurs sont présentées dans le tableau 6.5.

Nous remarquons que les APs des deux premiers testeurs sont supérieures à 0.5 alors que l'AP du dernier testeur est égale à 0.3. Le MAP dans ce cas est égal à 0.441, c'est-à-dire, une bonne réponse est rencontrée au bout de 2 à 3 réponses.

Dans ce qui suit, nous recalculons les précisions moyennes de ces mêmes testeurs en considérant les entités contextuelles comme étant des entités correctes (pertinentes). Les résultats sont présentés dans le tableau 6.7.

/	Testeur 1	Testeur 2	Testeur 3	**MAP**
Précision avec entités contextuelles	0.612	0.589	0.355	**0.518**

Table 6.7: Précision moyenne avec entités contextuelles "RPE"

Nous remarquons d'après le tableau 6.7, que si nous considérons les entités contextuelles comme étant des entités correctes, les précisions moyennes des 2 premiers testeurs dépassent largement 0.5 et la précisions moyenne des trois testeurs "MAP" est supérieure également à 0.5, ce qui représente un bon résultat.

Précision moyenne des entités de RMC

Le tableau 6.8 suivant présente la moyenne de précision des entités pertinentes de type RMC (nous considérons les réponses de 3 testeurs choisis aléatoirement).

/	Testeur 1	Testeur 2	Testeur 3	**MAP**
Précision des entités pertinentes	0.175	0.579	0.422	**0.392**

Table 6.8: Précision moyenne des entités pertinentes "RMC"

Nous remarquons d'après le tableau 6.8 que les APs des deux derniers testeurs (2 et 3) sont supérieures à 0.4 alors que l'AP du premier testeur est égale à 0.175. Le MAP est égal à 0.392, c'est-à-dire, une bonne réponse est rencontrée au bout de 3 réponses.

Dans ce qui suit, nous recalculons les précisions moyennes de ces mêmes testeurs en considérant les entités contextuelles comme étant des entités correctes aussi. Les résultats sont présentés dans le tableau 6.9.

/	Testeur 1	Testeur 2	Testeur 3	**MAP**
Précision avec entités contextuelles	0.834	0.579	0.834	**0.749**

Table 6.9: Précision moyenne avec entités contextuelles "RMC"

Nous remarquons d'après le tableau 6.9, que si nous considérons les entités contextuelles comme étant des entités correctes, les performances du système augmentent considérablement. Le MAP dépasse 0.7 et les APs individuelles dépassent 0.5. Nous remarquons que pour le testeur 2, l'AP n'a pas changé, ceci veut dire que pour ce testeur les entités sont soit pertinentes soit non pertinentes. La AP du testeur 1 a par contre augmenté considérablement, ce qui veut dire qu'il avait considéré la plupart des entités retournées comme des entités contextuelles.

6.3.3 Pertinence de la diversification des documents

Dans le but de mesurer la qualité des types et des catégories trouvées (diversification par types ou catégories), nous avons demandé aux testeurs de juger, en répondant à des formulaires, la pertinence des types trouvés pour 5 entités nécessitant la diversification par types et de juger la pertinence des catégories des documents trouvés pour 5 autres entités nécessitant la diversification par catégories.

Les réponses sont obtenues du corpus 20NewsGroups et du corpus Reuters, les utilisateurs devaient mettre "bon" si les réponses sont pertinentes, "moyen" si ce n'est pas très pertinent et "mauvais" si ce n'est pas pertinent du tout.

Les entités nécessitant la diversification sont présentées dans le tableau 6.10 et les résultats sont présentés dans la Figure 6.12 pour la diversification par types et la Figure 6.13 pour la diversification par catégories.

La Figure 6.11 suivante présente un exemple de formulaire de la diversification par catégories (avec résultats du corpus 20NewsGroups). Ce formulaire est disponible en ligne à l'adresse : https ://fr.surveymonkey.com/s/S2C3X5N

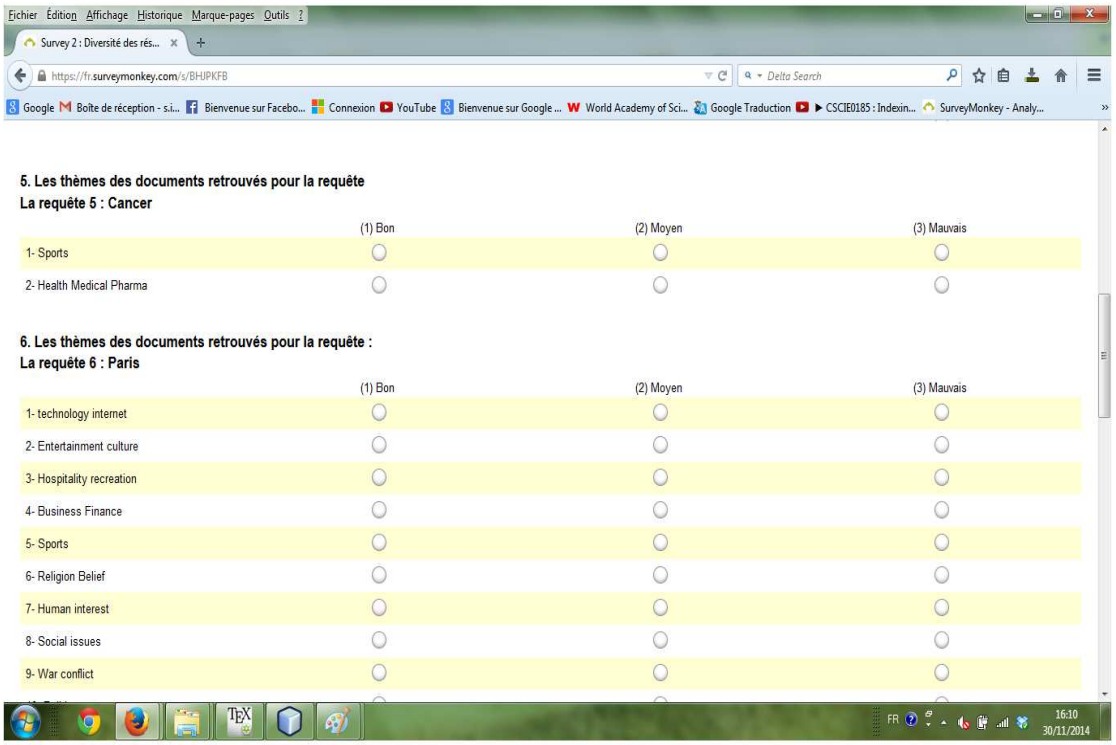

Figure 6.11: Survey 4. Exemple de formulaire de la diversification par catégories

Entités de la diversification par types	Entités de la diversification par catégories
Ford	Cancer
Washington	Paris
Lincoln	Bush
New York	Baghdad
Honda	Volvo

Table 6.10: Entités de diversification par types et catégories

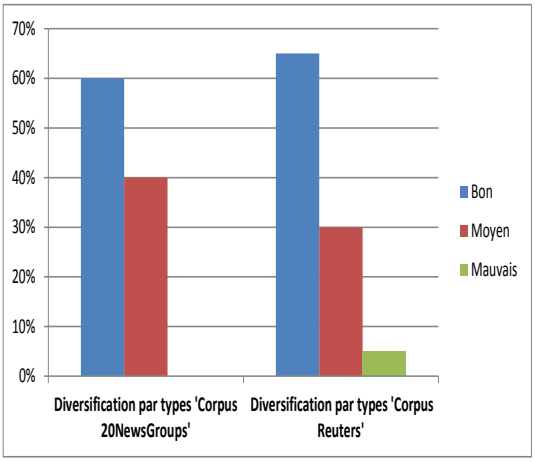

Figure 6.12: Pourcentage de pertinence de documents par rapport à la diversification de types

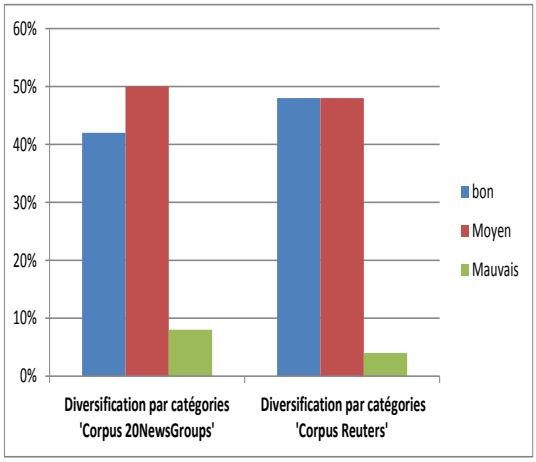

Figure 6.13: Pourcentage de pertinence de documents par rapport à la diversification de catégories

D'après la Figure 6.12 et dans la diversification par types, pour les résultats du corpus 20NewsGroups, les utilisateurs expriment que dans 60 % des cas, la diversification les a intéressé. Dans 40 % des cas, la diversification est moyenne parce qu'ils s'attendaient à un type précis (par exemple, Ford compagnie), les résultats sur Ford personne par exemple ne les intéressent pas.

Pour les résultats du corpus Reuters, les utilisateurs expriment que dans 65 % des cas, la diversification les a intéressés. Dans 30 %, la diversification est moyenne et dans 5 % elle est mauvaise. Ce cas (mauvaise diversification) coïncide avec la requête "Lincoln", certains disent qu'ils cherchaient le président Lincoln mais comme le corpus Reuters ne contient pas cette entité, les réponses parlaient uniquement de "Lincoln City" et "Lincoln voiture" ce qui n'est donc pas satisfaisant pour les utilisateurs.

Dans le cas de la diversification par catégories (Figure 6.13) et pour le corpus 20NewsGroups, dans 42 % des cas, cette diversification a été jugée bonne, 50 % moyenne et 8 % mauvaise.

Les utilisateurs ont argumenté les cas définis comme mauvais du fait que dans certains cas, ils sont intéressés par des thèmes plus que d'autres. Il est peut être alors intéressant de raffiner l'affichage en offrant à l'utilisateur une possibilité d'explorer les résultats par catégories et par types comme dans le cas des entités (celà pourra être

fait dans un travail ultérieur).

Pour le corpus Reuters, dans 48 % des cas cette diversification a été jugée bonne, 48 % moyenne et 4 % mauvaise. La richesse du corpus en types et en catégories a un impact sur la satisfaction des requêtes.

Un autre test réalisé sert à juger l'utilité de notre approche, nous avons voulu tester si les utilisateurs préfèrent notre approche à l'approche classique qui retourne une liste ordonnée de documents.

6.3.4 Utilité de notre approche

Notre but est d'analyser si les utilisateurs préfèrent notre proposition d'interpréter les requêtes en utilisant les entités des sources et d'organiser les documents de chaque entité à une approche classique qui retourne une liste ordonnée de documents en utilisant le traitement Top-k.

Nous voulons aussi tester si pour certains domaines notre approche est plus utile à l'utilisateur que l'approche classique. Nous utilisons alors le corpus 20 NewsGroups dont les données sont organisées en 20 catégories. Certaines catégories sont reliées (comp.sys.ibm.pc.hardware et comp.sys.mac.hardware) alors que d'autres catégories n'ont aucun lien. Nous avons regroupé différents newsgroups selon leurs catégories et nous avons soumis différentes requêtes (voir tableau 6.11).

Thèmes	Newsgroups	Requête	Notre approche	Approche classique
Ordinateur	comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	Disk drive	70 %	30 %
Business	misc.forsale	PC speaker	80 %	20 %
Loisirs	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	Coach of the year	60 %	40 %
Politique	talk.politics.misc talk.politics.guns talk.politics.mideast	Prime minister	70 %	30 %
Science	sci.crypt sci.electronics sci.med sci.space	Infections and tumors	60 %	40 %
Religion	talk.religion.misc alt.atheism soc.religion.christian	Catholic University	50 %	50 %

Table 6.11: Préférence des utilisateurs : Notre approche contre Approche classique

Nous avons présenté aux étudiants les résultats de notre approche et les résultats de l'approche classique qui répondent aux requêtes présentées dans le tableau 6.11. Nous leur avons demandé d'indiquer leur préférence en choisissant les meilleurs résultats entre une "Approche 1" (notre approche qui retourne un ensemble d'entités avec documents relatifs) et une "Approche 2" (l'approche classique qui retourne une liste de documents).

L'objectif est de savoir ce qui est plus utile, plus efficace et pratique pour effectuer une recherche. Nous avons ensuite calculé la moyenne pour chaque approche, c'est-à-dire, le nombre de personnes qui ont choisi l'approche divisé par le nombre total.

A partir des résultats, il est clair que les testeurs préfèrent notre approche qui est selon eux plus informative, facilite l'exploration des résultats et plus utile que l'approche classique, ce qui confirme notre motivation.

Notre approche est plus intéressante pour les utilisateurs, précisément dans les domaines où les documents sont écrits autour d'entités et portent sur un sujet particulier tel que (Politique, Ordinateur, Business). Pour les documents philosophiques ou littéraires (Religion), les entités sont moins utiles car les documents sont moins précis et plus descriptifs.

Nous avons aussi remarqué que lorsque la connaissance d'un utilisateur sur un sujet est limitée, notre approche lui apporte de la nouveauté en présentant les entités qui apparaissent dans son contexte de recherche.

6.4 Evaluation des performances

Dans le but d'étudier les performances de notre approche, nous avons calculé l'espace nécessaire aux stockage de nos index ainsi que le temps d'exécution des requêtes.

6.4.1 Espace de stockage

Il est connu qu'un index inversé peut contenir autant d'information que le corpus lui-même, son espace de stockage nécessaire peut être encore plus grand que celui requis par le corpus. Par contre, avec des techniques de compression, l'index obtenu est significativement plus petit que le corpus, mais un index occupera toujours au moins 10 % de l'espace requis pour stocker les documents eux-mêmes [98].

La Figure 6.14 suivante présente les variations de la taille des différents index utilisés en fonction de la taille du corpus.

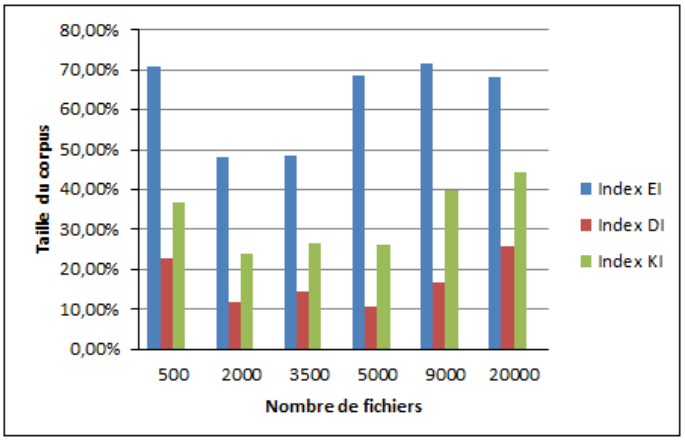

Figure 6.14: Variations de la taille des différents index en fonction de la taille du corpus

Nous remarquons d'après la Figure 6.14 qu'en variant la taille des corpus (nombre de fichiers du corpus) :

– La taille de l'index EI est de 50 % à 70 % de la taille du corpus. C'est l'index le plus gros vu qu'il a été construit pour permettre l'accès direct aux entités et qu'il contient toutes les informations sur les entités (documents, types, scores).

– La taille de l'index KI est de 25 % à 45 % de la taille du corpus. KI est l'index inversé de mots clés.

– La taille de l'index DI est de 10 % à 25 % de la taille du corpus. Cet index contient les entités des documents et leurs catégories.

Il est possible de fusionner les index et de les réorganiser autrement pour gagner en espace de stockage mais nous avons favorisé la subdivison des index, c'est-à-dire, cette organisation en trois index pour permettre un accès plus rapide aux données.

Le prix à payer pour les index inversés est l'espace supplémentaire de stockage pour la structure d'index [99]. En général, cet espace correspond à 40 % jusqu'à 200 % de la taille du corpus, selon la complexité de l'indexation. Mais ce besoin d'espace pose de moins en moins de problème maintenant [99].

6.4.2 Temps de réponse

Nous avons mesuré le temps d'exécution des requêtes. L'analyse des résultats nous permet d'affirmer que la préparation des index hors ligne permet d'effectuer des recherches efficaces et rapides. Nos index permettent également de réduire la complexité de calcul en facilitant le traitement en ligne. Les requêtes ont été exécutées 3 fois et la moyenne du temps des 3 exécutions est prise.

La Figure 6.15 présente les variations du temps d'exécution en fonction du nombre d'entités composées trouvées pour l'entité saisie (cas R1E). Différentes requêtes ont été exécutées, chacune ayant un nombre d'entités composées différent.

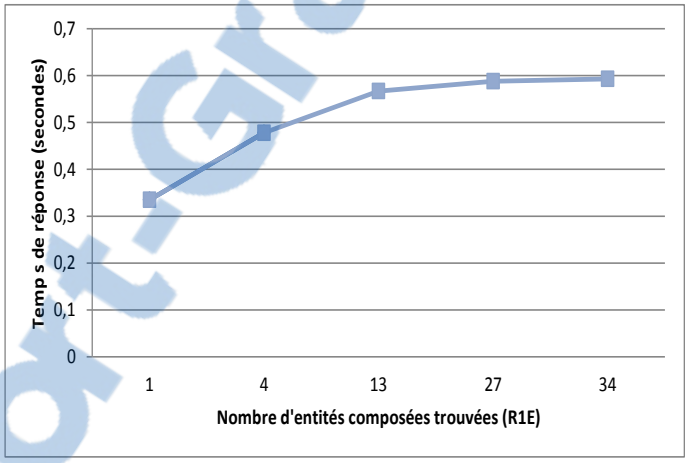

Figure 6.15: Variations du temps de réponse en fonction du nombre d'entités composées trouvées (R1E)

Nous remarquons dans la Figure 6.15 que le temps d'exécution augmente avec l'augmentation du nombre d'entités composées retournées mais l'augmentation du temps n'est pas significative.

La Figure 6.16 présente les variations du temps d'exécution en fonction du nombre d'entités relatives (pertinentes et contextuelles) trouvées (cas RPE).

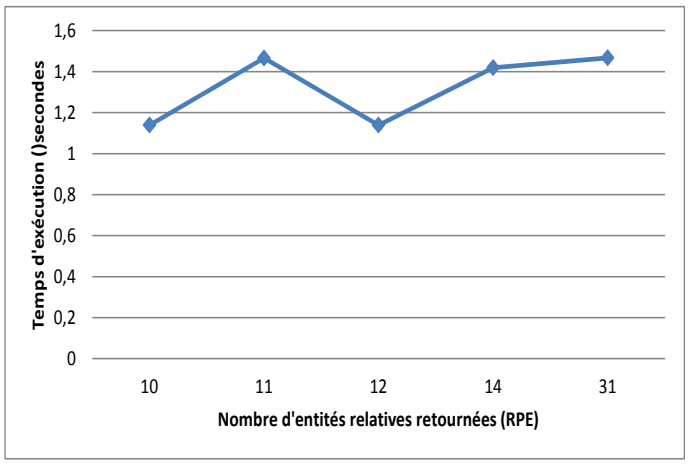

Figure 6.16: Variations du temps de réponse en fonction du nombre d'entités relatives trouvées (RPE)

La Figure 6.17 présente les variations du temps d'exécution en fonction du nombre d'entités relatives (pertinentes et contextuelles) trouvées (cas RMC).

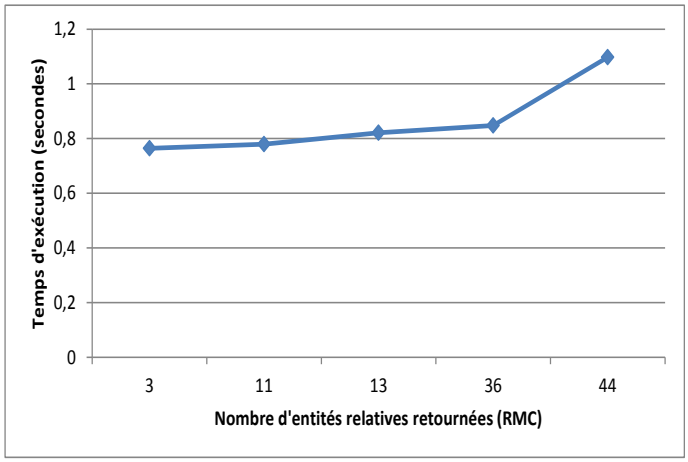

Figure 6.17: Variations du temps de réponse en fonction du nombre d'entités relatives trouvées (RMC)

Nous remarquons dans les figures 6.16, 6.17 précédentes que le temps d'exécution est plus élevé que le cas de recherche des entités composées (Figure 6.15), car la recherche des entités relatives aux requêtes RPE et RMC nécessitent d'abord un traitement : la recherche de documents communs pour RPE et la recherche des top k documents pour RMC pour ensuite retourner les entités relatives (pertinentes et contextuelles). Mais d'après les deux figures 6.16, 6.17 les variations ne sont pas significatives lorsque le nombre d'entités relatives trouvés augmente, ce qui représente une bonne performance.

6.5 Conclusion

L'objectif de ce travail est d'améliorer la pertinence des recherches d'informations en opérant une diversification des entités et des documents en exploitant les types d'entités présentes dans les documents, ainsi que les catégories des documents.

Nous proposons alors de construire un système de recherche diversifiée d'entités pour prendre en charge des requêtes construites par une ou plusieurs entités (Recherche par entités) ainsi que les requêtes de mots clés (Recherche d'entités par des mots clés).

Nous avons réalisé un prototype reflétant le fonctionnement de ce système et nous avons mené différentes expérimentations en utilisant deux corpus de données réelles.

Les corpus utilisés sont de taille raisonnable, mais nous pensons que l'indexation avec Lucene et l'utilisation de l'API scalable d'Open Calais permet de traiter les grands volumes des données, même si notre but initial a été seulement de tester notre approche de diversification.

Les expérimentations ont portées sur la pertinence des réponses aux requêtes via des notations réalisées par des testeurs (étudiants) ainsi que sur la précision des réponses retournées calculée par des métriques connues : l'AP (Average Precision) et la MAP (Mean Average Precision).

Des expériences quantitatives notamment sur les temps de réponses ont également été réalisées.

L'analyse des résultats expérimentaux nous a permis de confirmer la qualité des réponses retournées ainsi que la performance de notre approche.

Conclusion générale et perspectives

Conclusion générale

Alors que certains utilisateurs savent exactement ce qu'ils cherchent, d'autres explorent des résultats relatifs à un intérêt initial, c'est-à-dire, à une requête saisie. Quand la recherche initiale d'un utilisateur est liée aux entités, il est naturel de proposer explicitement d'autres entités relatives pour une future exploration.

D'une manière générale, lorsque l'utilisateur veut trouver une liste d'entités, par exemple des "politiciens Algériens", il est facile pour un moteur de recherche classique de retourner les documents politiques. Il est laissé à l'utilisateur d'extraire les informations sur les entités demandées à partir des résultats fournis. Notre objectif dans cette thèse est de proposer une approche qui retourne des entités pertinentes aux requêtes et les augmente par des entités relatives qui servent pour les explorations de l'utilisateur.

Notre système est motivé par l'apparition des systèmes d'annotation automatique de textes et permet à l'utilisateur de poser différents types de requêtes sur les documents annotés.

Trois types de requêtes sont considérés : recherche par une entité (R1E), recherche par plusieurs entités (RPE) et recherche par mots clés (RMC).

Notre approche s'appuie sur le fait qu'une entité peut composer d'autres entités ce qui peut intéresser l'utilisateur dans le cas de recherche par une entité (R1E). Les requêtes constituées de plusieurs entités (RPE) et celles constituées de mots clés (RMC) peuvent avoir en réponse des entités apparaissant dans leur contexte (entités contextuelles) en plus des entités pertinentes. Nous avons nommé cette approche diversification d'interprétations.

Nous proposons également de diversifier les documents relatifs aux entités retournées par type d'entité (ex. Washington interprété comme ville, personne) et par catégorie de documents (ex. Médecine, politique) pour faciliter l'interprétation de l'utilisateur. Nous avons nommé ces approches diversification par types et diversification par catégories.

Dans notre travail, nous contournons la difficulté du problème de diversification des documents et nous définissons une nouvelle notion de diversité qui nous permet d'indexer le traitement et de simplifier la complexité des requêtes.

La problématique traitée est importante et ses applications sont utiles :

1. Face au problème de la sur-spécialisation, afin de ne pas présenter des résultats trop homogènes à l'utilisateur.
2. Face au problème de la sur-personnalisation, afin de ne pas limiter les résultats aux préférences déjà connues de l'utilisateur.

Il s'agit également d'un problème difficile de recherche d'information, qui nécessite des techniques particulières pour assurer le passage à l'échelle en vu des gros volumes de données. Dans ce contexte, nous réalisons des expériences pour tester la taille des index que nous avons créé qui ont pour but de faciliter la recherche et optimiser le temps de réponse.

D'autres expérimentations sur des jeux de la littérature ont également été réalisées.

En résumé nous pouvons dire que l'objectif de ce travail était d'utiliser les annotations (types et catégories) pour réaliser une diversité des résultats dans le cadre de la recherche d'entités et de stocker les annotations dans des index pour faciliter les traitements.

L'efficacité de notre approche à retourner des résultats diversifiés aux utilisateurs a été démontrée et les expériences effectuées ont validé aussi bien le temps de réponse que la qualité des résultats obtenus.

Perspectives

Pour une continuation de ce travail, plusieurs perspectives peuvent être envisagées. Les plus immédiates sont :

– Etendre la partie annotations par d'autres systèmes automatiques tels que : Alchemy API, combiner deux systèmes en considérant les différences entre les annotations des deux systèmes et puis tester si ceci améliore les résultats.

– Améliorer la partie indexation pour traiter la vélocité de données qui est particulière à certains contextes tels que les forums.

– Proposer de nouveaux algorithmes pour le traitement du big data.

– Expliquer les entités retournées en ajoutant un descriptif (des petits résumés par exemple, des '*snippets*') et introduire la notion d'exploration par types et par catégories sur les documents relatifs aux entités pour affiner les résultats retournés pour chaque entité. Nous visons aussi à introduire la notion d'exploration par emplacement de l'entité pour catégoriser les entités composées retournées.

D'autres perspectives peuvent être réalisées ultérieurement :

– Ajouter un nouveau score au score final des résultats en considérant la fiabilité de l'extraction des systèmes d'annotations.

– Retourner les URIs des entités et introduire la notion de données liées (linked data).

– Introduire le contexte de l'utilisateur dans la recherche des résultats.

A

Annexe

Dans cette partie, nous présentons quelques notions complémentaires à notre thèse, à savoir : l'indexation par Hadoop et l'algorithme TA (Threshold Algorithm).

A.1 Indexation par MapReduce

Le modèle MapReduce permet le traitement rapide de grandes quantités de données. MapReduce peut être utilisé dans le cas où des traitements peuvent être parallélisés et appliqués sur un grand nombre de données obtenues en entrée. Les traitements sont une sorte de boucle qui peut être exécutée en parallèle sur plusieurs machines [76].

Le pseudo code donné dans [76] présente la création d'index inversé en MapReduce.

Fonction Map :

1 : class Mapper

2 : procedure Map (docid n, doc d)

3 : H <– new AssociativeArray

4 : for all term $t \in$ doc d do

5 : H$\{t\}$ <– H$\{t\}$ + 1

6 : for all term $t \in$ H do

7 : Emit (term t, posting <n,H$\{t\}$>)

Fonction Reduce :

1 : class Reducer

2 : procedure Reduce(term t, postings [<n_1, f_1>, ...])

3 : P <– new List

4 : for all posting <a, f> $\in$ postings [<n_1, f_1>, ...] do

5 : Append(P, <a, f>)

6 : Sort(P)

7 : Emit (term t, postings P)

Explication des classes Mapper, Reducer :

Pour la création de l'index, la fonction Map calculera pour tous les mots clés leurs tf (*term frequency*), voir la ligne 5 de la classe Mapper. La fonction Map émettra le terme t avec une Posting List contenant l'identificateur du document et le tf du mot clé dans ce document. Le Reduce, regroupera et ordonnera les Positing Listes de chaque terme, c'est-à-dire, tous les documents contenant le terme t avec le tf correspondant. Il émettra le terme t et sa Posting list.

A.2 Threshold Algorithm (TA)

L'algorithme TA (Threshold Algorithm) de Fagin [59] est le suivant :

Threshold Algorithm (TA) [59]

1: TA (Index Lists L_i, Query t_i, ..., t_m)
2: top-k := $\emptyset$;
3: candidates := $\emptyset$;
4: min-k := 0;
5: for all Index Lists L_i (i=1..m) do
6: // Perform next sorted access to L_i in round-robin mode
7: <d, s_i(d)> := L_i.getNext();
8: score(d) := s_i(d);
9: $high_i$:= s_i(d);
10: threshold := 0;
11: for all ν=1..m do
12: if $\nu \neq$ i then
13: // Perform random access for d's score s_ν(d) in L_ν
14: score(d) += s_ν(d);
15: end if
16: threshold += $high_\nu$;
17: end for
18: if score(d) > min-k then
19: top-k.removeMinkItem();
20: top-k.insert(d);
21: min-k := top-k.getMink();
22: else if threshold $\leq$ min-k then
23: return top-k;
24: end if
25: end for

Explication de l'algorithme [100] :

Soit une requête Query t_i, ..., t_m. TA utilise le round-robin pour ordonner les accès aux sources (les listes) L_1, ..., L_m.

L'algorithme TA utilise un modèle nommé "document à-un-temps-donné" (*document-at-a-time*), c'est-à-dire, chaque objet candidat qui est détecté lors d'un accès est recherché pour son score final dans toutes les autres listes restantes. Un score agrégé final est obtenu directement. De cette façon, aucun score d'incertitude est induit par l'algorithme.

Ensuite, un seuil initial est calculé en considérant la somme des scores $high_i$ à la position courante de chaque liste L_i. Ceci est une limite supérieure pour tous les candidats qui ne sont pas encore trouvés. Si ce seuil est inférieur au plus mauvais score "*worstscore*" parmi tous les items top-k trouvés, c'est-à-dire, au score du courant $k$ème résultat ordonné, l'algorithme peut se terminer, parce qu'aucun candidat non trouvé ne peut dépasser ce seuil.

Bibliographie

[1] T. Cheng, X. Yan, and K. C.-C. Chang. *Supporting entity search : a large-scale prototype search engine.* In *Proceedings of SIGMOD Conference*, pp. 1144–1146 (2007).

[2] T. Cheng, X. Yan, and K. C.-C. Chang. *Entityrank : Searching entities directly and holistically.* In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pp. 387–398 (2007).

[3] T. Cheng and K. C.-C. Chang. *Entity search engine : Towards agile best-effort information integration over the web.* In *Proceedings CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007*, pp. 108–113 (2007).

[4] I. Saidi, S. Amer-Yahia, and S. Nait-Bahloul. *Diversité dans la recherche d'entités.* Ingénierie des Systèmes d'Information **19**(3), 107 (2014).

[5] I. Saidi, S. Amer-Yahia, and S. Nait-Bahloul. *An approach to diversify entity search results.* In *Proceedings of the 1st International Conference on Advanced Aspects of Software Engineering, ICAASE 2014, Constantine, Algeria, November 2-4, 2014.*, pp. 44–51 (2014).

[6] I. Saidi, S. Amer-Yahia, and S. Nait-Bahloul. *Exploration diversifiée par entités*

nommées. To appear in Proceedings of the 2nd International Conference on New Technologies and Communication, ICNTC 2015, Chlef, Algeria, March 3-4, 2015.

[7] R. Jain. *Scaling lucene for indexing a billion documents.* (2013). URL `http://rahuldausa.wordpress.com/2013/01/14/scaling-lucene-for-indexing-a-billion-documents/`.

[8] M. Miller. *Scaling lucene and solr.* (2009). URL `http://searchhub.org/2009/09/02/scaling-lucene-and-solr/`.

[9] L. Qin, J. X. Yu, and L. Chang. *Diversifying top-k results.* PVLDB **5**(11), 1124 (2012).

[10] A. Angel and N. Koudas. *Efficient diversity-aware search.* In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pp. 781–792 (2011).

[11] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. *It takes variety to make a world : diversification in recommender systems.* In *Proceedings of EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009*, pp. 368–378 (2009).

[12] Y. Zhang, J. P. Callan, and T. P. Minka. *Novelty and redundancy detection in adaptive filtering.* In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pp. 81–88 (2002).

[13] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval* (Cambridge University Press, 2009).

[14] A. Z. Broder. *A taxonomy of web search.* SIGIR Forum **36**(2), 3 (2002).

[15] A. D. Mezaour. *Recherche ciblée de documents sur le Web.* Ph.D. thesis, L'université Paris Sud (2005).

[16] M. Wu and A. Marian. *A framework for corroborating answers from multiple web sources.* In *Inf. Syst.*, vol. 36, pp. 431–449 (2011).

[17] J. J. Lin and B. Katz. *Question answering from the web using knowledge annotation and knowledge mining techniques.* In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, USA*, pp. 116–123 (2003).

[18] S. T. Dumais, M. Banko, E. Brill, J. J. Lin, and A. Y. Ng. *Web question answering : is more always better ?* In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'02), Finland*, pp. 291–298 (2002).

[19] M. Zhou, T. Cheng, and K. C.-C. Chang. *Data-oriented content query system : searching for data into text on the web.* In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, NY, USA*, pp. 121–130 (2010).

[20] J. Mori. *Entity information extraction from the web using search engine : Methodology and application* (2007). URL `http://repository.dl.itc.u-tokyo.ac.jp/dspace/bitstream/2261/8143/1/mori.pdf`.

[21] L. Abouenour, K. Bouzoubaa, and P. Rosso. *Système de question/réponse dans le cadre d'une plateforme intégrée : cas de l'arabe.* In *Proceedings de la Rencontre Nationale en Informatique : Outils et applications. RNIOA'08* (Maroc, 2008).

[22] V. Moriceau, X. Tannier, and B. Grau. *Utilisation de la syntaxe pour valider les réponses à des questions par plusieurs documents.* In *Proccedings de la 6ème conférence en Recherche d'Information et Applications (CORIA 09)*, pp. 5–19 (Presqu'île de Giens, France, 2009).

[23] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. *Organizing and searching the world wide web of facts - step one : The one-million fact extraction challenge.* In *Proceedings, The 21st National Conference on Artificial Intelligence and the*

Eighteenth Innovative Applications of Artificial Intelligence Conference, USA, pp. 1400–1405 (2006).

[24] M. Cafarella and O. Etzioni. *A search engine for natural language applications.* In *Proceedings of the 14th international conference on World Wide Web, ACM New York, NY, USA*, pp. 442–452 (2005).

[25] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. *Web-scale information extraction in knowitall : (preliminary results).* In *Proceedings of the 13th international conference on World Wide Web, WWW 2004, NY, USA*, pp. 100–110 (2004).

[26] N. Wacholder, Y. Ravin, and M. Choi. *Disambiguation of proper names in text.* In *ANLP*, pp. 202–208 (1997).

[27] R. J. Gaizauskas, K. Humphreys, H. Cunningham, and Y. Wilks. *Description of the lasie system as used for muc-6.* In *Proceedings of the 6th Conference on Message Understanding, MUC'95*, pp. 207–220 (November, 1995).

[28] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. *Nyu : Description of the mene named entity system as used in muc-7.* In *In Proceedings of the Seventh Message Understanding Conference* (1998).

[29] D. Lin. *Using collocation statistics in information extraction.* In *Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[30] H. Mokeddem, F. Azouaou, and C. Desmoulins. *Ontologie de la sémantique de l'annotation pédagogique de l'apprenant.* In *Proceedings of the 2nd Conférence Internationale sur l'Informatique et ses Applications (CIIA'09), Saida, Algeria, May 3-4, 2009* (2009).

[31] W3C. *The world wide web consortium.* (2005). URL `http://www.w3.org/Amaya/User/doc/Annotations.html`.

[32] E. Desmontils, C. Jacquin, and L. Simon. *Dinosys : An annotation tool for web-based learning.* In *Proceedings of the Third International Conference on Advances in Web-Based Learning - ICWL 2004, Beijing, China, August 8-11, 2004, Proceedings*, pp. 59–66 (2004).

[33] S. B. Cousins, M. Baldonado, and A. Paepcke. *A systems view of annotations.* Tech. rep., Xerox PARC Tech Report P9910022 (2000).

[34] S. Endrullis, A. Thor, and E. Rahm. *Entity search strategies for mashup applications.* In *Proceedings of the 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pp. 66–77 (2012).

[35] M. J. Cafarella. *Extracting and querying a comprehensive web database.* In *Proceedings of CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings* (2009).

[36] S. Chakrabarti, K. Puniyani, and S. Das. *Optimizing scoring functions and indexes for proximity search in type-annotated corpora.* In *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pp. 717–726 (2006).

[37] *The road to the semantic web.* URL `http://readwrite.com/2006/11/14/semantic_web_road`.

[38] *Open calais api.* URL `http://www.opencalais.com/calaisapi/`.

[39] C. Bizer, T. Heath, and T. Berners-Lee. *Linked data - the story so far.* Int. J. Semantic Web Inf. Syst. **5**(3) (2009).

[40] *Evri api.* URL `http://www.programmableweb.com/api/evri`.

[41] *Alchemy api.* URL `http://www.alchemyapi.com/products/features/entity-extraction/`.

[42] *Zemanta api.* URL `http://developer.zemanta.com/`.

[43] *Open calais : Reuters nous rapproche du web sémantique.* URL `http://adscriptum.blogspot.com/2008/02/open-calais-reuters-nous-rapproche-du.html`.

[44] *Linked data, design issues* (2006). URL `http://www.w3.org/DesignIssues/LinkedData.html`.

[45] Y. Prié. *Sur la piste de l'indexation conceptuelle de documents : une approche par l'annotation.* Document numérique **4**(1-2), 11 (2000).

[46] X. Tannier. *Extraction et recherche d'information en langage naturel dans les documents semi-structués.* Thèse de doctorat, Université Jean Monnet, Saint-Etienne, France (2006).

[47] G. Salton, A. Wong, and C. Yang. *A vector space model for automatic indexing.* Communications of the ACM **18**(11), 613 (1975).

[48] G. Salton. *The use of extended boolean logic in information retrieval.* In *Proceedings of SIGMOD'84 Annual Meeting, Boston, Massachusetts, June 18-21, 1984*, pp. 277–285 (1984).

[49] S. E. Robertson, C. J. van Rijsbergen, and M. F. Porter. *Probabilistic models of indexing and searching.* In *SIGIR*, pp. 35–56 (1980).

[50] G. Salton and C. Buckley. *Term-weighting approaches in automatic text retrieval.* Inf. Process. Manage. **24**(5), 513 (1988).

[51] A. Mokrane. *Représentation de collections de documents textuels : application à la caractérisation thématique.* Thèse de doctorat, Université de Monpellier II, Monpellier, France (2006).

[52] A. Singhal. *Term Weighting Revisited.* Phd thesis, Université Jean Monnet, Department of Computer Science, Cornell University (1997).

[53] widad Mustafa El Hadi. *Indexation humaine et indexation automatisée : la place du terme et des environnements*. In *Proceedings of the Third International Conference of 7es Journées scientifiques AUF - LTT "Mots, termes et contextes"*, pp. 157–167 (2005).

[54] J. Pound, P. Mika, and H. Zaragoza. *Ad-hoc object retrieval in the web of data*. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pp. 771–780 (2010).

[55] *Wisdm : Web indexing and search for dynamic mining*. URL `http://wisdm.cs.uiuc.edu`.

[56] G. Demartini. *From people to entities : typed search in the enterprise and the web*. Thèse de doctorat, University Leibniz of Hanover, Hannover, Germany (2011).

[57] Q. Li and D. He. *Searching for entities when retrieval meets extraction*. In *Proceedings of The Nineteenth Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA, November 16-19, 2010* (2010).

[58] K. C. Chang and S. Hwang. *Minimal probing : supporting expensive predicates for top-k queries*. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*, pp. 346–357 (2002).

[59] R. Fagin, A. Lotem, and M. Naor. *Optimal aggregation algorithms for middleware*. J. Comput. Syst. Sci. **66**(4), 614 (2003).

[60] R. Fagin. *Combining fuzzy information from multiple systems*. J. Comput. Syst. Sci. **58**(1), 83 (1999).

[61] R. Akbarinia, E. Pacitti, and P. Valduriez. *Best position algorithms for top-k queries*. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pp. 495–506 (2007).

[62] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. *Diversifying search results.* In *Proceedings of the Second ACM International Conference on Web Search and Data Mining, WSDM '09. New York, NY, USA, 2009. ACM.*, pp. 5–14 (2009).

[63] M. Ehrmann. *Les entités nommées, de la linguistique au TAL : Statut théorique et méthodes de désambiguation.* Thèse de doctorat, Université Paris 7 - Denis Diderot, Paris, France (2008).

[64] D. Nouvel. *Reconnaissance des entités nommées par exploration de règles d'annotation.* Thèse de doctorat, Université François Rabelais de Tours, Tours, France (2012).

[65] P. Bailey, A. P. de Vries, N. Craswell, and I. Soboroff. *Overview of the trec 2007 enterprise track.* In *Proceedings of The Sixteenth Text REtrieval Conference, TREC 2007, Gaithersburg, Maryland, USA, November 5-9, 2007* (2007).

[66] A. McLean, A. M. Vercoustre, and M. Wu. *Enterprise peoplefinder : Combining evidence fromweb pages and corporate data.* In *Proceedings of Australian Document Computing Symposium* (2003).

[67] S. Abney, M. Collins, and A. Singhal. *Answer extraction.* In *Proceedings of the sixth conference on Applied natural language processing. Morristown, NJ, USA*, pp. 296–301 (2000).

[68] E. Meij, K. Balog, and D. Odijk. *Entity linking and retrieval for semantic search.* In *Proceedings of the Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, pp. 683–684 (2014).

[69] T. Cheng and K. C.-C. Chang. *Beyond pages : Supporting efficient, scalable entity search with dual-inversion index* (2010).

[70] K. C. Chang, B. He, and Z. Zhang. *Toward large scale integration : Building a metaquerier over databases on the web.* In *Proceedings of CIDR*, pp. 44–55 (2005).

[71] M. Bautin and S. Skiena. *Concordance-based entity-oriented search.* Web Intelligence and Agent Systems **7**(4), 303 (2009).

[72] M. Käki. *Findex : search result categories help users when document ranking fails.* In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 131–140 (2005).

[73] P. Fafalios, I. Kitsos, Y. Marketakis, C. Baldassarre, M. Salampasis, and Y. Tzitzikas. *Web searching with entity mining at query time.* In *Proceedings of the 5th International Retrieval Facility Conference - Multidisciplinary Information Retrieval, IRFC 2012, Vienna, Austria, July 2-3, 2012*, pp. 73–88 (2012).

[74] I. Kitsos, K. Magoutis, and Y. Tzitzikas. *Scalable entity-based summarization of web search results using mapreduce.* In *Distributed and Parallel Databases (DAPD), Springer Journals* (2013).

[75] J. Dean and S. Ghemawat. *Mapreduce : simplified data processing on large clusters.* vol. 51, pp. 107–113 (2008).

[76] J. Lin and C. Dyer. *Data-Intensive Text Processing with MapReduce* (Morgan & Claypool Publishers, 2010).

[77] A. Singhal. *Introducing the knowledge graph : things, not strings* (2012). URL `http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html`.

[78] S. M. McNee, J. Riedl, and J. A. Konstan. *Being accurate is not enough : how accuracy metrics have hurt recommender systems.* In *Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pp. 1097–1101 (2006).

[79] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. *Evaluating collaborative filtering recommender systems.* ACM Transactions on Information Systems **22**(1), 5 (2004).

[80] G. Adomavicius and A. Tuzhilin. *Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions.* IEEE Trans. Knowl. Data Eng. **17**(6), 734 (2005).

[81] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. *Recommendation diversification using explanations.* In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pp. 1299–1302 (2009).

[82] P. Pu and L. Chen. *Trust building with explanation interfaces.* In *IUI* (2006).

[83] Z. Chen and T. Li. *Addressing diverse user preferences : A framework for query results navigation.* vol. 32, pp. 41–48 (2009).

[84] E. Vee, J. Shanmugasundaram, and S. Amer-Yahia. *Efficient computation of diverse query results.* vol. 32, pp. 57–64 (2009).

[85] D. Xin, H. Cheng, X. Yan, and J. Han. *Extracting redundancy-aware top-k patterns.* In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pp. 444–453 (2006).

[86] J. Carbonell and J. Goldstein. *The use of mmr, diversity-based reranking for reordering documents and producing summaries.* In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA*, SIGIR '98, pp. 335–336 (ACM, 1998).

[87] F. Radlinski and S. T. Dumais. *Improving personalized web search using result diversification.* In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pp. 691–692 (2006).

[88] K. Liu, E. Terzi, and T. Grandison. *Highlighting diverse concepts in documents.* In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pp. 545–556 (2009).

[89] M. Drosou and E. Pitoura. *Search result diversification.* SIGMOD Record **39**(1), 41 (2010).

[90] C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, and et al. *Novelty and diversity in information retrieval evaluation.* In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pp. 659–666 (2008).

[91] M. Géry. *Indexation et interrogationde chemins de lecture en contextepour la Recherche d'Information Structuréesur le Web.* Thèse de doctorat, Université Joseph Fourier - Grenoble I, Grenoble, France (2002).

[92] *Lucene, librairie open source.* URL `http://www.doculibre.com/apache-lucene-librairie-java-open-source`.

[93] *When to hadoop, and when not to.* URL `http://www.datanami.com/2014/01/27/when_to_hadoop_and_when_not_to/`.

[94] R. Steinberger, B. Pouliquen, M. A. Kabadjov, and E. V. der Goot. *Jrc-names : A freely available, highly multilingual named entity resource.* CoRR **abs/1309.6162** (2013).

[95] G. Salton. *The performance of interactive information retrieval.* Information Processing Letters (2), 35 (1971).

[96] *average precision.* URL `https://sanchom.wordpress.com/tag/average-precision/`.

[97] *Intuition behind average precision and map.* URL `http://makarandtapaswi.wordpress.com/2012/07/02/intuition-behind-average-precision-and-map/`.

[98] J. Zobel and A. Moffat. *Inverted files for text search engines.* ACM Comput. Surv. **38**(2) (2006).

[99] J. Y. Nie. *Recherche d'information* (2012). URL `http://www.iro.umontreal.ca/~nie/IFT6255/Introduction.pdf`.

[100] M. Theobald. *TopX, Efficient and Versatile Top-k Query Processing for Text, Structured, and Semistructured Data*. Phd thesis, Max-Planck-Institut für Informatik, Germany (2006).