UNIVERSITÉ DU QUÉBEC À MONTRÉAL

RECHERCHE D'INFORMATION TRANSLINGUISTIQUE BASÉE SUR UNE TRADUCTION DE REQUÊTES DIRECTE ET TRANSITIVE UTILISANT L'ENCYCLOPÉDIE EN LIGNE WIKIPÉDIA

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

HABIBA CHAKOUR

MARS 2015

REMERCIEMENTS

Je tiens d'abord à remercier *Dieu* le tout puissant et miséricordieux, qui m'a donné la force et la patience pour accomplir ce travail de recherche.

Mes vifs remerciements à ma directrice de recherche Mme *Fatiha Sadat*, pour ses conseils, son soutien précieux, et son aide durant toute la période de travail.

Mes sincères remerciements à tous les professeures et professeurs qui m'ont enseignés durant mes années d'études.

Je dédie ce mémoire à mes parents, mes beaux-parents, mes frères et sœurs, mes beaux-frères et belles-sœurs pour leur patience, soutien moral et prière.

J'ai envie de remercier vivement mon époux, *Boubaker Bernouk,* qui m'a soutenu et encouragé du début à la fin de ce mémoire. Je le remercie fortement pour son aide précieuse tout le long de la période de mes études.

Enfin, je remercie chaleureusement ma nièce *Aicha* pour son aide dans la réalisation de ce mémoire, et mes amies fidèles pour leur prière et soutien moral *Fatma Zohra*, *Nadjette*, *Karima* et *Nabila Nouaouria*.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

L'encyclopédie multilingue *Wikipédia* est devenue une ressource très utile pour la traduction des requêtes et la construction des ressources linguistiques, comme les dictionnaires et les ontologies.

Dans cette étude, nous nous sommes intéressées à l'exploitation de *Wikipédia* pour la traduction des requêtes pour la recherche d'information translingue, et ce, pour la paire de langues arabe-anglais. Toutes les traductions candidates possibles sont extraites à partir des titres des articles *Wikipédia*, en s'appuyant sur les liens inter-langues arabe-anglais pour la traduction directe, ou en utilisant le français comme langue pivot pour fournir une traduction transitive. Un prétraitement et une segmentation de la requête en plusieurs unités lexicales peuvent être effectués, si aucune traduction ne peut être trouvée pour la requête entière. Une traduction transitive peut être également opérée, si aucun résultat n'est retourné suite à la traduction directe.

Pour la segmentation de la requête, nous avons proposé une nouvelle méthode qui tient compte de la complexité morphologique de la langue arabe. De plus, elle peut être appliquée à d'autres langues sans aucune modification, comme par exemple l'anglais, le français, l'espagnol et l'allemand. Pour le chinois une légère adaptation est nécessaire, car il n'y a pas d'utilisation de l'espace en tant que séparateur entre deux mots ou groupes de mots.

Afin de réduire la complexité de notre application, notre méthode de segmentation est basée sur l'application de deux formules destinées respectivement aux requêtes courtes et longues. Pour cela, deux seuils sont utilisés et peuvent être modifiés par l'utilisateur en fonction des performances de sa machine.

Par ailleurs, nous avons utilisé la boite à outils externe MADA+TOKAN pour une analyse morphologique des requêtes arabes et nous avons étudié l'effet d'un tel prétraitement sur les performances de notre méthode de traduction.

Les évaluations du système monolingue et translingue ont été effectuées selon quatre variantes, sans prétraitement, avec prétraitement, analyse morphologique avec MADA+TOKAN, et traduction transitive combinée à la traduction directe. La meilleure précision a été obtenue en appliquant le prétraitement.

Néanmoins, comparativement à la variante sans prétraitement, le résultat de MADA+TOKAN est meilleur. En outre, la performance de la dite méthode de traduction a été comparée avec celles de Google Translate et Mymemory.

MOTS CLÉS : Traduction de requêtes, Recherche d'information translingue, Wikipédia.

ABSTRACT

The multilingual encyclopaedia *Wikipedia* has become a very useful resource for query translation and construction of language resources such as dictionaries and ontologies.

In this study, we are interested by the exploitation of *Wikipedia* for query translation in Cross-Language Information Retrieval for the Arabic-English pair of languages. All possible translation candidates are extracted from the titles of *Wikipedia* articles based on the inter-links between Arabic and English for direct translation, or using French as a pivot language to provide a transitive translation. Preprocessing and segmentation of the query into multiple tokens can be done if no translation can be found for the entire query. A transitive translation can also be made, if no results are returned after the direct translation.

For query segmentation, we proposed a new method that considers the morphological complexity of the Arabic language. In addition, it can be applied to other languages without modification, such as English, French, Spanish and German. For Chinese a slight adaptation is necessary because there is no use of space as a separator between two words or group of words.

To reduce the complexity of our application, our segmentation method is based on the application of two formulas destined respectively to short and long queries. For this, two thresholds are used and can be modified by the user according to the performance of his machine.

Furthermore, we used the external toolbox MADA+TOKAN for morphological analysis of arabic queries, and thus study the effect of such pretreatment on the performance of our translation method.

Assessments for monolingual and crosslingual systems were performed in four variants, without pretreatment, with pretreatment, MADA+TOKAN morphological analysis, and transitive translation combined with the direct translation. The best accuracy was achieved by applying the pretreatment. However, compared to the variant without pretreatment the result of MADA+TOKAN is better. Furthermore, the performance of this translation method was compared with those of Google Translate and MyMemory.

KEYWORDS: Query Translation, Cross-Lingual Information Retrieval, Wikipédia.

INTRODUCTION

La recherche d'information translingue (RIT), est devenue un champ de recherche important dans le domaine de traitement automatique des langues naturelles (TALN), dont l'objectif est de traiter des données linguistiques exprimées dans une langue naturelle [31]. Ce domaine a émergé grâce à l'explosion de la quantité d'informations électroniques accessibles via Internet dans une grande varieté de langues [1].

Traditionnelement, la Recherche d'information (RI) est basée sur l'extraction des documents pertinents, en faisant correspondre les mots d'une requête à ceux d'un document. La requête est le plus souvent une séquence de mots qui exprime le besoin d'informations d'un utilisteur. Dans la RIT, les documents sont écrits dans une langue différente de la langue de la requête, et pour les mettre en correspondance, une traduction est donc nécessaire (traduction de la requête vers la langue des documents ou l'inverse) [1][2][3][4][5].

La traduction de la requête (généralement réduite à quelques termes) d'une langue source vers une langue cible (la langue des documents) est un processus commun à la plupart des systèmes RITs, au lieu de la traduction des documents dans la langue de la requête qui est une tâche plus coûteuse [1][5].

Dans la littérature, il existe plusieurs méthodes de traduction basées sur :

- La traduction automatique : Appelée aussi traduction logicièlle, elle consiste à traduire entièrement un texte sans l'intervention de l'être humain, parmi les

systèmes de traduction automatique les plus connus, on peut citer Google Translate[1], Systran[2], et Babelfish[3] [30].

- Les corpus parallèles : Sont un ensemble de documents bilingues alignés par phrase, c'est-à-dire qu'ils sont la traduction idéale l'un de l'autre, comme le Hansard canadien, un corpus parallèle aligné qui contient la transcription officielle des débats parlementaires souvent utilisés dans des recherches en traduction automatique [32].
- Les corpus comparables : Un ensemble de documents bilingues non alignés par phrase, et qui peuvent porter sur le même thème, comme les articles de *Wikipédia* ayant des liens inter-langues et les articles de journaux publiés pendant une période donnée [32].
- Les dictionnaires electroniques (*MRD : Machine Readable Dictionary)* : Se sont des dictionnaires bilingues sous une forme electronique, il s'agit des bases de données lexicales qui peuvent être interrogées par une application logicièlle.
- Les vocabulaires contrôlés (*Thésaurus*) : Un Thésaurus est un ensemble de termes et d'associations ou de relations entre ces termes. WordNet, est un exemple notable construit par un grand nombre de contributeurs, et contient plus de 80 000 noms et 90 milliers de relations hiérarchiques [33].
- *Wikipédia*[4] : Est une encyclopédie libre, en rédaction continue par un grand nombre de contributeurs.
- Les approches combinées : Peuvent être toute combinaison des approches précédentes.

---

[1] https://translate.google.ca/?hl=fr

[2] http://www.systran.fr/lp/traduction-en-ligne/

[3] http://www.babelfish.com/

[4] http://fr.wikipedia.org/wiki/Wikipédia

Néanmoins, la traduction automatique n'est pas idéale, sauf si les phrases de la requête sont grammaticalement correctes. De même pour les corpus parallèles ou comparables sont rarement disponibles dans les sujets de toutes les requêtes [1][5][6]. Les dictionnaires électroniques par contre, qui peuvent être utilisés dans les systèmes RITs sont faciles à trouver. Cependant, il n'est pas toujours facile de trouver les MRDs appropriées entre les langues, même entre les langues européennes communes [1]. Si on ne peut traduire directement d'une langue A vers une langue B, on peut faire une traduction transitive, c'est-à-dire trouver un dictionnaire entre la langue A et C et entre la langue C et B [1][2][3][4].

Dans ce contexte, plusieurs problèmes se posent. Pour les approches à base de dictionnaires, la question fondamentale est l'ambiguïté due aux variations morphologiques (des mots différents sont utilisés pour représenter le même sens), et sémantiques (des mots ont plusieurs sens). Un mot peut donc avoir plusieurs traductions qui ne sont pas toutes pertinentes [1][2][3][4] [7][8].

Les mots introuvables dans les sources de traduction (*OOV : Out Of Vocabulary*), sont aussi une source d'erreur commune aux systèmes RITs. La plupart des OOVs sont souvent des entités nommées (des noms propres, des noms d'organisations, des noms de lieux, etc) qui peuvent être rendus dans l'orthographe de la langue cible, ce processus est appelé translittération [9][10].

Une proportion significative des mots OOVs peuvent être les mots les plus importants dans la requête. Larkey et al. (2003) [11], ont montré que la performance de la RIT en termes de la précision moyenne (voir section 1.4), est réduite à plus de 50% quand les entités nommées dans les requêtes ne sont pas traduites [11]. En outre, les termes issus de la traduction peuvent être aussi non suffisants pour exprimer la requête initiale (problème d'expansion de requêtes) [8].

Les approches basées sur *Wikipédia* pour la traduction des requêtes, proposent des solutions aux dits problèmes et présentent les avantages suivants par rapport aux ressources existantes utilisées (comme les dictionnaires bilingues, les corpus parallèles, etc) [12]:

- Une meilleure couverture des entités nommées et des termes spécifiques au domaine, ce qui pourrait la rendre appropriée pour la gestion des traductions des noms propres.
- Les articles de Wikipédia fournissent plus de contexte par rapport aux dictionnaires en ligne (utile pour la désambiguïsation sémantique des mots), car c'est une encyclopédie multilingue à laquelle sont attachés plusieurs contributeurs, qui la tiennent constamment à jour dans une très grande variété de domaines.
- La présence des pages de redirection représentant des noms de concepts (par exemple, des synonymes, des abréviations et des variantes orthographiques), et qui se composent d'un lien qui dirige vers l'article principal qu'elle représente (elles peuvent être utilisées pour l'expansion de requêtes).

Néanmoins, la couverture des mots communs dans *Wikipédia* est plus faible que les dictionnaires d'usage général (dictionnaires bilingues), et certains termes ont plusieurs sens, certains très spécifiques et rares, rendent la désambiguïsation lexicale plus difficile. Par exemple, dans *Wikipédia* le terme «*house*» a des sens comme «*novel*», «*song*», «*operating system*» ou «*game*» [12].

Dans le cadre de notre travail de recherche, nous présentons une approche de traduction de requêtes pour la RIT arabe-anglais en exploitant l'encyclopédie multilingue *Wikipédia*. Ce travail repose sur une proposition de Gaillard et al. (2010) [22], appliquée sur une langue source ayant une morphologie très riche, qui est l'*arabe*. Aussi, la traduction de la requête est effectuée en s'appuyant sur les liens inter-langues

des articles *Wikipédia* (traduction directe ou transitive). Pour la traduction transitive le français a été choisi comme langue pivot.

Le contenu de ce mémoire est organisé en plusieurs chapitres. Le chapitre I et le chapitre II sont consacrés respectivement à la recherche d'information et la recherche d'information translingue. Les caractéristiques morphologiques de la langue arabe sont présentées dans le chapitre III. L'état de l'art et la problématique examinée sont décrits respectivement dans les chapitres IV et V. La méthodologie proposée est détaillée dans le chapitre VI. Les évaluations et les résultats sont présentés dans le chapitre VII. Enfin, une conclusion suit.

CHAPITRE I

LA RECHERCHE D'INFORMATION

La recherche d'information (RI), est un domaine dont l'objectif est de trouver du matériel (généralement des documents) de nature non structurée (généralement du texte), qui satisfait un besoin d'information au sein de grandes collections (généralement stockées sur support informatique) [34].

Un système RI permet de rechercher des éléments pertinents (une liste de documents), en réponse au besoin d'informations de l'utilisateur (la requête). La pertinence, est une notion particulière qui peut être influencée par des facteurs non explicitement disponibles dans le document ou la requête (les préférences de l'utilisateur, les connaissances antérieures, etc) [13].

La requête de l'utilisateur peut sous-spécifier (manquer de termes essentiels), ou sur-spécifier le besoin d'informations désiré (contenir des termes incorrects). Par conséquent, il n'est pas toujours garanti que les documents pertinents contiennent nécessairement la totalité ou même l'un des termes de recherche fourni [13].

La tâche d'un système RI monolingue qui reçoit une requête en entrée, et fournit une liste de documents classés en sortie, est séparée en deux principales phases : la phase d'indexation, et la phase d'appariement (Figure 1.1) [13].

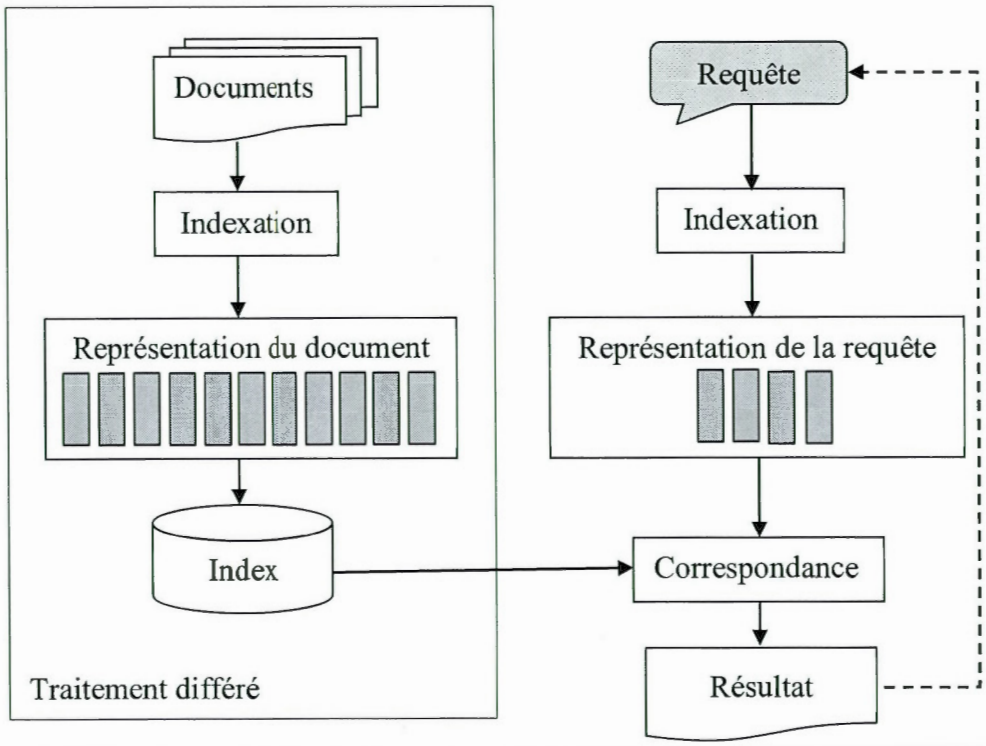


Figure 1.1 Architecture conceptuelle de base d'un système de recherche d'information. Source: Peters et al. (2012) [13].

1.1 La phase d'indexation

L'indexation construit des représentations appropriées des documents et de la requête en plusieurs étapes successives. L'indexation des documents est effectuée avant le traitement de la requête. Parmi les étapes d'indexation les plus courantes on peut citer [13]:

- Prétraitement : En raison de l'utilisation de différents systèmes de codage et plus particulièrement ceux spécifiques à une langue donnée, tous les documents sont convertis à un codage de caractères commun (*Unicode*), qui englobe toutes les langues d'intérêt et leurs propriétés. Un prétraitement additionnel est possible,

telle que, la suppression des sections ne portant pas sur le contenu (en-têtes, pieds de page, des zones de navigation, les messages de copyright, etc).

- Identification de la langue : S'il n'y a aucune indication claire de la langue d'un document ou d'une requête, un composant d'identification de la langue doit être utilisé. Les principales méthodes d'identification sont basées sur :

  - l'utilisation des mots vides (*Stopwords*) qui sont les mots non significatifs dans un texte, et sont inutiles à indexer ou à utiliser dans la recherche, comme «*le, la, de, ce*...etc.» pour le français;

  - la fréquence de caractères *n-gramme*;

- Formation de documents : Les documents stockés dans les systèmes RI peuvent considérablement varier en taille. Afin de répondre à l'efficacité d'extraction et à améliorer les résultats pour les utilisateurs, une granularité plus ou moins fine que le document entier est choisi. Pour cela, il existe plusieurs scénarios où les systèmes opèrent au niveau du :

  - Sous-document : les passages de longueur fixe (les documents sont divisés en morceaux contenant un nombre fixe de tokens), la détection de la frontière du paragraphe ou la détection de la limite de la phrase. Les tokens sont les mots ou les unités lexicales produites à la suite de l'analyse lexicale du document (tokenisation).

  - Super-document : assemblage de texte à partir de plusieurs documents hyperliés, ou enrichissement du texte d'un document en incluant des métadonnées.

- Segmentation, tokenisation, analyse : Pour permettre leur correspondance à la requête, les documents doivent être segmentés en unités plus courtes. Pour la plupart des langues européennes, le choix évident est la segmentation en mots.

Si un système est axé sur la recherche de textes, le mot «termes» (terms) est souvent utilisé pour décrire les unités, tandis que le mot «traits» (features) est un bon choix si le contenu non textuel est également couvert dans le discours. Les deux mots «termes» et «traits» se produisent sous forme de flux de tokens, une séquence des unités en sortie à l'aide d'un composant de segmentation ou de tokenisation.

Pour un système robuste, et en particulier en ce qui concerne les entités nommées, les termes «*O'Brien*», «*F/A-18*», «*Coca-Cola*», «*Yahoo! Mail*» sont divisés en plusieurs tokens. Par conséquent, pour éviter une telle division, un dictionnaire des entités nommées ou un traitement linguistique plus sophistiqué est nécessaire. En outre, afin de réduire le nombre de tokens à indexer, l'élimination des mots vides (les déterminants, les conjonctions, les prépositions, les interjections et autres) est également bénéfique pour une recherche plus précise.

- Normalisation des traits : Dans l'espoir d'améliorer le potentiel d'appariement entre la requête et les documents, les candidats valables identifiés pour les traits de recherche peuvent être normalisés. Les principales raisons du mésappariement sont, l'utilisation des synonymes, des formes de surface alternatives du même lexème, et différentes applications de conventions d'écriture (signes diacritiques, capitalisation, orthographe, etc).

- Racinisation (*stemming*) : Est une tentative pour minimiser le mésappariement entre les requêtes et les documents, due à l'utilisation des différentes formes de mots. Un composant de stemming va supprimer les suffixes ou les préfixes communs des mots, en se rapprochant ainsi du lemme (de la racine ou de la forme du dictionnaire) de ce mot. Par exemple, l'algorithme de Porter pour l'anglais (Tableau 1.1). La première règle de ce tableau (*SSES* ➔ *SS*) supprime le suffixe «*ES*» pour le mot «*caresses*» par exemple, le résultat fourni par Porter est

«*caress*». La mesure *m* compte le nombre de séquences de voyelle-consonne dans un mot.

Comme on peut le voir d'après les exemples donnés pour les étapes 4 et 5, les règles peuvent produire des chaînes qui ne constituent pas des mots anglais valides.

Pour un grand nombre de langues, le stemming a été montré efficace, et en particulier pour les langues à morphologie complexe, à la fois flexionnelles et dérivationnelles. Pour la langue arabe par exemple, les stems peuvent être réduits à leurs racines (*roots*). Bien qu'un stemmer arabe puisse généralement supprimer les préfixes et les suffixes d'un mot, les infixes sont également supprimés pour obtenir la racine. Par exemple, pour le mot arabe «الكتابة» («*AlktAbp*»[5], «*L'écriture*»)[6], sa racine «كتب» («*ktb*», «*É*crire») est obtenue en supprimant le préfixe «ال : *Al*», le suffixe « ة: *p*» et l'infixe « ا : *A*» [13].

Tableau 1.1 Exemple de règles à partir de différentes étapes du stemmer Porter (cité de Porter, 1980). Source: (Peters et al. 2012) [13].

Exemple de règles	Exemple du résultat des règles
Étape 1: SSES → SS	caresses → caress
Étape 2: (m > 0) ATOR → ATE	operator → operate
Étape 3: (m > 0) NESS →	goodness → good
Étape 4: (m > 1) IBLE →	defensible → defens
Étape 5: (m > 1) E →	probate → probat

[5] La translittération arabe de *Buckwalter* : {(A : ا), (b : ب), (t : ت), (v : ث), (j : ج), (H : ح), (x : خ), (d : د), (* : ذ), (r : ر),(z : ز),(s : س), ($: ش), (S : ص), (D : ض), (T : ط), (Z : ظ), (E : ع), (g : غ), (f : ف), (q : ق), (k : ك), (l : ل), (m : م),(n : ن), (h : ه), (w : و),(y : ى/ي), (p : ة), (| : آ), (< : إ), (> : أ), (} : ئ), (' : ء) , (~: ّ), (a : َ) ,(u : ُ) ,(i : ِ) ,(o : ْ), (F : ً), (N : ٌ), (K : ٍ)}

[6] La translittération arabe de *Buckwalter* du mot arabe «الكتابة» et sa traduction en français.

Par ailleurs, la décomposition est une question étroitement liée au stemming. Un certain nombre de langues, comme les langues germaniques (allemand, suédois, néerlandais, finlandais) et coréen entre autres, font un large usage d'un mécanisme de formation de mots où de nouveaux mots composés sont formés de plusieurs mots (de base), et où le mot composé est écrit sans espace séparant ses constituants [13].

Les techniques *n-grams* sont une alternative utile au stemming, elles sont indépendantes de la langue. Pour la recherche des caractères *n-grammes*, des mots sont divisés en sous-unités, c'est à dire, un ensemble de chaînes de caractères qui se chevauchent, dont la longueur varie généralement entre quatre et six caractères. Par exemple, le mot «*airport*», peut être divisé en caractères *n-grammes* de longueur 4 comme suit : « *_air*», «*airp* », «*irpo* », «*rpor*», «*port*», «*ort_* ». Cette technique donne généralement un certain nombre de caractères communs *n-grammes* pour des paires de mots qui devraient être confondues, et ces *n-grammes* ont tendance à donner une représentation adéquate de la racine du mot [13].

Dans le but d'améliorer les performances des systèmes de recherche d'information, des étapes de traitement additionnelles peuvent être menées, telles que, la détection des expressions poly-lexicales (*MWE* : *MultiWord Expression*), un groupe de mots ayant un comportement idiosyncrasique qui peuvent être ajoutées à l'index [13].

Par ailleurs, plusieurs techniques d'expansion existent dans la littérature (*PRF : Pseudo Relevance Feedback*, *Thesauri,* etc), elles consistent à ajouter des termes au texte source afin d'améliorer son expressivité [13].

1.2 La phase d'appariement

Cette phase, consiste à effectuer la correspondance des représentations d'index respectives de la requête et des documents. Voici quelques approches [13]:

• Paradigme sac de mots (Bag of Words) : Les approches d'appariement dominantes traitent les flux résultants des tokens comme un sac (bag) non trié (*multiset*), c'est à dire, qu'elles ignorent l'ordre dans lequel les mots se sont produits dans le texte. Le Tableau 1.2, montre un exemple de représentation d'un document et d'une requête sous forme de sac de mots. Soit «*index structures in IR systems*» une requête de l'utilisateur, on peut remarquer qu'elle ne partage aucun mot commun avec le texte du document donné (le premier bloc du Tableau 1.2), sauf le mot vide «*in*». Le document et la requête ont été indexés selon les étapes d'indexation décrites précédemment.

Évidemment, après indexation, il y a un meilleur appariement des mots «*index*» et «*system*». Certains points plus fins et limitations du stemming deviennent également évidents : La confusion sur «*structures*» est probablement un indésirable, alors que dans le texte on parle de «*Structuring the discussion*», et l'utilisateur est à la recherche de «*index structures*». En outre, aucune confusion entre «*Information Retrieval*» et «*IR*» n'a lieu, cette forme de correspondance nécessite des ressources supplémentaires comme un thésaurus spécifique à un domaine.

• Index inverse : La tâche principale du composant d'appariement, est de calculer la similarité entre le sac représentant la requête et les sacs pour chaque document contenu dans le système, en d'autres termes, le système conceptuellement doit calculer un très grand nombre de scores de similarité (par exemple, de l'ordre de milliards dans le cas des services de recherche Web). La formule de calcul des scores, est appelée schéma de pondération (weighting scheme). Afin de calculer efficacement les scores, deux simplifications de base sont utilisées par presque tous les systèmes de recherche:

- les documents qui ne contiennent aucun des traits contenus dans la représentation indexée de la requête sont attribués un score zéro (0).

-seulement, les correspondances exactes entre les traits de la représentation indexée de la requête et la représentation indexée du document sont considérées.

Tableau 1.2 Version originale et représentation sac de mots pour un échantillon de texte et une requête possible associée respectivement[7].

Source: (Peters et al. 2012) [13].

«The information retrieval system stands at the core of many information acquisition cycles. Its task is the retrieval of relevant information from document collections in response to a coded query based on an information need. In its general form, when searching unstructured, natural language text produced by a large range of authors, this is a difficult task: in such text there are many different valid ways to convey the same information. Adding to the complexity of the task is an often incomplete understanding of the desired information by the user. In this chapter, we discuss the mechanisms employed for matching queries and (textual) documents within one language, covering some of the peculiarities of a number of widely spoken languages. Effective within-language retrieval is an essential prerequisite for effective multilingual information access. The discussion of within-language information retrieval or monolingual information retrieval can be structured into two main phases: the indexing phase, commonly implemented as a pipeline of indexing steps, producing a representation that is suitable for matching; and the matching phase, which operates on the indexed representations and produces a ranked list of documents that are most likely to satisfy the user's underlying information need.»
access(1) acquisit(1) ad(1) author(1) base(1) chapter(1) code(1) collect(1) commonli(1) complex(1) convei(1) core(1) cover(1) cycl(1) desir(1) difficult(1) discuss(2) document(3) effect(2) emploi(1) essenti(1) form(1) gener(1) implement(1) incomplet(1) index(3) inform(10) languag(5) larg(1) list(1) main(1) match(3) mechan(1) monolingu(1) multilingu(1) natur(1) number(1) oper(1) peculiar(1) phase(3) pipelin(1) prerequisit(1) produc(3) queri(2) rang(1) rank(1) relev(1) represent(2) respons(1) retriev(5) satisfi(1) search(1) spoken(1) stand(1) step(1) structur(1) suitabl(1) system(1) task(3) text(2) textual(1) underli(1) understand(1) unstructur(1) user(2) valid(1) wai(1) wide(1).
«index structures in IR systems»
index(1) ir(1) structur(1) system(1) index(1) ir(1) structur(1) system(1)

[7] Contrairement à cet exemple, dans les systèmes réels, un sac de mots n'a pas d'ordre particulier.

Le paradigme *Bag of words*, permet d'identifier les traits associés à un document donné, et l'index inverse permet d'identifier les documents qui sont associés à un trait donné. Le système maintient une structure de données (une table de hachage), qui permet une consultation efficace d'un trait en retournant une liste de tous les documents contenant le trait donné. En effet, pour une requête à n traits, le système doit effectuer n consultations dans l'index inversé en collectant les listes de documents contenant chaque trait, afin de produire un score de similarité pour chaque document. Le système, doit par la suite trier les documents par score, et retourner la liste triée. La Figure 1.2 montre comment les paires d'index inverse associent chaque trait (*terme*) dans la collection à une table de paires (*identifiant du document, la fréquence*), qui dénotent combien de fois ce terme apparaît dans un document.

Pour la tâche d'appariement, la liste des techniques décrites ci-dessus n'est pas exhaustive, il existe une variété de schémas qui sont appliquées dans les différents modèles de recherche d'information existants. Ces modèles sont décrits brièvement dans la section suivante.

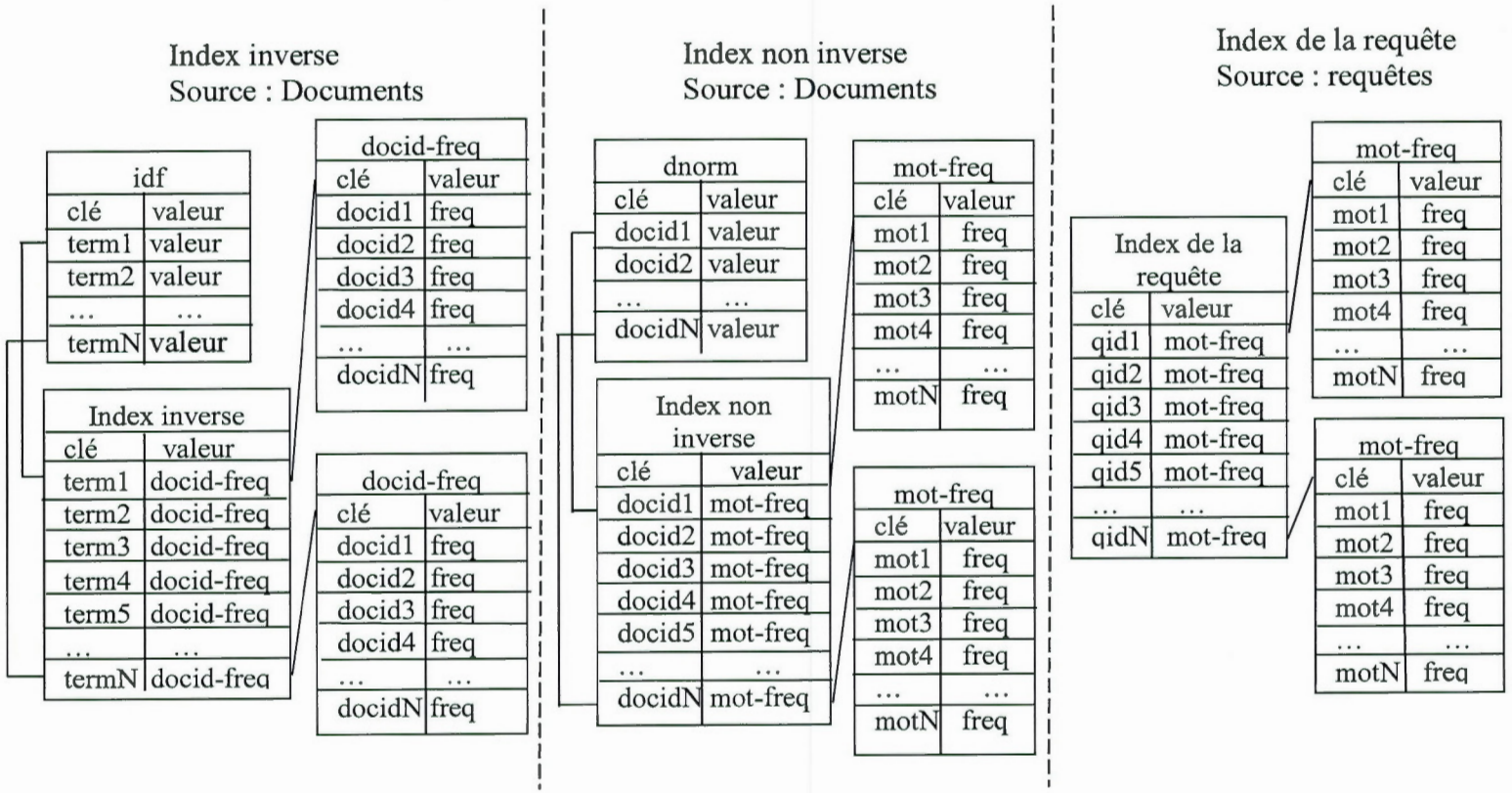


Figure 1.2 Indexes généralement utilisés pour la recherche[8]. Source: (Peters et al. 2012) [13].

[8] «*L'index inverse*» est obligatoire pour calculer efficacement la plupart des schémas de pondération. Certains shémas de pondération bénéficient également de la construction d'un «*index non-inverse*» et d'un «*index de la requête*» analogues.

1.3 Les modèles de recherche d'information

Dans la littérature, il y a eu un développement rapide des modèles de recherche d'information, en raison de l'énorme quantité des textes disponibles dans Internet. Les modèles les plus importants sont décrits dans les paragraphes qui suivent.

1.3.1 Le modèle booléen

Le modèle de recherche booléen est un modèle simple, basé sur l'utilisation de la théorie des ensembles et de l'algèbre de *Boole.* En effet, les représentations des documents et de la requête sont des expressions logiques formées à l'aide des opérateurs : «$\wedge$: *And*», «$\vee$: *Or*» et « $\neg$: *Not*».

Dans ce modèle, un document est une conjonction logique de termes : $d = t_1 \wedge t_2 \wedge ... \wedge t_n$, et une requête est une expression logique quelconque de termes : $q = (t_1 \wedge t_2) \vee (t_3 \wedge t_4)$. Tout document qui satisfait l'expression de la requête, est prédit comme pertinent, ce résultat est fourni par la relation de correspondance R comme suit [14]:

- $R(d,t_i) = 1$ *si* $t_i \in d$; *0 sinon.*
- $R(d, q_1 \wedge q_2) = 1$ *si* $R(d, q_1) = 1$ *et* $R(d, q_2) = 1$; *0 sinon.*
- $R(d, q_1 \vee q_2) = 1$ *si* $R(d, q_1) = 1$ *ou* $R(d, q_2) = 1$; *0 sinon.*
- $R(d, \neg q_1) = 1$ *si* $R(d, q_1) = 0$; *0 sinon.*

Exemple : Soient quatre documents, d_1 = «*Information Retrieval has 2 models and Information*», d_2 = «*Boolean is a basic Information Retrieval classic model*», d_3 = «*Information is a data that processed, Information*», et d_4 = «*When a Data Processed the result is Information, Data*» [15].

Pour la requête q = («*Data*» $\wedge$ «*Information*») $\vee$ ($\neg$ «*Retrieval*»), le résultat de la recherche est : $\{d_3, d_4\}$, car les documents d_3 et d_4 contiennent les termes «*Data*» et «*Information*», et le terme «*Retrieval*» n'appartient à aucun d'eux.

Bien que ce modèle soit simple, il retourne une liste de documents non triés, ce qui peut rendre difficile la distinction des documents pertinents désirés et surtout si la liste des résultats est longue. En outre, la requête peut être mal exprimée par des utilisateurs maitrisant peu (ou pas) l'utilisation des opérateurs logiques. Des extensions de ce modèle ont été proposées afin de remédier aux dits inconvénients [14].

1.3.2 Le modèle vectoriel

Les documents ainsi que la requête sont représentés dans un espace vectoriel basé sur la pondération des termes. Ainsi, chaque poids dans le vecteur du document ou de la requête désigne l'importance des termes qui leur correspondent. Une base de données contenant ***d*** documents, décrit chacun par ***t*** termes, est représentée par une matrice (Figure 1.4) dont les colonnes sont formées par les ***d*** vecteurs. Chaque élément a_{ij} de la matrice, représente la fréquence pondérée du terme ***i*** dans le document ***j***. Une variété de schémas est disponible pour pondérer les éléments de la matrice [14] [15].

La Figure 1.3, montre la représentation des document d_1, d_2, et de la requête q dans un espace vectoriel de dimensions trois (*3*). Selon les trois vecteurs correspondants on a :

- le document d_1, contient *2* fois le terme t_1, *3* fois le terme t_2 et *5* fois le terme t_3.
- le document d_2, contient *3* fois le terme t_1, *7* fois le terme t_2 et *1* fois le terme t_3.
- la requête q, contient 2 fois le terme t_3.

Les éléments a_{ij}, ainsi que les b_i (Figure 1.4), peuvent prendre des valeurs binaires (présence/absence d'un terme) ou des valeurs pondérées. Le degré de correspondance entre le vecteur du document et celui de la requête, est déterminé par la mesure de leur similarité qui peut être calculée de plusieurs façons, voici quelques formules [14]:

$$- Sim(d, q) = \sum_i (a_i * b_i) \qquad (produit\ interne) \qquad (1)$$

$$- Sim(d, q) = \frac{\sum_i (a_i * b_i)}{\sqrt{\sum_i a_i^2 * \sum_i b_i^2}} \qquad (Cosinus) \qquad (2)$$

$$- Sim(d, q) = \frac{2\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2} \qquad (Dice) \qquad (3)$$

$$- Sim(d, q) = \frac{\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2 - \sum_i (a_i * b_i)} \qquad (Jaccard) \qquad (4)$$

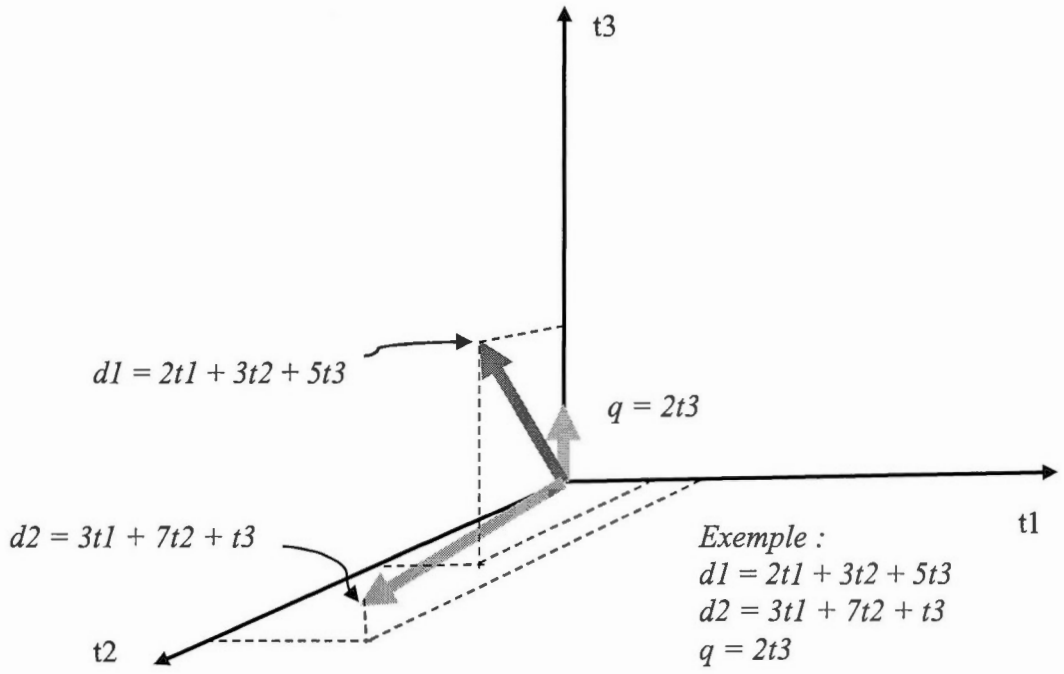


Figure 1.3 Modèle d'espace vectoriel.

Source : (Notes de cours : Sadat, 2013) [30].

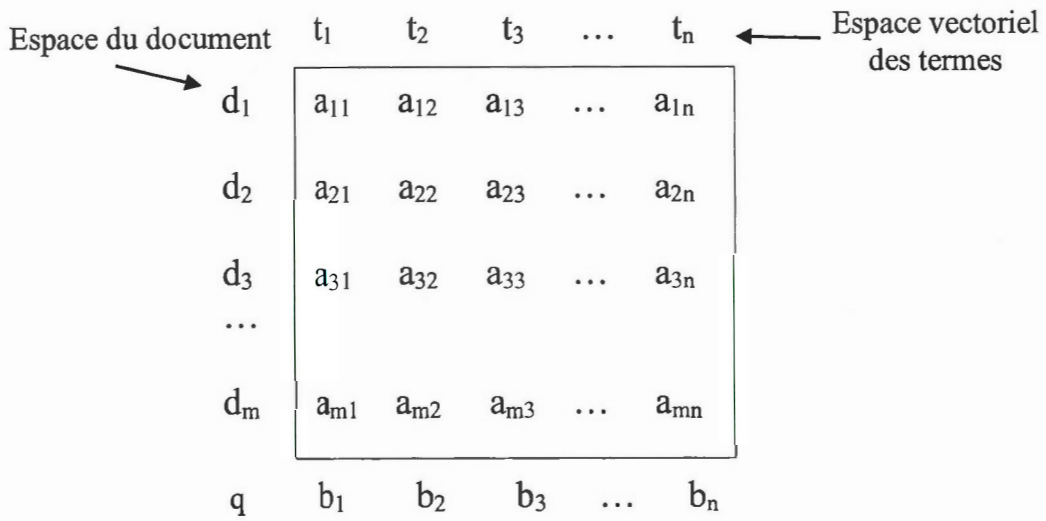


Figure 1.4 Représentation matricielle des documents et de la requête.
Source : (Notes de cours : Sadat, 2013) [30].

Exemple : Soit la requête q = *«A big house»,* pour une collection de trois documents d_1, d_2, et d_3, qui contiennent respectivement les textes «*A house and a garden*», «*The garden is big*», «*The house is big*».

La matrice suivante (Tableau 1.3), est le résultat de traitement obtenu durant la phase d'indexation, une liste triée des termes avec leurs fréquences. Pour chaque document, on a les vecteurs : $d_1 = (2, 1, 0, 1, 1, 0, 0)$, $d_2 = (0, 0, 1, 1, 0, 1, 1)$, $d_3 = (0, 0, 1, 0, 1, 1, 1)$. Pour la requête, on a le vecteur : $q = (1, 0, 1, 0, 1, 0, 0)$.

Le schéma de pondération appliqué dans cet exemple est le schéma «*Tf*Idf*», où *Tf* est la fréquence d'un terme, *Idf* = *log (D/Dfi)*, *D* est le nombre de documents dans la collection (*D=3*), et *Dfi* est la fréquence documentaire d'un terme (le nombre de documents contenant le terme ***i***). La matrice des poids a_{ij} obtenue (les colonnes «*Poids*») est montrée par le Tableau 1.4. En appliquant la formule (1) ci-dessus, la similarité entre le vecteur du document d_1 et celui de la requête est de *0.65*.

Le résultat retourné par ce modèle est une liste de documents triés par ordre décroissant de similarité.

Tableau 1.3 Matrice de fréquences des termes des documents d_1, d_2, et d_3 et de la requête q.

	q	d_1	d_2	d_3
A	1	2	0	0
And	0	1	0	0
Big	1	0	1	1
Garden	0	1	1	0
House	1	1	0	1
Is	0	0	1	1
The	0	0	1	1

Tableau 1.4 Matrice de poids des termes des documents d_1, d_2, et d_3 et de la requête q.

	Comptage			*Poids* (a_{ij}) = *Tf*Idf*			
	Dfi	*D/Dfi*	*Idf*	q	d_1	d_2	d_3
A	1	3/1	0.47	0.47	2*0.47	0	0
And	1	3/1	0.47	0	1*0.47	0	0
Big	2	3/2	0.17	0.17	0	1*0.17	1*0.17
Garden	2	3/2	0.17	0	1*0.17	1*0.17	0
House	2	3/2	0.17	0.17	1*0.17	0	1*0.17
Is	2	3/2	0.17	0	0	1*0.17	1*0.17
The	2	3/2	0.17	0	0	1*0.17	1*0.17

1.3.3 Le modèle probabiliste

Basé sur une méthode probabiliste, ce modèle permet d'estimer la probabilité de pertinence d'un document par rapport à la requête. Pour une requête donnée, il s'agit de déterminer les probabilités $P(R|D)$ et $P(NR|D)$ qui sont respectivement, les probabilités d'obtenir de l'information pertinente et non pertinente dans un document D retrouvé. Afin d'illustrer de manière simple le principe de cette méthode, les hypothèses suivantes sont considérés [16]:

- les termes ne sont pas pondérés;
- juste la présence (*1*) ou l'absence (*0*) d'un terme est prise en compte;
- la requête est fixe (Q).

Les probabilités $P(R|D)$ et $P(NR|D)$ correspondent donc, aux probabilités $P(R_Q|D)$ et $P(NR_Q|D)$ pour la requête Q. Une fois calculés, on peut classer les documents par ordre décroissant de la fonction «*Odd*» définie par la formule : $O(D) = P(R|D) / P(NR|D)$. En appliquant le théorème de *Bayes*, on a [16]:

$$- P(R|D) = P(D|R)\, P(R) / P(D), \quad (1)$$

$$- P(NR|D) = P(D|NR)\, P(NR) / P(D), \quad (2)$$

Où $P(D|R)$, est la probabilité que D appartient à l'ensemble des documents pertinents (R), $P(R)$ est la probabilité que le document choisi au hasard dans le corpus est pertinent, et $P(D)$ est la probabilité que le document choisi au hasard est D. Pour l'estimation des probabilités $P(D|R)$ et $P(D|NR)$ des formules (1) et (2), le document est décomposé en un ensemble d'évènements, une série de $(t_i = x_i)$, où x_i indique la présence ou l'absence (*0* ou *1*) du terme t_i dans le document D. Ainsi on a [16]:

$$- P(D|R) = P(t_1 = x_1, t_2 = x_2, t_3 = x_3, \ldots |R) \quad (3)$$

$$- P(D|NR) = P(t_1 = x_1, t_2 = x_2, t_3 = x_3, \ldots |NR) \quad (4)$$

Selon la théorie de probabilité et en supposant l'indépendance entre les évènements, les probabilités précédentes (les formules (3) et (4)) sont estimées comme suit [16]:

$$- P(D|R) = \prod(t_{i=}x_i) \in D \quad P(t_i = x_i \mid R) \quad (5)$$

$$- P(D|NR) = \prod(t_{i=}x_i) \in D \quad P(t_i = x_i \mid NR) \quad (6)$$

Finalement, il ne reste qu'à déterminer les probabilités $P(t_i = x_i \mid R)$, et $P(t_i = x_i \mid NR)$ des formules (5) et (6). Pour cela, il suffit d'avoir des échantillons de documents déjà

jugés pour une requête, et puis construire pour chaque terme t_i la table de distribution suivante (Tableau 1.5) [16]:

Tableau 1.5 Table de distribution pour un échantillon de N documents.

#doc pertinents contenant t_i : $\boldsymbol{r_i}$	#doc pertinents ne contenant pas t_i : $\boldsymbol{n - r_i}$	#doc pertinents : $\boldsymbol{n}$
#doc non pertinents contenant t_i : $\boldsymbol{R_i - r_i}$	#doc non pertinents ne contenant pas t_i : $\boldsymbol{N - R_i - n + r_i}$	#doc non pertinents : $\boldsymbol{N - n}$
#doc contenant t_i : $\boldsymbol{R_i}$	#doc ne contenant pas t_i : $\boldsymbol{N - R_i}$	#échantillons : $\boldsymbol{N}$

$$- P(t_i = 1 \mid R) = \frac{r_i}{n} \quad ; \; P(t_i = 0 \mid R) = \frac{n - r_i}{n}$$

$$- P(t_i = 1 \mid NR) = \frac{R_i - r_i}{N - n} \quad ; \; P(t_i = 0 \mid NR) = \frac{N - R_i - n + r_i}{N - n}$$

1.3.4 Le modèle de langue

La formulation de ce modèle est similaire au principe décrit dans le modèle probabiliste. Elle consiste à déterminer la probabilité $P(Q|D)$, qui désigne la probabilité que la requête Q peut être générée par le document D. Le modèle de langue est définit à l'aide d'une fonction de probabilité P (formule (1)), qui attribue $P(s)$ à chaque mot (ou à une séquence de mots) en une langue. Cette fonction permet d'estimer la probabilité qu'une séquence de mots quelconques peut être générée par le modèle de la langue [17].

$$P(S) = \prod_{i=1}^{l} P(m_i \mid m_1 \ldots m_{i-1}) \qquad (1)$$

Les modèles de langues utilisés en pratique sont les modèles *n-grammes,* dans lesquels un mot m_i ne dépend que de ses $n-1$ mots précédents. Les modèles les plus utilisés sont les *uni-grammes* (formule (2)), les *bi-grammes* (formule (3)), et les *trigrammes* (formule (4)) [17].

$$- P(S) = \prod_{i=1}^{l} P(m_i) \quad (2)$$

$$- P(S) = \prod_{i=1}^{l} P(m_i \mid m_{i-1}) = \prod_{i=1}^{l} \frac{P(m_{i-1}\, m_i)}{P(m_i)} \quad (3)$$

$$- P(S) = \prod_{i=1}^{l} P(m_i \mid m_{i-2} m_{i-1}) = \prod_{i=1}^{l} \frac{P(m_{i-2}\, m_{i-1} m_i)}{P(m_{i-2}\, m_{i-1})} \quad (4)$$

L'estimation des probabilités $P(m_i)$, $P(m_{i-1}\, m_i)$, $P(m_{i-2}\, m_{i-1}\, m_i)$ est effectuée à partir d'un corpus de textes C. Si le corpus est suffisamment grand, alors, il peut refléter la langue et peut représenter approximativement le modèle de langue pour ce corpus, la probabilité $P(\alpha|C)$ est calculée en utilisant la formule (5) suivante [17]:

$$P(\alpha|C) = \frac{|\alpha|}{\sum_{\alpha_j \in C} |\alpha|} = \frac{|\alpha|}{|C|}, \quad (5)$$

Où $|\alpha|$ est la fréquence d'occurrence du *n-gramme* α, et $|C|$ est le nombre total de mots dans le corpus. Ces estimations sont appelées les vraisemblances maximales (*ML : Maximum Likelihood*).

La notion de pertinence dans le modèle de langue, est donc en rapport avec la probabilité que la requête peut être générée par le modèle de langue du document (M_D). Le score du document D par rapport à la requête Q est : $Score(Q, D) = P(Q|M_D)$, et puisque la requête est représentée généralement comme une suite de termes, alors la formule du score peut s'écrire : $Score(Q, D) = P(t_1 t_2 \ldots t_n|M_D)$ [17].

1.4 Évaluation des systèmes de recherche d'information

L'objectif de tout système RI, est de fournir aux utilisateurs un accès facile à l'information pertinente à leurs besoins et de leur permettre de l'utiliser efficacement. La tâche d'évaluation consiste à mesurer le succès du système RI à atteindre le dit objectif. Ce succès, dépend d'un certain nombre de facteurs, y compris le mécanisme d'extraction des données utilisées, le mécanisme de traitement de la requête de

recherche, ainsi que la manière dont les résultats sont présentés et dans quelle mesure ils répondent aux attentes de l'utilisateur. L'évaluation dans la RI joue également un rôle très important, grâce aux progrès réalisés dans la recherche et dans le développement des systèmes RI, qui ont été atteints à travers les résultats d'expériences visant à évaluer la contribution particulière d'une caractéristique du système, ou d'un mécanisme de recherche [13].

Il existe deux grandes catégories d'évaluation de la RI : orientée système (*System-oriented*) et orientée utilisateur (*User-oriented).* L'évaluation *User-oriented* permet de mesurer la satisfaction de l'utilisateur avec le système, et comme elle a tendance d'être coûteuse et difficile à faire correctement, les chercheurs de la RI se sont surtout concentrés sur l'évaluation *System-oriented* pendant plusieurs années. Cette dernière, vise à mesurer la performance d'un système RI ou d'une stratégie de recherche dans un environnement objectif contrôlé [13].

De nombreuses mesures ont été proposées pour évaluer les performances des systèmes RI en se basant sur les collections de test. Au cours de la recherche, différentes stratégies d'indexation et d'appariement, et différentes techniques de traduction mèneront à différents ensembles de documents pertinents. La précision et le rappel sont les mesures de base de quantification de l'efficacité d'un système RI. Elles sont basées sur l'hypothèse que l'utilisateur veut extraire autant d'éléments pertinents que possible, et extraire peu d'éléments non pertinents que possible. Pour calculer ces mesures, les formules (1) et (2) suivantes sont utilisées [13] :

$$Precision = \frac{le\ nombre\ d'elements\ pertinents\ dans\ l'ensemble^{9}}{le\ nombre\ total\ d'elements\ dans\ l'ensemble^{10}} \qquad (1)$$

[9] Il s'agit de l'ensemble des documents pertinents trouvés par le système RI.

[10] Il s'agit de l'ensemble des documents (pertinents ou non) trouvés par le système RI.

$$Rappel = \frac{le\ nombre\ d'élements\ pertinents\ dans\ l'ensemble}{le\ nombre\ d'élements\ pertinents\ dans\ la\ collection} \quad (2)$$

Selon ces formules, on peut remarquer qu'une précision parfaite est obtenue lorsque l'ensemble des documents trouvés contient seulement les pertinents, et un rappel parfait est atteint, lorsque tous les éléments pertinents sont contenus dans l'ensemble. Comme le rappel est indépendant de la taille de l'ensemble des documents trouvés, un simple renvoi de toute la collection pour chaque requête de recherche, réalise un rappel parfait qui conduirait certainement à une très faible précision. Ces deux mesures ont toujours une relation inverse, une augmentation de la précision, entraîne une diminution du rappel et vice-versa (Figure 1.5). Par conséquent, elles doivent être adaptées pour une utilisation avec la liste des résultats classés du système RI [13].

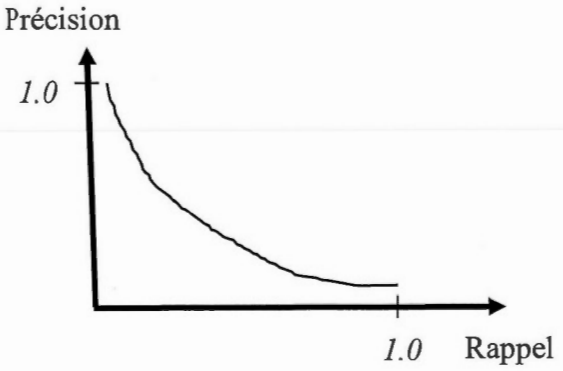


Figure 1.5 Courbe Rappel / Précision.

Pour faciliter la comparaison, la précision moyenne (*MAP : Mean Average Precision*) est une mesure fréquemment utilisée, et est devenue le principal indicateur dans de nombreuses évaluations. Cette mesure est définie par la formule (3) suivante [13]:

$$MAP = \frac{1}{|Q|}\left(\sum_{q_i \in Q} AP_i\right) \quad (3)$$

Et

$$AP_i = \frac{1}{|D^{rel}(q_i)|} \left(\sum_{r=1}^{|D|} \rho_r (q_i) * rel(r) \right) \qquad (4)$$

Où, D : est l'ensemble de documents, Q : l'ensemle de requêtes, q_i: une requête ($q_i \in Q$), d_j: un document ($d_j \in D$), $D^{rel}(q_i)$: l'ensemble des documents pertinents pour la requête q_i, $\rho_r(q_i)$: la valeur du rappel de l'ensemble $D_r(q_i)$, $D_r(q_i)$: les r premiers documents de l'ensemble D par rapport à la requête q_i, $rel(r) = 1$ si le document de rang r est pertinent, et $rel(r) = 0$ sinon. Le *MAP*, est la précision moyenne pour l'ensemble de requêtes Q, et AP_i est la précision moyenne pour la requête q_i (formule (4)).

1.5 Les compagnes d'évaluation

L'objectif d'une campagne d'évaluation est de soutenir, d'encourager la recherche en fournissant l'infrastructure nécessaire pour les essais à grande échelle, la comparaison des techniques et des méthodes, et pour augmenter la vitesse de transfert de la technologie. Les produits finis, sont des collections de test précieuses de ressources qui peuvent être utilisées pour l'évaluation des systèmes RI [13].

Les évaluations modernes de la RI ont commencé avec la première édition de la conférence TREC[11] (*Texte REtrieval Conference*) en *1992*. TREC est co-sponsorisée par l'institut NIST *(National Institut of Standards and Technology)* et le département américain de la défense. Les premiers exercices d'évaluation ont été présentés dans le domaine de la RI multilingue et translingue, qui a ouvert la voie par la suite au travail du forum CLEF[12] (*Cross-Language Evaluation Forum*) pour les langues européennes de l'institut NTCIR[13] *(National Institute of Informatics Text Collection for IR*) pour les

[11] http://trec.nist.gov/

[12] http://www.clef-campaign.org/

[13] http://research.nii.ac.jp/ntcir/

langues asiatiques, et du forum FIRE[14] (*Forum for Information Retrieval Evaluation)* pour les langues indiennes [13].

Pour la construction des collections de test, TREC a introduit une méthodologie qui a été également adoptée par NTCIR, CLEF et FIRE, tout en permettant les adaptations appropriées qui peuvent être appliquées pour créer leur propre corpus d'évaluation. Une collection de test inclut [13]:

- Les documents : C'est la collection de documents utilisée dans les expérimentations qui doit être appropriée pour la tâche de recherche à évaluer. Les journaux et les agences de presse se sont révélés être des sources généreuses, un grand nombre de collections de test les plus connues sont constituées de ce type de données. En outre, il y a d'autres bonnes sources à but non lucratif, telles que, les organisations gouvernementales et les archives nationales.

- Les requêtes : C'est un ensemble de requêtes sous une certaine forme, qui peut être obtenu à partir des journaux de requêtes des véritables sessions de recherche du monde réel, ou créé artificiellement comme une simulation des besoins réels d'informations. Cette dernière option est la plus utilisée, et est connue selon la terminologie de TREC sous le nom de *topic*. Généralement, les *topics* de TREC sont structurés en trois champs [13]:
  - Titre : est un brèf titre qui contient les mots-clés principaux de la requête.
  - Description : est une expression de la notion véhiculée par les mots-clés.
  - Narration : elle ajoute la syntaxe et la sémantique supplémentaires précisant les conditions d'évaluation de la pertinence.

Voici un exemple d'un *topic* anglais de la conférence CLEF- 2002 :

[14] http://www.isical.ac.in/~clia/index.html

Title: AI in Latin America
Description: Amnesty International reports on human rights in Latin America
Narrative: Relevant documents should inform readers about Amnesty International reports regarding human rights in Latin America, or on reactions to these reports.

- Les jugements de pertinence : À TREC et CLEF, la pertinence est déterminée sur une base binaire, un document est soit considéré comme pertinent ou non. Traditionnellement, l'évaluation de la pertinence implique un évaluateur humain, qui lit chaque document et décide s'il est pertinent ou non par rapport au *topic* en question. Afin d'assurer la cohérence du jugement, à TREC la personne qui créé un *topic* est responsable de l'évaluation de sa pertinence [13].

Les collections de test gratuites fournies par la conférence TREC ne répondent pas toujours aux besoins de tous les chercheurs pour effectuer leurs évaluations. En effet, certaines collections de test requises ne sont pas disponibles pour les non-participants, et sont dispendieuses. C'est pour cela que notre choix a été de construire notre propre modèle d'évaluation. Nous avons utilisé la collection de test gratuite anglaise de TREC-2002 (*AP88-90*), en faisant manuellement la traduction de ses requêtes de l'anglais vers l'arabe.

CHAPITRE II

LA RECHERCHE D'INFORMATION TRANSLINGUISTIQUE

Le *Web* est de nature multilingue, les internautes moyens par contre, sont généralement unilingues ou au mieux maîtraisant un petit nombre de langues différentes. En effet, les moteurs de recherche d'information translingues (RITs) sont proposés comme une solution partielle au problème de décalage entre l'utilisateur et l'information disponible (une barrière linguistique). Ces moteurs de recherche fournissent des moyens d'accès à l'information indépendamment de la langue d'écriture. Un système RIT est une spécialisation d'un système de recherche d'informations traditionnel, qui assume l'existence d'une requête (le besoin d'informations de l'utilisateur) et les documents (le corpus ou la collection de documents) [18].

La RIT est un sous-domaine de la recherche d'information (RI). Elle est basée sur la recherche des documents et des informations contenues dans ces documents. Dans la RIT, les requêtes et les documents sont écrits dans des langues différentes, contrairement à la RI où la requête et les documents sont dans la même langue. C'est pour cela, dans un système RIT, une traduction de la requête et / ou des documents est nécessaire avant d'effectuer la recherche. Le processus de traduction peut être effectué selon les trois modes [18]:

- Traduction de la requête dans la langue des documents (Figure 2.1).
- Traduction des documents dans la langue de la requête (Figure 2.2).

- Traduction des documents et de la requête dans une troisième langue (ou espace sémantique), ce processus est appelé *traduction duale* (Figure 2.3).

L'architecture du système RIT montrée par la Figure 2.1, consiste à effectuer une recherche d'information translinguistique en traduisant la requête (de la langue source) de l'utilisateur dans la langue des documents (vers la langue cible). Pour une comparaison appropriée et efficace, les documents et la requête sont représentés sous une forme interne en appliquant les techniques d'indexation détaillées dans le chapitre précédent. Par la suite, un appariement (correspondance) est lancé entre la représentation interne traduite de la requête dans la langue cible avec celle des documents. Enfin, les résultats de la recherche sont fournis à l'utilisateur sous forme d'une liste de documents triés par ordre décroisant de leurs scores de similarité.

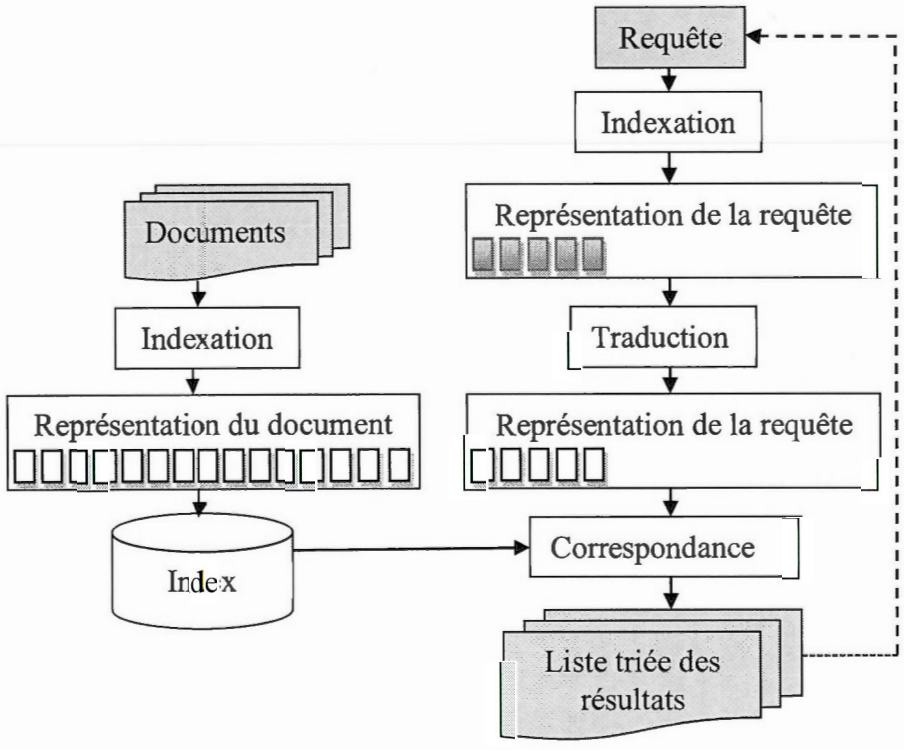


Figure 2.1 La recherche d'information translingue utilisant la traduction de la requête. Source: (ZHOU et al. 2012) [18].

Bien que la traduction de la requête est le mode le plus privilégié en raison de sa solution de calcul économique au problème de mésappariement, les deux derniers font encore l'objet de recherche.

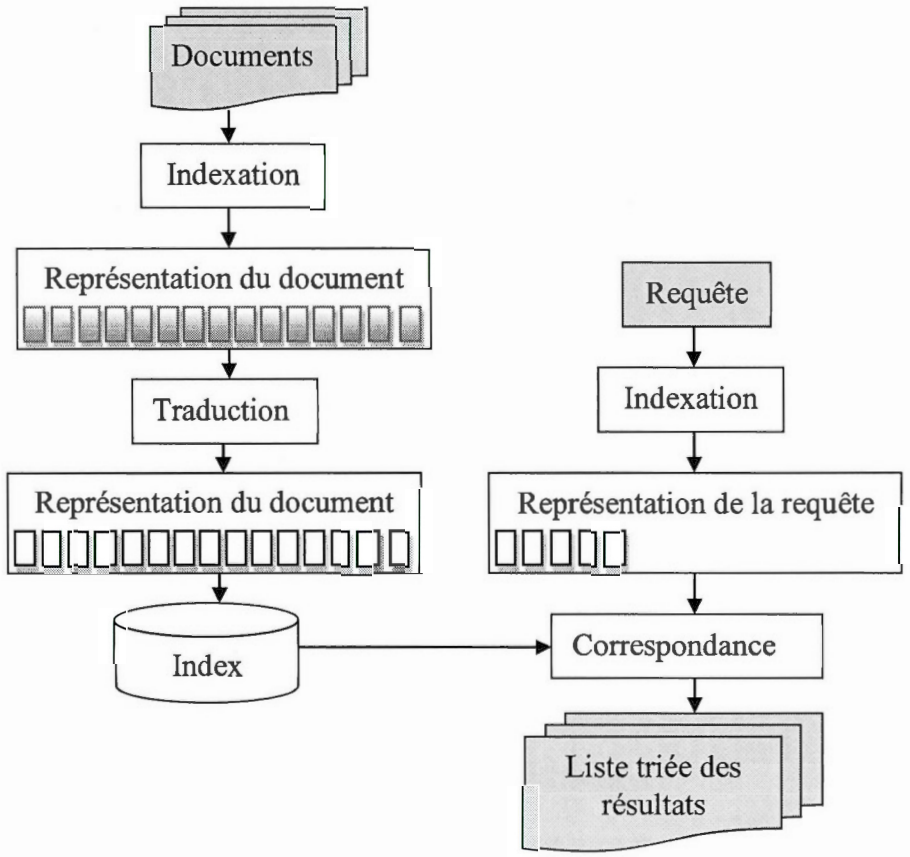


Figure 2.2 La recherche d'information translingue utilisant la traduction de documents. Source: (ZHOU et al. 2012) [18].

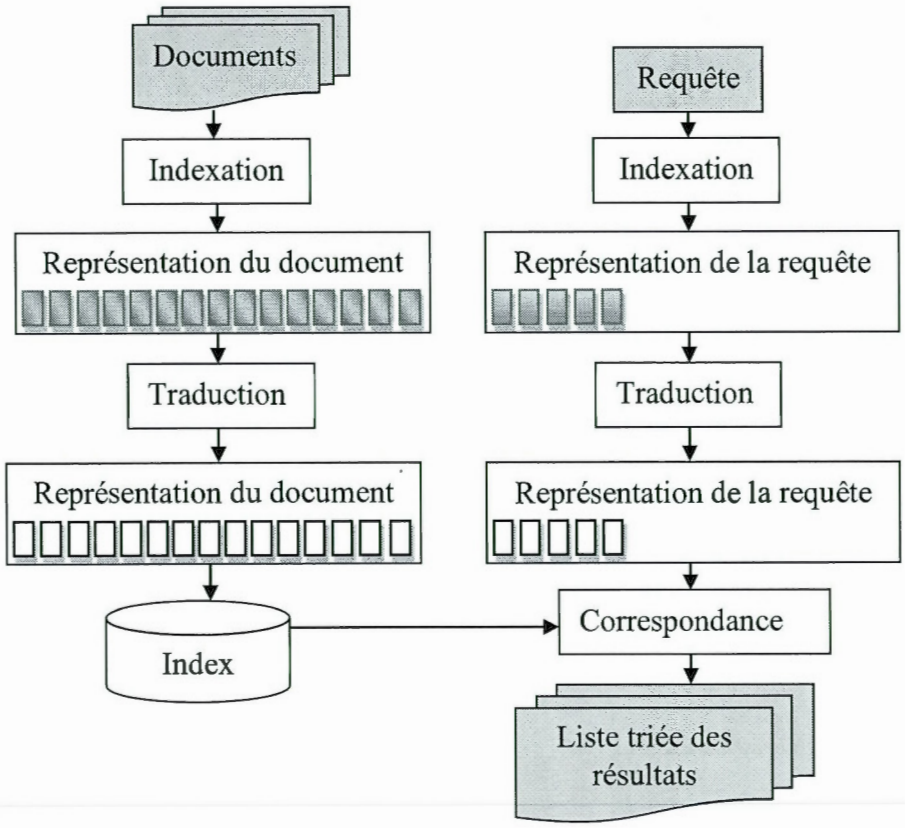


Figure 2.3 La recherche d'information translingue utilisant la traduction duale. Source: (ZHOU et al. 2012) [18].

2.1 Les modules d'un système RIT

Un système RIT orienté traduction peut être constitué d'un certain nombre de modules (Figure 2.4). Dans la plupart des moteurs RITs, les modules principaux sont : Un module de pré-traduction, un module de traduction, un module de post-traduction, et un module de recherche d'informations [18].

Les modules de pré-traduction et de recherche d'informations incluent les différentes étapes de prétraitement et de recherche déjà présentées dans la section précédente. En outre, le module de pos-traduction peut comprendre l'expansion du texte traduit en utilisant les techniques d'expansion mentionnées plus haut [18].

Les paragraphes qui suivent, seront donc consacrés au module de traduction qui représente le noyau de la chaîne du processus RIT. Dans le module de traduction, deux approches générales de traduction peuvent être employées (Figure 2.5) : la traduction directe, et la traduction indirecte [18].

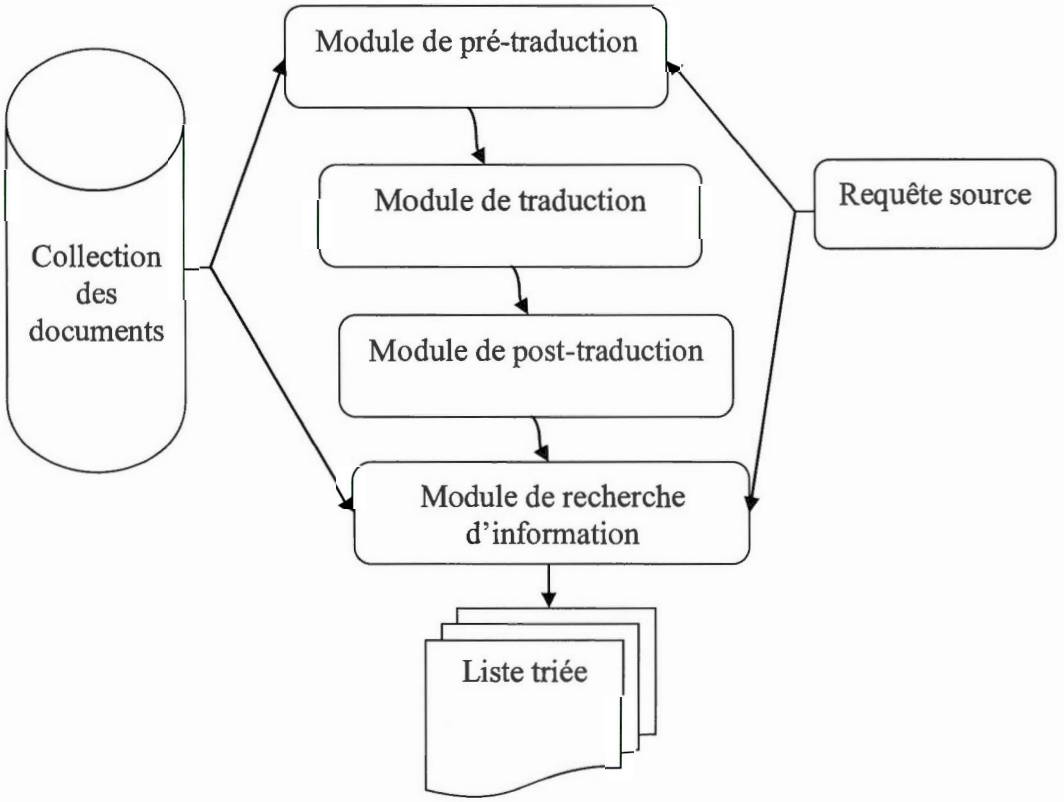


Figure 2.4 RIT, un cadre conceptuel. Source: (ZHOU et al. 2012) [18].

2.2 Les techniques de traduction dans les systèmes RITs

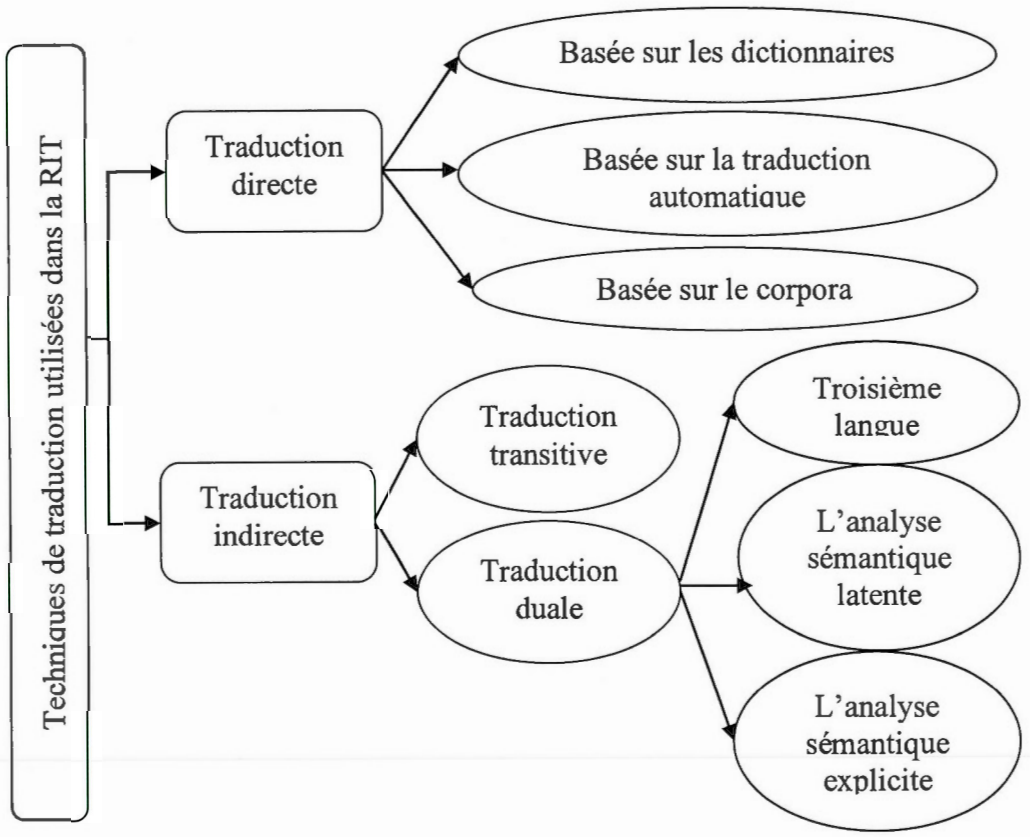


Figure 2.5 Techniques de traduction dans la RIT, une taxonomie.
Source: (ZHOU et al. 2012).

2.2.1 La traduction directe

Pour traduire un texte source, la traduction directe est basée sur l'exploitation des dictionnaires bilingues électroniques, des corpus parallèles, et des algorithmes de traduction automatique [18]:

- Les dictionnaires bilingues électroniques : C'est des dictionnaires sous une forme numérique, ils sont de plus en plus disponibles et sont souvent utilisés dans les modules de traduction des systèmes RITs. Bien que cette approche de traduction soit relativement simple, elle souffre de deux inconvénients majeurs,

à savoir, l'ambiguïté (multiples traductions pour un terme donné) et la faible couverture (le problème des termes hors vocabulaire (*OOVs* : *Out-Of-Vocabulary*).

- Les corpus parallèles : Cette approche est basée sur l'utilisation de corpus parallèles, qui sont de grandes collections de textes parallèles acquises à partir de divers sources. Dans ce genre de corpus, les documents sont alignés par phrase qui sont des traductions mutuelles l'une de l'autre. Une des caractéristiques importante de ce modèle, est qu'il ne fait aucune hypothèse sur l'ordre des mots. Une position dans la phrase cible peut être alignée à n'importe quelle position dans la phrase source. La probabilité d'alignement d'un mot à une position spécifique dans la phrase source, dépend de la présence du mot correspondant dans la phrase cible.

Ce modèle simplifié s'est avéré très efficace dans l'environnement RIT, dans lequel l'ordre des mots dans les requêtes est relativement peu important. L'un des principaux inconvénients de cette approche, est la difficulté inhérente d'obtention des collections de documents appropriées. La production des corpus parallèles est une tâche extrêmement fastidieuse, même si on se limite à un domaine spécifique. Pour cela, les corpus comparables (par exemple, *Wikipédia*) sont devenus une bonne alternative à utiliser.

Les corpus comparables sont définis par *Déjean* et *Gaussier (2002)* [56] comme suit : «*Deux corpus de deux langues l_1 et l_2 sont dits comparables s'il existe une sous-partie non négligeable du vocabulaire du corpus de langue l_1, respectivement l_2, dont la traduction se trouve dans le corpus de langue l_2, respectivement l_1*».

- La traduction automatique (*Machine Translation*) : Les systèmes de traduction automatique sont devenus très populaires ces dernières années, en raison de la disponibilité des ressources linguistiques nécessaires pour les former, mais aussi grâce aux excellents résultats obtenus dans les expériences. Par exemple, dans CLEF-2009, des systèmes RITs utilisant la traduction automatique ont atteints 90% à 99% de précision par rapport au système monolingue sur des collections de données anglaises, françaises et allemandes. Néanmoins, leurs performances peuvent être faibles, pour le cas des langues pauvres en ressources ou des paires de langues avec peu de points en communs (par exemple, l'anglais et le chinois). En outre, les systèmes de traduction automatique tiennent compte généralement de la structure syntaxique du texte à traduire (qui peut être sans importance). Aussi, ils ignorent souvent les OOVs qui ont un impact significatif sur l'efficacité de la recherche [18].

2.2.2 La traduction indirecte

En l'absence (ou manque) de ressources pour effectuer la traduction directe, la traduction indirecte se présente comme une solution, qui consiste en l'utilisation d'un intermédiaire entre la requête source et la collection de documents cible. Les différentes approches sont les suivantes [18] :

- Traduction transitive : La traduction transitive est basée sur l'utilisation d'une langue intermédiaire (ou pivot) entre la requête source et la collection de documents cible. Par exemple, pour traduire de manière transitive le mot en arabe « أمي: >*my*» vers l'anglais en passant par le français comme langue pivot, tout d'abord, le mot arabe sera traduit vers le français par «*Ma mère*», puis cette dernière traduction sera traduite vers l'anglais par «*My mother*». Dans certains cas, plusieurs langues pivots (traduction transitive triangulaire) sont utilisées, afin de remédier par exemple au problème de l'ambiguïté. Étant donnée des ressources de traduction appropriées, les résultats d'un système RIT avec un

moteur de traduction transitive peuvent s'approcher de ceux du système de traduction directe.

- Traduction duale : Ce genre de double traduction, tente de résoudre le problème de mésappariement entre la requête et les documents, en traduisant leurs représentations dans un troisième espace, avant d'entamer la comparaison (un langage humain, un langage abstrait, ou un interlingua). Ce type de traduction, comprend également des techniques qui induisent une correspondance sémantique entre la requête et les documents dans l'espace dual translingue défini par les documents, telles que, l'analyse sémantique latente (*LSA : Latent Semantic Analysis)*, et l'analyse sémantique explicite *(ESA : Explicit Semantic Analysis*).

2.3 La traduction automatique

La traduction automatiquement est basée principalement sur trois approches : La traduction automatique à base de règles (*RBMT : Rules-Based Machine Translation*), la traduction automatique statistique (*SMT : Statistic Machine Translation*), et la traduction guidée par l'exemple *(EBMT : Example-Based Machine Translation).* Pratiquement, ces approches peuvent être combinées pour obtenir des traductions de qualité.

2.3.1 La traduction automatique à base de règles

Les systèmes de traduction automatique à base de règles sont dotés d'un ensemble de fonctions permettant l'analyse du texte à traduire. En effet, cette analyse est effectuée en plusieurs étapes, une analyse morphologique, syntaxique et/ou sémantique, suivie du transfert lexical et structurel de la langue source en langue cible, et de la génération de la phrase cible. Pour la réalisation de ces différentes fonctions, ces systèmes utilisent les dictionnaires, ainsi que des grammaires dont les règles sont le plus souvent formulées par des linguistes. Par conséquent, en raison du problème d'acquisition et

de maintenance des connaissances linguistiques, le développement de ce type de systèmes est lent [19].

Cette approche de traduction est basée sur les trois méthodes suivantes [39] :

- Méthode de traduction littérale (*Literal Translation Method*) : S'appelle aussi, traduction directe, traduction à base de mots (*Word-Based Translation*), ou traduction à base de dictionnaires (*Dictionary-Based Translation*). Elle consiste à traduire les mots littéralement, c'est-à-dire mot-à-mot comme dans un dictionnaire. Les premiers systèmes de traduction ont été principalement basée sur cette méthode. En 1954, l'*IBM701* était le premier système de traduction automatique du monde basée sur une traduction littérale. En outre, à l'origine *Systran* était un système de traduction littérale typique, et il ne traduit que du russe vers anglais. Cependant, actuellement il permet la traduction entre différentes langues, et a une grande influence sur le développement de la traduction automatique.

- Méthode à base de transfert (*Transfert-Based Method*) : Basée sur la traduction mot-à-mot, elle permet d'analyser la structure de la phrase et génére le texte dans la langue cible en fonction des règles linguistiques des différentes langues. Pour cela, trois dictionnaires sont utilisées : le dictionnaire de la langue source, le dictionnaire de la langue cible, et un dictionnaire bilingue pour la paire de langues *source-cible*. Tout d'abord, cette méthode effectue une analyse morphologique et syntaxique (et parfois sémantique) du texte en entrée pour créer une représentation interne. À partir de cette dernière, les traductions seront générées en utilisant à la fois les dictionnaires bilingues et les règles de la grammaire, comme illustré par la Figure 2.6.

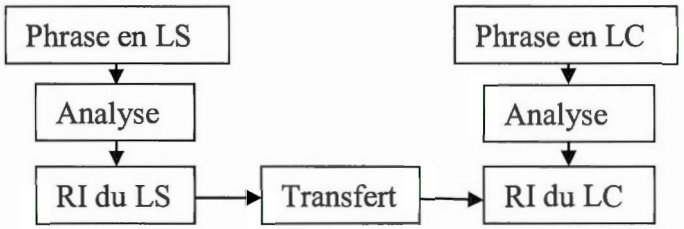


Figure 2.6 Traduction automatique à base de Transfert[15].
Source : (Peng, 2013) [39].

- Méthode basée sur l'interlingua (*Interlingua-Based Method*) : Est née grâce au développement de la méthode littérale et la méthode à base de transfert. Comme le montre la Figure 2.7, il s'agit en premier lieu d'analyser la langue source, puis la convertir dans une langue intermédiaire (l'*Interlingua*) qui est une représentation abstraite indépendante de la langue. Et enfin, la repésentation obtenue est convertie dans la langue cible.

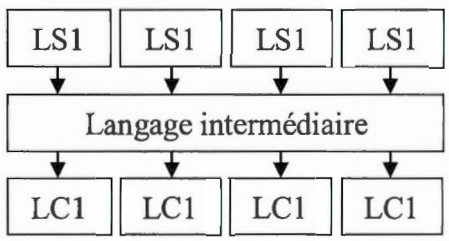


Figure 2.7 Traduction automatique à base d'interlingua.
Source : (Peng, 2013) [39].

Cette méthode peut être considérée comme la meilleure alternative par rapport aux deux méthodes précédentes, car avec l'*Interlingua* il devient inutile de faire des paires de traduction entre chaque paire de langues dans le système. En d'autres termes, au lieu de créer *N*(N-1)* paires de langues pour effectuer des

[15] LS = langage source, LC = langage cible, RI = représentation interne.

traductions multilingues, le système n'a besoin que de *2*N* paires entre les *N* langues et de l'*Interlingua* (*N* est le nombre de langues dans le système). Néanmoins, l'idéal de cette méthode est limitée dans un domaine spécifique très étroit, car il est trop difficile de définir une sorte d'*Interlingua* approprié.

Bien que les systèmes de traduction automatique à base de règles soient généralement largement utilisés à ce jour, ils sont confrontés au problème d'obtention des connaissances, qui sont écrites manuellement par des linguistes et ne peuvent pas répondre aux besoins réels de l'application. Par conséquent, l'amélioration de leur performance est une tâche très difficile surtout quand la base de règles devient de plus en plus grande.

2.3.2 La traduction automatique statistique

Elle est basée sur la théorie des probabilités, et en particulier, sur l'article de (*Brown et al., 1990)* [57]. Les systèmes statistiques sont basés sur l'apprentissage de deux modèles probabilistes, le modèle de traduction et le modèle de la langue cible, à partir respectivement du corpus bilingue et du corpus monolingue. La meilleure traduction est obtenue en maximisant la fonction suivante [19] :

$$\arg\max_{s} P(t|s) = \arg\max_{s} \{P(t) * P(s|t)\},$$

Où $P(s|t)$ est le modèle de traduction, et $P(t)$ est le modèle de langue.

Ces modèles sophistiqués, ont donné lieu à une famille de modèles qui sont actuelement l'état de l'art pour de nombreuses paires de langues, à savoir, les modèles de segments (*Phrase-Based Models*), où la probabilisation des appariements n'est plus entre les mots, mais entre des séquences de mots de taille variable. Dans cette approche, la traduction de la phrase source consiste à déterminer toutes les segmentations possibles en segments, ainsi que tous les équivalents de traduction des segments sources.

Et enfin, à trouver tous les arrangements possibles dans la langue cible [36]. Les systèmes de traduction automatiques statistiques (*SMTs)* se composent de trois principaux éléments [37] :

- Le modèle de langue : Est une séquence de suites de mots extraits à partir de textes. Ces mots ou groupes de mots sont appelés *n-grammes,* comme les *unigrammes*, les *bigrammes,* et peuvent aller jusqu'aux *octogrammes*. Pour ces modèles *n-grammes*, des tables sont crées pour contenir les mots ou les groupes de mots avec leur fréquence. Le Tableau 2.1 illustre un exemple de tables pour les modèles *three-grammes* et *quatre-grammes*.

  Ce modèle sert pour lever les ambigüités, c'est-à-dire qu'il permet de choisir le bon mot ou groupe de mots. En reconnaissance vocale par exemple, la phrase «*Je me dirige vers le mur*» présente deux ambiguïtés homophoniques pour le mot «*vert*» et le mot «*mur*», où le système doit décider s'il s'agit de «vert», «*verre*», «*vair*» ou «*vers*», puis s'il s'agit de «*mur*» ou de «*mûre*». Dans les tables déjà crées, il trouvera probablement le groupe de mots «*vers le mur*» qui est le plus fréquent.

- Le modèle de traduction : Est un ensemble de tables de groupes de mots et de leurs équivalents de traduction créées à partir de corpus bilingues alignés. Ce modèle permet de trouver les mots en langue source et de les remplacer par leur équivalents en langue cible. En partant des phrases alignées, ce modèle procède à un alignement de mots. Et par élimination, il atteint le bon mot ou le groupe de mots dans la plupart des cas. Par exemple, «*chat = cat*» et que «*Canada = Canada*». Par la suite, le même processus est effectué pour les *bigrammes*, *trigramme*, jusqu'aux *octogrammes*. Par exemple, même si on a «*time = temps*» et «*flies = mouches*», la traduction de «*Time flies*» sera «*Le temps passe vite*».

Tableau 2.1 Un exemple de tables des modèles *three-grammes* et *quatre-grammes*. Source : (Notes de cours : Sadat, 2013) [30].

Exemple sur le modèle 3-gramme	Exemple sur le modèle 4-gramme
ceramics collectables collectibles 55	serve as the incoming 92
ceramics collectables fine 130	serve as the incubator 99
ceramics collected by 52	serve as the independent 794
ceramics collectible pottery 50	serve as the index 223
ceramics collectibles cooking 45	serve as the indication 72
ceramics collection , 144	serve as the indicator 120
ceramics collection . 247	serve as the indicators 45
ceramics collection </S> 120	serve as the indispensable 111
ceramics collection and 43	serve as the indispensible 40
ceramics collection at 52	serve as the individual 234
ceramics collection is 68	serve as the industrial 52
ceramics collection of 76	serve as the industry 607
ceramics collection | 59	serve as the info 42
ceramics collections, 66	serve as the informal 102

Le Tableau 2.2, montre un modèle de traduction pondéré basé sur les segments pour la paire de langue *allemand-anglais*. Dans chaque ligne, il y a le mot source en allemand suivi de sa traduction en anglais et de la probabilité conditionnelle correspondante. Pour la première ligne par exemple, la probabilité que le mot «*der*» soit la traduction du mot «*the*» (*P(der|the)* est de *30%*.

- Le décodeur : Il est appelé *décodeur*, car les langues en traduction automatique sont considérées comme des codes qu'il faut décoder. Son rôle est de trouver des segments, des plus longs aux plus courts, à partir du texte en langue source en appliquant le modèle de langue de la langue cible. En conséquence, on peut éviter des traductions loufoques, comme «*mouches du temps*» pour traduire «*time flies*».

Tableau 2.2 Un exemple d'une table de traduction pour la paire de langue allemand-anglais. Source : (Notes de cours : Sadat, 2013) [30].

der ||| the ||| 0.3
das ||| the ||| 0.4
das ||| it ||| 0.1
das ||| this ||| 0.1
die ||| the ||| 0.3
ist ||| is ||| 1.0
ist ||| 's ||| 1.0
das ist ||| it is ||| 0.2
das ist ||| this is ||| 0.8
es ist ||| it is ||| 0.8
es ist ||| this is ||| 0.2
ein ||| a ||| 1.0

Généralement, les systèmes de traduction automatiques basés sur cette approche sont construits à partir des corpus parallèles. Cependant, en raison de l'absence de ce type de ressources pour les langues peu dotées, les corpus comparables qui sont plus disponibles sont également utilisés. L'encyclopédie *Wikipédia* est considéré comme un corpus comparable, et vu l'apport de son utilisation dû à la large couverture des termes, elle a été exploitée dans plusieurs travaux de recherche pour la construction des ressources linguistiques [58].

Les performances des SMTs sont évaluées au sens de la métrique *BLEU,* qui mesure la qualité des hypothèses de traduction retournées par les SMTs en les comparant à des références de traduction produites par des humains. C'est pour cela, le développement de systèmes performants nécessite la disponibilité d'un grand volume de corpus

bilingues parallèles, qui est une ressource relativement rare pour certaines paires de langues. Pour remédier à cet inconvénient, plusieurs travaux de recherche ont proposé l'intégration de ressources complémentaires telles que, des logiciels d'analyse linguistique, des dictionnaires, des terminologies bilingues, ou autres collections documentaires [36].

2.3.3 La traduction guidée par l'exemple

L'approche EBMT est devenue populaire à la suite des résultats positifs publiés dans un certain nombre de travaux de recherche. Elle est basée sur l'existence d'un grand volume de textes bilingues parallèles, traduits par des professionnels maîtrisant la langue et ayant une expertise spécialisée. En effet, un exemple est un couple de textes dans deux langues différentes, dont l'un est la traduction de l'autre. Les textes peuvent être de n'importe quelle taille à tout niveau linguistique (mot, expression, phrase, paragraphe, paragraphe d'événement). Les exemples peuvent être décomposés dans des sous-structures ou des exemples plus courts, pour qu'ils puissent se superposer les uns aux autres. Par conséquent, le nombre d'exemples peut être exponentielle par rapport à la taille du corpus, ce qui influt sur l'aspect pratique et la plausibilité de cette approche [38].

Cette méthode est caractérisée par une traduction par analogie, elle consiste à rechercher les meilleurs exemples de référence dans une base de données, ou à modifier et adapter des séquences de mots qui diffèrent des exemples de la base ou des mots à traduire [19].

En général, ce processus s'execute en quatre étapes : L'acquisition des exemples, la gestion de la base d'exemples, l'application des exemples et la synthèse de la phrase cible. La première étape consiste à savoir comment acquérir les exemples à partir des corpus bilingues parallèles, la deuxième étape porte sur la façon dont les exemples sont stockés et maintenus, la troisième étape se préoccupe de la décomposition de la phrase d'entrée en exemples et de la conversion de textes sources en textes cibles, et enfin, la

dernière étape permet de composer la phrase cible en mettant les exemples convertis dans le bon ordre afin d'améliorer la lisibilité après la conversion [38].

2.4 Google Translate

Google Translate est un service de traduction gratuit en ligne, dans environ 66 langues différentes. Ce système est basé sur une approche de traduction automatique statistique combinée à une approche à base de règles. Pour obtenir la meilleure traduction, il effectue une recherche des modèles dans des documents, tels que, les textes des livres, de l'*Organisation des Nations Unies,* et des sites Web. C'est pour cela que *Google Translate* peut obtenir des versions intelligentes de la traduction appropriée à partir des modèles statistiquement significatifs [20].

Pour le développement d'un système de traduction automatique statistique solide pour une nouvelle paire de langues, *Google* recommande d'avoir un corpus de textes bilingues de plus d'un million de mots, et de deux corpus monolingues de plus d'un milliard de mots chacun. Les modèles statistiques obtenus seront utilisés par la suite pour la traduction entre les langues [35].

À force d'analyser des documents dans une langue donnée, *Google Translate* produit une traduction de meilleure qualité. Par conséquent, l'exactitude de la traduction pour les langues, où il n'y a pas suffisamment de documents traduits peut varier. Avec *Google Translate,* la participation de l'utilisateur est envisagée en lui permettant de sélectionner parmi les choix des traductions possibles la meilleure traduction. Comparé à d'autres systèmes de traduction automatique très populaires, tels que *Promp, Babylon* et *Bing, Google Translate et Systran* sont toujours classés dans les premiers rangs [20].

2.5 MyMemory [16]

MyMemory est une très grande mémoire de traduction, elle permet de stocker des segments de textes et leurs traductions. Son architecture est collaborative centralisée, tout le monde peut participer via Internet, et la qualité de son contenu est soigneusement contrôlée. Afin de faciliter les recherches ultérieures, *MyMemory* télécharge des millions de pages Web traduites pour l'extraction de nouveaux segments, qui seront classés par source, sujet et qualité. De plus, elle permet la création des mémoires personnalisées pour des projets spécifiques, en répondant aux besoins des traducteurs et des fournisseurs de services linguistiques. Ces mémoires faites sur mesure peuvent être ouverte avec n'importe quel outil de traduction assisté par ordinateur (*TAO*), comme le *CafeTran, Déjà Vu-DVX2, Fluency, Google Translator Toolkit* ou autres.

2.6 L'encyclopédie *Wikipédia* [17]

Wikipédia provient du mot «*Wiki*» (un système de gestion du contenu de site Web), et du mot «*pédia*» (encyclopédie), elle a été créée par *Jimmy Wales* et *Larry Sanger* le *15 janvier 2001*. C'est une encyclopédie multilingue, réalisée de façon collaborative sur Internet et l'édition de ses articles est ouverte à tous les internautes, qui ont développé plusieurs règles et recommandations pour améliorer la qualité de son contenu. *Wikipédia* regroupe les articles rédigés dans la même langue (sa version dans cette langue). Jusqu'au *6 avril 2014*, il y a *287* éditions par langue, le nombre total d'articles de l'ensemble des éditions de *Wikipédia* est *31 214 669*. La langue *arabe* compte *271 850* articles à la même date.

Les pages de *Wikipédia* sont regroupées dans différents espaces de noms (*«Principal», «Discussion», «Aide»*). L'espace «*Principal*» contient les articles encyclopédiques, où

---

[16] http://mymemory.translated.net/doc/fr/

[17] http://fr.wikipedia.org/wiki/Wikipédia

chacun d'eux est lié à une page de discussion permettant aux internautes de discuter de la rédaction de l'article. Les pages peuvent être organisées dans une ou plusieurs catégories, qui forment une hiérarchisation arborescente et thématique. En outre, elles sont reliées par des hyperliens internes (les mots en bleu), qui mènent le lecteur vers l'article correspondant au concept abordé.

Des liens inter-langues (dans le cadre à gauche de la page *Wikipédia*), permettent également de passer d'un article dans une langue donnée à l'article correspondant dans une autre langue. La syntaxe utilisée pour l'ajout des liens inter-langues dans une page *Wikipédia* est : [[*code de langue*:titre étranger de la page]], par exemple pour un titre en anglais, on peut écrire : [[*en*:Black Scoter]]. Par ailleurs, un simple clic sur l'un des hyperliens externes, déplace le lecteur vers d'autres sources d'information pour approfondir le sujet.

CHAPITRE III

CARACTÉRISTIQUES DE LA LANGUE ARABE

L'*arabe* [18], est une langue sémitique originaire de la péninsule *Arabique*. Parlée par environ *240 Millions* de locuteurs dans les pays arabes du *Moyen-Orient*, les pays de l'est de l'*Afrique* (*Égypte, Libye*), et dans les pays de l'*Afrique du Nord* (*Algérie, Maroc, Tunisie*). C'est la langue officielle de *26* états, et elle est aussi l'une des six langues officielles de l'*Organisation des Nations Unies*. L'*arabe* littéral est constitué de l'*arabe* classique (la langue de la poésie, du *Coran*, et de la civilisation *arabo-musulmane*), et de l'*arabe* standard moderne (une variante moderne de la langue *arabe*) enseignée dans les écoles contemporaines.

3.1 Alphabet de la langue arabe [18]

L'alphabet de la langue arabe (Tableau 3.1), est constitué de *29* lettres fondamentales (les consonnes) y compris la lettre *Hamza* «ء : '» [19] qui se comporte soit comme une lettre, soit comme un diacritique. Cet alphabet s'écrit de droite à gauche. Les notions de majuscule et minuscule n'existent pas. Les lettres changent de forme en fonction de leur position dans le mot (début, milieu, fin, isolée). Les diacritiques ou les voyelles (brèves, longues, ou autres) sont rarement notées, sauf dans les ouvrages didactiques ou religieux, ou pour lever des ambiguïtés. Un mot peut avoir plusieurs sens avec des

---

[18] http://fr.wikipedia.org/wiki/Arabe

[19] La lettre *Hamza* en arabe et sa translittération *Buckwalter*.

diacritiques différents. Pour l'encodage, plusieurs jeux de caractères sont utilisés pour la langue *arabe*, parmi ceux-ci, on peut citer, l'*ISO 88 ISO-8859-6* et l'*Unicode*.

Tableau 3.1 Alphabet de la langue arabe

Séparée / Isolée	Début	Milieu	Finale	Nom de la lettre	Séparée / Isolée	Début	Milieu	Finale	Nom de la lettre
ء	أ, إ, ؤ, ئ			hamza	ض	ضـ	ـضـ	ـض	ḍād
ا			ـا	ʾalif	ط	طـ	ـطـ	ـط	ṭāʾ
ب	بـ	ـبـ	ـب	bāʾ	ظ	ظـ	ـظـ	ـظ	ẓāʾ
ة, ت	تـ	ـتـ	ـة،ـت	tāʾ	ع	عـ	ـعـ	ـع	ʿayn
ث	ثـ	ـثـ	ـث	ṯāʾ	غ	غـ	ـغـ	ـغ	ġayn
ج	جـ	ـجـ	ـج	ǧīm	ف	فـ	ـفـ	ـف	fāʾ
ح	حـ	ـحـ	ـح	ḥā	ق	قـ	ـقـ	ـق	qāf
خ	خـ	ـخـ	ـخ	ḫāʾ	ك	كـ	ـكـ	ـك	kāf
د	—		ـد	dāl	ل	لـ	ـلـ	ـل	lām
ذ	—		ـذ	ḏāl	م	مـ	ـمـ	ـم	mīm
ر	—		ـر	rāʾ	ن	نـ	ـنـ	ـن	nūn
ز	—		ـز	zāy	ه	هـ	ـهـ	ـه	hāʾ
س	سـ	ـسـ	ـس	sīn	و			ـو	wāw
ش	شـ	ـشـ	ـش	šīn	ي	يـ	ـيـ	ـي	yāʾ
ص	صـ	ـصـ	ـص	ṣād					

Pour plus de détails sur l'utilisation des voyelles dans la langue *arabe*, voici quelques définitions et exemples illustratifs (Tableau 3.2), tout en montrant la position de ces diacritiques sur la lettre, leur prononciation, ainsi que leur effet sur la variation du sens du mot :

- Les voyelles brèves : Sont quatre petits signes (◌َ : *a*, ◌ُ : *u*, ◌ِ : *i*, ◌ْ :*o*)[20] qu'on place en dessous ou au-dessus de la lettre (appelés : «تشكيل» , «*t$kyl*», «*Diacritiques*»).
- Les voyelles longues : Sont représentées par les trois lettres (ا: *A*, و: *w*, ي: *y*), et la prononciation des lettres est beaucoup plus prolongée. Lorsque ces lettres ne jouent pas le rôle de voyelles, elles se comportent comme des consonnes.
- Autres voyelles : C'est le signe (◌ّ : ~), prononcé comme une lettre double et placé au-dessus, il y a également les trois signes (◌ً: *F*, ◌ٌ: *N*, ◌ٍ : *K*) ou doubles voyelles qui se prononcent à la fin du mot (appelés : «تنوين» , «*tnwyn*»).

Soit le mot en arabe «كتب: *ktb*» voyellé de différentes façons («كَتَبَ», «*kataba*», «*Il a écrit*»), et («كُتُبْ», «*kutubo*», «*Des livres*»). De même pour le mot «ذهب: **hb*», les deux voyellations («ذَهَبَ», «**ahaba* », «*Il est parti*»), et («ذَهَبْ», «**ahabo*», «*L'or*»). On peut observer que le sens d'un même mot change en fonction des diacritiques appliquées.

3.2 Morphologie de la langue arabe

La morphologie de la langue arabe est très riche, elle est à la fois fortement flexionnelle et dérivationnelle. Avec une racine de trois lettres on peut dériver environ une trentaine (*30*) de mots différents [40]. En effet, plusieurs mots arabes peuvent correspondre à un mot dans une autre langue, comme par exemple les mots arabes («كأس», «*k>s*»), («قدح», «*qdH*»), et («كوب», «*kwb*») correspondent au mot français «*Verre*», mais pour l'arabe, le premier mot désigne un verre s'il est plein, et les deux suivants s'il est vide.

---

[20] Les diacritiques arabes et leurs translittérations *Buckwalter*

Tableau 3.2 Exemple des voyelles arabes brèves et longues.

Nom de la voyelle	Lettre diactrisée	Translittération arabe de *Buckwalter*
Voyelles brèves		
Fatha(◌َ)	فَ	fa
Dhamaa(◌ُ)	فُ	fu
Kasra(◌ِ)	فِ	fi
Soukoun(◌ْ)	فْ	fo
Voyelles longues		
Alif (ا)	فَا	fA
Waw(و)	فُو	fwu
Ya(ي)	فِي	fyi
Autres Voyelles		
Chadda(◌ّ)	فّ	f~
Tanween avec Alif(ا)	فً	fF
Tanween Waw(و)	فٌ	fN
Tanween Ya(ي)	فٍ	fK

3.2.1 Structure d'un mot arabe

La structure d'un mot arabe est complexe, suite à l'agglutination de morphèmes lexicaux et grammaticaux. Un mot arabe peut désigner toute une phrase dans une autre langue, par exemple, le mot (*«أتتفكروننا», «>ttfkrwnnA»*) qui veut dire en français : «*Est-ce que vous vous souvenez de nous* ?». Il est donc souvent, sous forme d'une séquence de : *antefixe*, *préfixe*, *noyau*, *suffixe*, *postfixe* (Tableau 3.3) [27] [41].

Les *antéfixes*, sont généralement des prépositions agglutinées au début des mots. Les *préfixes*, sont représentés souvent par une seule lettre et indiquent la personne de conjugaison des verbes au présent de l'indicatif. Les *suffixes*, sont les terminaisons de conjugaison des verbes et les marques du dual, du pluriel, ou féminine pour les noms. Enfin, les *postfixes* représentent les pronoms attachés à la fin des mots. Tous ces affixes

peuvent être attachés à la racine du mot, et avoir donc une forme plus compliquée. Un exemple illustratif est montré par le Tableau 3.4 [27].

Pour obtenir le noyau («فاوض», «*fAwD*», «*Négocier*»)[21], le traitement appliqué sur le mot («ليفاوضونهم», «*lyfAwDwnhm*», «*De négocier avec eux*») du *Tableau 3.4*, est appelé *stemming*, il est basé sur la détermination de la racine du mot (l'entrée dans le dictionnaire). En outre, une autre technique qui peut être également utilisée, est le *stemming* léger qui consiste à tronquer le mot des deux extrémités [27].

Tableau 3.3 Les affixes arabes. Source : (Kadri et Nie, 2006) [27].

Les antefixes	Les préfixes	Les suffixes	Les postfixes
وبال ,وال ,بال ,فال ,كال ,ولل , ال ,وب ,ول ,لل ,فس ,فب ,فل , وس ,ك ,ف ,و ,ب ,ل	ا ,ن ,ي ,ت	تما ,يون ,تين ,تان ,ات , ان ,ون ,ين ,وا ,تا ,تم , تن ,نا ,ت ,ن ,ا ,ي ,و	كما ,هما ,كن ,هن , تي ,ها ,نا ,هم ,كم , ك ,ه ,ي
Le sens des prépositions respectivement : *et avec, le/la, et le/la, avec le/la, puis le/la, comme le/la, et à/pour le/la, le/la, et avec, pour, puis sera, puis avec, ensuite/pour, et sera, comme, alors, et, avec, à /pour.*	Lettres qui indiquent la personne de conjugaison des verbes au présent de l'indicatif.	Les terminaisons de conjugaison des verbes et les marques du dual/ pluriel/ féminine pour les noms.	Le sens des pronoms respectivement: *votre, leur, votre, leur, mon/ma, sa, notre, leur, votre, ta, son, ma/mon.*

Pour les mots arabes, les préfixes les plus fréquents à supprimer à partir du début du mot sont : {وبال (*wbAl, et avec le*), وال (*wAl, et le*), بال (*bAl, avec le*), فال (*fAl, puis le*), كال (*kAl, comme le*), لل (*ll, pour/à*), وب (*wb, et avec*), ول (*wl, et pour/à*), ولل (*wll, et*

[21] Le mot en arabe est suivi d'une translittération *Buckwalter*, et de sa traduction en français. Cette dernière, n'est pas fournie pour les mots représentants des terminaisons de conjugaison ou des signes du genre et du nombre.

pour/à), فب (*fbi, puis avec*), فل (*fl, puis pour*), ال (*Al, le*), ا (*A, -*), ب (*b, avec*), ل (*l, pour/à*), و (*w, et*) }.

Et les suffixes les plus fréquents à tronquer sont : {تي (*ty, mon*), هما (*hmA, leur*), وا (*wA, -*), ك (*k, ton/ta*), نا (*nA, notre*), هم (*hm, leur*), ون (*wn,-*), ات (*At, -*), ان (*An, -*), و (*w, -*), ين (*yn, -*), ها (*hA, sa*), ت (*t, -*), ي (*y, -*), ن (*n, -*), ه (*h, son*), ا (*A, -*)} [27].

Tableau 3.4 Une forme agglutinée d'un mot arabe qui signifie «*de négocier avec eux*». Source : (Kadri et Nie, 2006) [27].

Les antefixes	Les préfixes	La racine	Les suffixes	Les postfixes
لـ	يـ	فاوض	ون	هم
Le sens de la préposition «*à/pour*».	Lettre qui indique le mode et la personne de conjugaison.	*Négocier*	Terminaison de conjugaison.	Le sens du pronom «*eux*».

3.2.2 Catégorie d'un mot arabe

Un mot arabe peut être dans l'une des catégories suivantes [40] [41]:

- Le verbe : La plupart des verbes arabes sont dérivés d'une racine de trois (3) consonnes, et rarement de quatre ou cinq consonnes. Ils se déterminent à partir d'un radical par l'ajout de préfixes et (ou) de suffixes, comme c'est le cas de la langue française. Leur conjugaison dépend du :

  - temps (accompli : le passé, inaccompli : le présent, le futur);
  - nombre (singulier, pluriel, duel);
  - genre (féminin, masculin);
  - de la personne (première, deuxième et troisième);
  - mode (passif ou actif).

Voici un exemple du verbe («فتح», *«fth», «Ouvrir»*) composé de trois consonnes *(ح: H, ف: f, ت: t)* :

- l'accompli passé, au singulier, pour la 3ème personne, et au mode actif : masculin («فتح», *«fth», «Il a ouvert»*), et féminin («فتحت», *«Ftht», «Elle a ouvert»*).
- l'accompli passé, au pluriel, pour la 3ème personne, et au mode actif : masculin («فتحوا», *«ftHwA», «Ils ont ouvert»*), et féminin («فتحن», *«ftHn», «Elles ont ouvert»*).
- l'accompli passé, au duel, pour la 3ème personne, et au mode actif : masculin («فتحا», *«ftHA», «Ils ont ouvert»*), et féminin («فتحتا», *«ftHtA», «Elles ont ouvert»*).
- l'inaccompli présent, au singulier, pour la 3ème personne, et au mode actif : Masculin («يفتح», *«yftH», «Il ouvre»*), et féminin («تفتح», *«tftH», «Elle ouvre»*).
- l'inaccompli futur, au singulier, pour la 3ème personne, et au mode actif : masculin («سيفتح, سوف يفتح», «s*yftH, swf yftH», «Il ouvrira, Il va ouvrir»*), et féminin («ستفتح, سوف تفتح», *«stftH, swf tftH», «Elle ouvrira, Elle va ouvrir»*). Le futur est obtenu soit en ajoutant l'antéposition «س : *s*» ou « سوف : *swf*».

• Les noms : Les noms arabes appartiennent à deux classes, les noms variables dont la plupart sont dérivés à partir d'une racine verbale, et les autres sont invariables ou fixes comme les pronoms personnels (affixés ou isolés), les adverbes, et autres types de pronoms. Les noms de la première classe sont conjugables ou semi-conjugables, c'est-à-dire qu'ils peuvent avoir les accords du genre et du nombre, et ceux de l'autre classe sont non-conjugables, ils sont fixes quel soit leur contexte.

Voici un exemple illustratif de noms arabes fixes ou variables :

- noms fixes : les pronoms personnels («أنا, أنت, هو, هي, نحن, أنتم, أنتن, هم, هن», «*>nA, >nt, hw, hy, nHn, >ntm, >ntmn, hm, hn* », « *Je, Tu, Il, Elle, Nous, Vous, Vous, Ils, Elles*»), ou les adverbes («أين, حيث», «*>yn, Hyt*», «*Où, Où*»).
- noms variables dérivés d'une racine verbale : le nom arabe au singulier féminin («مكتبة», «*mktbp*», «*Bibliothèque*»), au pluriel féminin («مكتبات», «*mktbAt*», «*Bibliothèques*»), et le nom arabe au singulier masculin («كاتب», «*kAtb*», «*Écrivain*»), au pluriel masculin («كتاب», «*ktAb*», «*Écrivains*»), sont dérivés de la racine verbale («كتب», «*ktb*», «*Il a écrit*»).
- noms variables non dérivés d'une racine verbale : le nom arabe au singulier («رأس», «*r>s*», «*Tête*»), au pluriel c'est («رؤوس», «*r&ws*», «*Têtes*»), le nom au singulier («كرسي», «*krsy*», «*Chaise*»), le pluriel est («كراسي», «*krAsy*», «*Chaises*»), et le nom au singulier («أخ», «*>x*», «*Frère*»), le pluriel est («إخوة», «*<xwp*», «*Frères*»).

Le pluriel de certains noms arabes ne peut pas être obtenu en appliquant les mêmes règles que pour les autres noms, ce phénomène est celui du pluriel irrégulier qui suit une diversité de règles complexes dépendantes du nom. Comme par exemple, la forme plurielle des mots («إمرأة», «*<mr>p*», «*Femme*») et («طفل», «*Tfl*», «*Enfant*») qui sont repectivement («نسوة», «*nswp*», «*Femmes*») et («أطفال», «*>TfAl*», «*Enfants*»).

- Les particules : Les particules sont des lemmes invariables (comme les mots outils pour une langue donnée), ils servent à situer les événements et les objets par rapport au temps. On distingue plusieurs types :

- préposition : {(ك: *k*), (ل: *l*), (ب: *b*), (عن, *En, Sur/À propos de*), (:حتى , *HtY, Même/Encore/Voire*)}
- particules de coordination : {(أو, *>w, Ou/Ou bien*), (ثم, *vm, Puis/Ensuite*), (ف: *f*), (و: *w*)}
- particules interrogatives : {(ما: *mA, Qu'est-ce-que/Quoi*), (هلْ, *hlo, Est-ce-que*), (أ: *>*)}
- particules d'affirmation : {(أجَل, *>jal, Oui*), (بَلى, *balY, Oui*), (نـعَم, *nEam, Oui*)}
- particules de négation : {(لـم, *lm, Non/Ne pas*), (لـن, *ln, Non/Ne pas*), (لا, *lA, Non/Pas de/Aucun/Aucune*)}
- particules distinctive : {(أي, >y, *Comme*)}
- particules relatives : {(ما, *mA, Ce que*)}
- particules de futur : {(سـ: *s*), (سـوف, *swf, Sera/Ira*)}
- particules conditionnelles : {(إن, *<n, Si*), (لـو, *lw, Si*)}

En plus du problème de leur identification, les particules rajoutent de la complexité s'ils sont attachés à des préfixes et à des suffixes.

3.3 Les analyseurs morphologiques pour l'arabe

Comme pour les autres langues, les analyseurs morphologiques pour la langue arabe sont des outils qui s'interessent à l'étude des formes de mots. Cette analyse permet de déterminer les valeurs d'un grand nombre de traits ou d'attributs morphologiques d'une unité lexicale (un mot), comme la catégorie grammaticale (*Nom, Verbe, Adjectif, Préposition*, etc.) «*Part-Of-Speesh*» (*POS*), le genre, le nombre, etc.

Dans ce qui suit, nous allons cités quelques outils développés dans ce domaine basés sur différentes approches [49] [50]:

- L'analyseur de *Shaa-lan (1989)* : C'est un analyseur à base de règles (écrites en *SICStus Prolog),* et il utilise un ancien système de translittération. Ce système necessite une certaine maîtrise du *Prolog* difficile à réaliser par les linguistes.

- L'analyseur *Morph3* : Il est basé sur un modèle hybride, qui combine une base de connaissance des règles et une base de connaissances statistiques.

- L'analyseur *Sakhr* : Il permet de traiter les deux variétés de la langue arabe (l'arabe classique et moderne). Après l'extraction des suffixes et préfixes, cet analyseur fournit toutes les formes de base possibles d'un mot. Néanmoins, il ne s'intéresse pas à la désambiguïsation.

- L'analyseur *ARAMORPH* : Réalisé par *Tim Buckwalter*, avant tout traitement il effectue une translittération en *ASCII* du texte en entrée, et fournit un résultat reconverti en arabe. *ARAMORPH* n'accepte pas les textes contenant des chiffres de *0 à 9*.

- L'étiqueteur *Arabic Part-of-speech Tagger (APT)* de *Khoja* : Il combine des données statistiques et des règles techniques. Ses étiquettes (au nombre de *131*) sont dérivées du système d'étiquetage *BNC* de l'*Anglais*, et adaptées aux contraintes de la grammaire *Arabe*. Ce système a été entrainé sur un corpus de *50 000* mots du journal saoudien *Al-Jaziira*.

- L'étiqueteur *Freeman*: Il est basé sur la méthode d'*Eric Brill,* il utilise *146* étiquettes pour l'étiquettage des lexèmes.

- L'analyseur morphologique de *XEROX* : Réalisé par *Kenneth Beesley*, il utilise la technologie d'états finis *(FST : Finite State Technologie)* développée par *XEROX.*

- Le système de *Maamouri* et *Cieri* : Il utilise l'analyseur morphologique de *Tim Buckwalter.* Ce système est basé sur l'étiquetage automatique d'un corpus contenant *734* fichiers de l'*Agence France Presse (AFP).*

- L'analyseur morphologique *Sebawai de Darwish* : Développé par *Darwish* en 2003. Il permet de trouver les racines de mots avec un taux de réussite de *84%.*

- L'analyseur morphologique et désambiguïsateur de l'*arabe* (*MADA : Morphological Analyzer and Disambiguator of Arabic)* : Cet analyseur a été entrainé sur les données d'apprentissage du corpus «*Penn Arabic Treebank*» *(PATB*). Il permet d'effectuer la segmentation, la diacritisation, la lemmatisation, l'étiquetage grammatical et l'analyse morphologique. Pour chaque mot, MADA génère toutes ses analyses possibles et utilise par la suite une machine à vecteurs de support (*SVM* : *Support Vector Machines)* pour la prédiction de quelques traits morphologiques. Les analyses retournées par MADA sont hiérarchisées, et la meilleure analyse est celle qui s'accorde le plus avec la prédiction. L'inconvénient de cet analyseur est qu'il ne prend pas en considération les diacritiques de l'entrée.

CHAPITRE IV

ÉTAT DE L'ART

Les approches lexicales de traduction des requêtes pour la RIT se heurtent à des difficultés de couverture lexicale et d'ambiguïté. L'encyclopédie en ligne multilingue *Wikipédia,* propose des solutions à ces problèmes en mettant à disposition une quantité conséquente de connaissances, constamment mises à jour et accessibles. Il est donc possible d'en extraire des lexiques, dont la couverture est optimale et qui sont organisées sémantiquement par des catégories fournies par Zesh et al. (2007) [22].

En effet, *Wikipédia* a été le banc d'essai de nombreuses tentatives de l'exploitation des phrases parallèles. Grâce à la structure de ses articles liés au même sujet, elle est sans doute le plus grand corpus fortement comparable disponible en ligne après le *Web* lui-même [26].

Adafre et Rijke (2006), ont été parmi les premiers à extraire des phrases parallèles de *Wikipédia*. Leur approche est basée sur deux expériences, dans la première, ils ont utilisé le traducteur automatique *Babelfish* pour traduire de l'*anglais* vers le *néerlandais*, puis, par chevauchement des termes, la similarité entre les phrases traduites et les phrases originales est mesurée. La deuxième approche, exploite le lexique de traduction induit automatiquement à partir des titres des articles liés, puis, ils mesurent la similarité entre les phrases sources (en *anglais*) et cibles (en *néerlandais*) en les mappant aux entrées dans le lexique. Les expériences ont été effectuées sur 30 paires de documents *anglais-néerlandais* choisis au hasard, et produisant quelques centaines de paires de phrases parallèles. Mohammadi et

GhasemAghaee (2010) ont continué le travail d'Adafre et Rijke (2006), en imposant certaines limites sur les paires de phrases qui peuvent être formés à partir d'une paire de documents *Wikipédia*. La longueur des phrases parallèles candidates doit corréler, et la similarité de Jaccard des entrées du lexique mappé à la source et la cible doit être aussi élevée que possible. De même que pour Adafre et Rijke, le travail effectué par Mohammadi et GhasemAghaee, ne fait pas générer un corpus parallèle mais seulement quelques centaines de phrases parallèles conçues comme une preuve de concept [26].

Par ailleurs, Gaillard et al. (2010) ont proposé une méthode de traduction automatique de requêtes basée sur *Wikipédia*. Cette approche, s'appuie sur le seul lexique issu des titres des articles de *Wikipédia,* et maximise la taille des unités lexicales extraites sur la requête dans son ensemble. Le processus de traduction a été effectué en deux étapes, une étape de segmentation de la requête en unités lexicales qui sont traduites en s'appuyant sur les liens multilingues des articles *Wikipédia*. Pendant la deuxième étape, la traduction qui maximise l'homogénéité thématique est choisie parmi toutes les alternatives possibles. L'homogénéité thématique est la somme des proximités sémantiques de toutes les paires d'unités traduites. La proximité sémantique de deux alternatives est définie par la similarité du *cosinus* de leurs vecteurs de catégories. Cette mesure de proximité est fondée uniquement sur les catégories, ce qui offre une représentation plus concise du thème sémantique d'un article que celle issue du texte [22].

Sadat et Terrassa (2010) ont extrait des lexiques bilingues (*arabe-français* et *yoruba-français*) à partir de l'encyclopédie en ligne *Wikipédia* en exploitant les liens inter-langues des articles, au profit de la traduction automatique statistique [24]. Sellami et Sadat (2012) ont exploité la dite encyclopédie dans le but d'enrichir et de construire des ressources linguistiques (ontologies multilingues) [29]. En outre, Sadat (2010) a proposé une approche d'extraction d'une terminologie bilingue en exploitant *Wikipédia* comme un corpus comparable pour la construction des ressources linguistiques

(ontologies et dictionnaires), et pour nourrir les systèmes RITs avec des termes qui leur seront utiles pour l'expansion des requêtes. Sa méthodologie se déroule en plusieurs étapes. Premièment, les termes ainsi que leurs positions de discours *(POS : Part-Of-Speech), nom, verbe, adverbe, adjectif*, sont extraits à partir des documents sources et cibles. Deuxièment, le vecteur de contexte de chaque terme est construit pour les deux langues, en se basant sur l'information mutuelle comme mesure de la tendance de la cooccurence. Puis, les vecteurs de contexte sources sont traduits dans la langue cible en utilisant *Wikipédia* comme ressource de traduction. Pour la traduction d'un mot, cette étape utilise les liens inter-langues des articles *Wikipédia*, et le *Wiktionnaire* pour surmonter les limites de *Wikipédia*. Et enfin, les alternatives de traduction obtenues sont classées par ordre décroissant de similarité (entre les vecteurs de contexte sources et cibles) calculée en utilisant la mesure du *cosinus*. Les premières traductions de la liste classée sont retenues et les autres sont écartées. Les évaluations de cette approche ont été effectuées pour les langues *anglais, français et japonais* [43].

Le travail de Rahimi et Shakery (2010) s'est intéressé à une question importante dans le domaine de la RIT, à savoir, franchir la barrière linguistique entre la requête et les documents. Pour cela, ils ont construit un dictionnaire d'association bilingue *persan-anglais* basé sur *Wikipédia* pour la traduction des requêtes dans la RIT *persan-anglais*. Contrairement à d'autres approches qui utilisent entièrement les titres liés comme traductions, cette étude exploite les liens inter-langues de *Wikipédia* dans le but d'aligner les titres liés dans différentes langues, puis extraire les associations mot-à-mot à l'aide de la cooccurrence des mots dans les alignements, tout en calculant la similarté entre chaque paire de mots. Cette façon de faire a permis d'éviter l'apparition de mots sans rapport avec la requête dans la traduction, ce qui peut influencer négativement sur la performance du système de recherche. Pour la construction du dictionnaire d'association, cette approche utilise les titres de *Wikipédia* comme un corpus aligné par phrase. Par conséquent, en reconnaissant des structures typique dans les titres *Wikipédia*, les alignements sont divisés en phrases alignées plus courtes. Par

exemple, les caractères ":", "," et "()" sont souvent utilisés dans les titres pour séparer des concepts distincts ou pour expliquer des informations supplémentaires. Si le titre source est sous le pattron «*s_phrase1 (s_phrase2)*» et aligné au titre cible «*t_phrase1 (t_phrase2)*», alors cet alignement est divisé en deux aligments plus courts, «*s_phrase1*» est aligné à «*t_phrase1*» et «*s_phrase2*» est aligné à "*t_phrase2*". Comparativement à une simple approche à base de dictionnaires, les résultats d'évaluation de cette méthode ont montré une amélioration significative des performances de la recherche en terme de la précision moyenne [44].

La construction manuelle des dictionnaires bilingues est coûteuse, et la nouvelle terminologie spécifique à un domaine est difficile à couvrir. Afin de contribuer à satisfaire la demande de ce type de ressources dans les domaines de recherche, tels que la traduction automatique ou la recherche d'information multilingue, Erdmann et al. (2009) ont examiné cette question en se basant sur *Wikipédia* comme un corpus pour l'extraction automatique d'une terminologie bilingue. À la différence d'autres approches qui se sont limitées aux liens inter-langues, les auteurs ont analysés plusieurs types de liens de *Wikipédia* pour obtenir une couverture élevée des dictionnaires, tout en assurant une grande précision. Tout d'abord, cette méthode commence par créer un dictionnaire de base à partir des liens inter-langues de *Wikipedia*. Comme les pages de redirection sont souvent supposées des expressions synonymes, alors, pour un article dans la langue source *sp,* lié à un article *tp* dans la langue cible, les titres *t* et *s* sont considérés comme une paire de traduction de termes. Par la suite, le dictionnaire est renforcé en remplaçant *s* avec les titres de toutes les pages de redirection de *sp,* et de nouvelles paires de traduction sont alors ajoutées. De même pour chaque titre *t* remplacé par les titres de toutes les pages de redirection de *tp*. Et enfin, le dictionnaire est amélioré en remplaçant respectivement *s* et *t* avec les textes d'ancrage de tous les liens de retour internes de la page *sp* et ceux de la page tp, qui sont ajoutés aussi en tant que nouvelles paires de traduction. L'exactitude des paires de termes de traduction extraites a été effectuée à l'aide d'un classifieur basé sur une machine à vecteurs de

support (*SVM : Support Vector Machine*). Les résultats des expérimentations ont montré que le classifieur SVM a été efficace en termes de performances, en raison de l'extraction de nombreux traits différents dans l'ensemble des données d'entrainement [45].

Collin et al. (2010) se sont basés sur les données de *Wikipédia* pour la constitution automatique de ressources sémantiques lexicales. Cette approche permet d'affecter à chaque entrée d'un lexique pré-identifié, une ou plusieurs étiquettes permettant de la caractériser au sein d'une taxonomie ou un treillis organisé hiérarchiquement. Elle permet en outre, de différencier les entrées de même forme (homonymes) et de regrouper les entrées de sens équivalent (synonymes). L'espace de représentation est un sous-ensemble du treillis des catégories de *Wikipédia,* et dont la structure permet d'observer des relations d'hypéronymie et de catégorisation thématiques pertinentes. Ce travail a été utilisé dans deux applications, la première concerne l'indexation et la classification des pages *Wikipedia,* et la deuxième concerne la désambiguïsation dans le contexte d'un traducteur de requêtes pour la paire de langues *français-anglais* [23].

Dong et al. (2009) ont utilisé *Wikipédia* comme une ressource de traduction pour la RIT, en traitant les articles *Wikipédia* comme des représentations de concepts auxquels la requête sera mise en correspondance. Cependant, la couverture des mots communs dans *Wikipédia* est plus faible que les dictionnaires, et certains termes ont plusieurs sens, d'autres sont spécifiques et non communs. Par conséquent d'autres ressources (*WordNet*, *EuroWordNet* ou autres) peuvent être incorporées pour améliorer la qualité de la traduction. Cette approche diffère de l'approche traditionnelle, puisqu'elle utilise le texte et les liens internes qui ne sont pas disponibles dans les approches à base de dictionnaires et de corpus parallèles. Elle diffère également des autres approches utilisant *Wikipédia* comme celles de Su et al., et Schönhofen et al. qui ont utilisé *Wikipédia* pour améliorer leur traductions. L'avantage est qu'elle permet l'extraction

de phrases à partir des thèmes, depuis que les titres des articles *Wikipédia* sont souvent des phrases [12].

Lu, a utilisé la cooccurrence des informations pour l'extraction de la traduction. Un premier obstacle, est que l'expansion de la requête de mot clés, introduit du bruit à l'acquisition de la traduction, le second réside dans la grande quantité de fragments utilisés comme ressource bilingue pour l'extraction de la traduction. Pour résoudre ces problèmes, Schönhofen a essayé *Wikipédia* comme une ressource précise et parallèle pour l'extraction de la traduction des OOVs. Il a utilisé une stratégie d'appariement globale pour le mappage des thèmes de la requête et de l'article, qui améliore partiellement la performance de l'extraction de la traduction [25].

Attia et al. (2010) ont également exploité les titres des articles *Wikipédia* en se basant sur les liens multilingues afin d'extraire de façon automatique les expressions poly-lexicales (*MWEs* : *MultiWord Expressions*). Les MWEs sont des interprétations idiosyncrasiques, et dont le sens exacte de la MWE ne peut pas être obtenu à partir de ses constituants. Ce lexique a une importance significative comme une ressource linguistique, car ces expressions ne peuvent être analysées littéralement (mot-à-mot). En effet, leur avantage a été prouvé dans plusieurs applications de TALN, telles que l'extraction de texte, l'analyse syntaxique, et la traduction automatique. L'idée derrière cette approche, est qu'elle considère le trait de la non-compositionnalité sémantique comme une indication puissante pour qu'une phrase soit une MWE. Elle repose sur la correspondance asymétrique (la relation de *plusieurs-à-un*) entre les titres des articles *Wikipédia* arabes et les titres dans *21* langues différentes. Ainsi, pour la détection de ces expressions, tous les titres composés de plus d'un mot sont considérés comme candidats, puis ceux qui n'ont pas obtenu une traduction littérale sont classés comme des MWEs [42].

CHAPITRE V

PROBLÉMATIQUE

Une des questions les plus connues dans la RIT, est la mise en correspondance de la requête et des documents écrits dans deux langues différentes. Pour permettre la recherche de documents à partir d'une requête dans une autre langue, une traduction est donc nécessaire. La traduction de la requête dans la langue des documents est l'approche la plus utilisée par les systèmes de recherche d'information, car elle est moins coûteuse que celle qui consiste à traduire les documents dans la langue de la requête [1][5]. C'est dans ce cadre que s'inscrit notre contribution, et concerne particulièrement une approche de traduction de requêtes pour la RIT *arabe-anglais*. La langue source de notre système de recherche (l'*arabe*) a attiré l'attention de la communauté du TALN durant la dernière décennie, en raison de son importance politique et des différences linguistiques qu'elle présente par rapport à d'autres langues. En effet, la richesse, la complexité morphologique, et le phénomène d'agglutination de la dite langue présentent des défis motivants pour les chercheurs qui ont proposé des approches de traduction variées, à base de règles, statistiques, guidées par l'exemple, ou des méthodes hybrides combinant les approches précédentes [46].

Contrairement à de nombreuses langues européennes, la structure de la langue *arabe* diffère de celle de l'*anglais*, par exemple, la structure de la phrase anglaise est SVO *(Sujet-Verbe-Objet),* pour la phrase arabe la structure par défaut est VSO (*Verbe-Sujet-Objet).* Par conséquent, la traduction en anglais de la phrase arabe «زار يوسف عبد الله» («*zAr ywsf Ebd All~h*», «*Youcef a visité Abdoulah*») est «*Yusuf visited Abdullah*». Dans une phrase nominale, l'ordre des mots par défaut est *Sujet-Prédicat,* mais çà peut

changer en raison de contraintes particulières. Pour les phrases verbales, les différents ordres de mots SVO *(Sujet-Verbe-Objet)*, VSO *(Verbe-Sujet-Objet)*, VOS *(Verbe-Objet-Sujet)* et OVS *(Objet-Verbe-Sujet)*, sont tous acceptables dans la langue arabe. Voici un exemple illustratif de quelques phrases nominales et verbales en arabe [47] :

- Phrases nominales :

  - un nom suivi d'un adjectif : «السماء صافية», «*AlsmA' SAfyp*», «*Le ciel est clair*».
  - un nom suivi d'un nom : «هذا مدرس جيد», «*h*A mdrs jyd*», «*Il s'agit d'un bon enseignant*».
  - un nom suivi d'une préposition : «الطفل في الحديقة», «*Altfl fy AlHdyqp*», «*L'enfant est dans le jardin*».
  - un adjectif suivi d'un nom : «سلام هي», «*slAm hy*», «*C'est la paix*».

  Pour les trois premières phrases le sujet précède le prédicat, mais dans la dernière c'est l'inverse.

- Phrases verbales :

  - un exemple de SVO : «الرسول حفظ القرآن», «*Alrswl HfZ Alqr|n*», «*Le prophète a appris le Coran*».
  - un exemple de SOV : «نحن إياك نشكر», «*nHn <yAk n$kr*», «*Nous vous remercions*».
  - un exemple de VOS : «حفظ القرآن الرسول», «*HfZ Alqr|n Alrswl*», «*Le prophète a appris le Coran*».
  - un exemple de VSO : «حفظ الرسول القرآن», «*HfZ Alrswl Alqr|n*», «*Le prophète a appris le Coran*».
  - un exemple d'OVS : «القرآن حفظ الرسول», «*Alqr|n HfZ Alrswl*», «*Le prophète a appris le Coran*».

- un exemple d'OSV : «إياك نحن نشكر», «*<yAk nHn n$kr*», «*Nous vous remercions*».

On peut remarquer que dans la langue arabe, le mot a un ordre relativement libre, des ordres différents de mots peuvent correspondre à la même phrase en anglais ou en français.

Pour l'arabe moderne standard (*MSA : Modern Standard Arabic*), la structure de la phrase est moins complexe que dans l'arabe classique, car la MSA n'accepte pas les structures de phrases OSV et SOV comme ordres de mots [47].

De plus, la langue arabe présente d'autres difficultés pour les applications de TALN, la question la plus relevée est celle de l'ambiguïté morphologique qui est le principal défi pour le traitement automatique de l'arabe. Cette ambiguïté se manifeste quand l'analyseur morphologique associe plusieurs valeurs pour certains attributs d'une unité lexicale. En effet, un mot non voyellé peut conduire à de nombreuses solutions morphologiques. Par exemple, le mot «وقف» («*wkf*»), en dehors du contexte, il peut avoir des interprétations différentes, comme «وَقَفَ» («*waqafa*», «*Il s'est levé*»), «وَقْفٌ» («*waqofn*», «*cession*»), ou «وَقِفْ» («*waqifo*», «*Et lève-toi*»), ce dernier mot est une concaténation de la conjonction «و», («*w*», «*Et*») et du verbe «قِفْ» («*qifo* »,«*Se lever*») conjugué à l'impératif [46][48].

D'autres complexités de la langue arabe peuvent avoir lieu, telles que [46] :

- Il n'y a pas de lettres majuscules en arabe, comme c'est le cas pour l'anglais et le français, où les noms propres par exemple sont identifiés par le fait qu'ils débutent par une lettre majuscule.
- Les noms arabes doivent être soit masculins ou féminins. Généralement, les noms féminins sont dérivés des noms masculins, par exemple, pour le mot

masculin «طبيب» («*Tbyb*», «*Medecin*»), le nom féminin correspondant est «طبيبة» («*Tbybp*», «*Medecin*»), la lettre « ـة: *p*» est rajoutée à la fin du mot masculin. Néanmoins, pour certains cas la règle précédente n'est pas appliquée, comme par exemple pour les mots masculins «ولد» («*wld*», «*Garçon*»), «إمرأة» («*<mr>p*», «*Femme*»), «ديك» («*dyk*», «*Coq*»), les noms féminins sont respectivement «بنت» («*bnt*», «*Fille*»), «رجل» («*rjl*», «*Homme*»), «دجاجة» («*djAjp*», «*Poulet*»).

- L'arabe est une langue à sujet nul «*pro-drop*», c'est-à-dire que le sujet peut être omis, par exemple dans la phrase suivante : «يكتب الدرس» («*yktb Aldrs*», «*Il écrit la leçon*»).
- L'arabe est une langue agglutinante, il y a des mots qui tiennent le sens d'une phrase complète. Le mot «سنلعب» («*snlEb*») signifie en français «*Nous allons jouer*».
- L'arabe n'a pas d'auxiliaires être et avoir, par exemple, «الباب مفتوح» («*AlbAb mftwH*») et «لها كتاب» («*lhA ktAb*»), signifient respectivement «*La porte est ouverte*», et «*Elle a un livre*».
- Le système de nombres en arabe comprend la forme duale, deux suffixes («ان : *An*», «ين : *yn*») sont rajoutés au singulier (*stem*), selon qu'il s'agit d'un cas nominatif ou accusatif et génitif. Les formes duales des mots (féminin et masculin) au singulier «مهندسة» («*mhndsp*», «*Ingénieure*») et «مهندس» («*mhnds*», «*Ingénieur*»), sont pour le nominatif «مهندستان» («*mhndstAn*», «*Deux ingénieures*»), «مهندسان» («*mhndsAn*», «*Deux ingénieurs*»), pour l'accusatif et le génétif sont «مهندستين» («*mhndstyn*», «*Deux ingénieures*»), «مهندسين» («*mhndsyn*», «*Deux ingénieurs*»).
- De même pour le pluriel masculin et féminin, les suffixes («ين : *yn*», «ون : wn») et («اتٌ: *AtN*», «ات: *AtK*») sont rajoutés à la forme au singulier. Soient les deux mots (masculin et féminin) au singulier «معلم» («*mElm*», «*Professeur*»), et «معلمة» («*mElmp*», «*Professeure*»). Le pluriel masculin est «معلمون» («*mElmwn*», «*Professeurs*») pour le nominatif et «معلمين» («*mElmyn*», «*Professeurs*») pour

l'accusatif et le génitif. Le pluriel féminin est «معلماتٌ» («*mElmAtN*», «*Professeures*») pour le nominatif et «معلماتِ» («*mElmAtK*», «*Professeures*») pour l'accusatif et le génitif.

- Certains mots arabes n'ont pas de règle fixe pour leur pluriel. C'est le phénomène du pluriel irrégulier, comme par exemple les mots «باب» («*bAb*», «*Une porte*»), et «قلم» («*qlm*», «*Un stylo*»), le pluriel est «أبواب» («*>bwAb*», «*Des portes*»), et «أقلام» («*>qlAm*», «*Des stylos*»).

À travers les difficultés exposées précédemment, on peut conclure que la langue arabe est un sujet de recherche très important dans le traitement automatique des langues (TAL). C'est pour cela qu'elle a attiré l'intérêt de la communauté de TAL en enregistrant plus de progrès durant ces dernières années. Néanmoins, comparativement à l'anglais et à d'autres langues européennes, cet apport reste insuffisant dû à la richesse et la complexité de cette langue. En effet, les ressources linguistiques appropriées pour le développement et l'évaluation des applications de TALN sont peu ou pas disponibles. On peut citer par exemple, les grands corpus annotés et les ressources lexicales comme les gazetiers (*gazeteers*), une collection de listes prédéfinies des entités typées (des noms classés par types d'entités nommées). Ce genre de ressources constitue la principale source d'informations pour les systèmes de reconnaissance des entités nommées, et les systèmes de traduction automatiques [51].

Ainsi, nous nous sommes intéressées dans notre travail de recherche à la dite langue, et particulièrement à l'arabe moderne (MSA). Notre participation consiste à examiner la question de la traduction des requêtes arabes vers l'anglais, en exploitant l'encyclopédie multilingue *Wikipédia* qui est devenue une ressource très utile pour la construction et l'enrichissement des ressources linguistiques. Notre approche est basée sur un processus de segmentation de la requête arabe en plusieurs unités lexicales, en prenant en considération les unités dont la taille est la plus élevée qui apparaissent au début ou à la fin de la requête.

L'idée dernière notre méthode de segmentation est que dans la langue arabe, l'ordre des mots est libre, les premiers mots de la phrase anglaise peuvent être les derniers mots dans la requête source. Gaillard et al. (2010) par contre, dans leur méthode de segmentation privilégient les plus grandes unités lexicales qui apparaissent au début de la requête, car ils se sont intéressés au français comme langue source, qui présente moins de complexité en termes de structure de la phrase que la langue arabe.

CHAPITRE VI

MÉTHODOLOGIE

Dans le présent chapitre, nous allons décrire de façon détaillée l'approche que nous avons adopté et proposé comme solution à la problématique étudiée. Notre méthodologie est basée sur une proposition de Gaillard et al. (2010) qui ont exploité *Wikipédia* pour la traduction de requêtes du *français* vers *anglais*. Notre étude par contre, concerne la traduction de requêtes en utilisant *Wikipédia* pour la paire de langues *arabe-anglais,* où la langue source est connue par sa richesse et ses problèmes d'ambiguïté syntaxiques et morphologiques. En outre, nous nous sommes intéressées dans notre travail à l'étude de l'arabe moderne (MSA).

6.1 Extraction des titres bilingues

Pour la traduction des requêtes, les titres bilingues arabe-anglais, arabe-français et français-anglais des articles *Wikipédia* ont été extraits en se basant sur leurs liens inter-langues. Chaque paire de titres est enregistrée séparément dans deux fichiers textes (alignés par phrase), soient respectivement les fichiers : «*arTitles.txt*», «*enTitles.txt*», «*ariTitles.txt*», «*frTitles.txt*», «*friTitles.txt*», «*eniTitles.txt*».

Les articles *Wikipédia* ont été téléchargées des sites http://dumps.wikimedia.org/arwiki/latest/ et http://dumps.wikimedia.org/frwiki/latest/ :

- Archive de l'arabe (*arwiki-latestpages-articles.xml*) du 10 Mars 2013.
- Archive du français (*frwiki-latestpages-articles.xml*) du 09 Mars 2013.

Les tailles des archives *Wikipédia* téléchargées, ainsi que celles des six (6) fichiers contenant les titres des articles extraits sont représentés par le Tableau 6.1. Ce dernier montre que la taille des fichiers de l'archive arabe est faible comparativement à celle du français, et que la taille des lexiques bilingues arabe-anglais extraits est supérieure à celle des lexiques arabe-français. C'est pour cela que nous avons choisi l'anglais comme langue cible pour une traduction directe, et le français (la langue pivot) pour une traduction transitive. En outre, en plus de l'arabe et le français, nous avons étendu l'extraction des titres pour l'anglais, l'espagnol, l'allemand et le chinois, afin de pouvoir utiliser plus d'une langue pivot, et par conséquent, une traduction par triangulation sera possible.

Tableau 6.1 Tailles des archives et des titres extraits de *Wikipédia*.

Fichiers	Taille en Ko/ Mo
arwiki-latestpages-articles.xml	1.42 Go
frwiki-latestpages-articles.xml	10.1Go
arTitles.txt	4.44 Mo
enTitles.txt	2.98 Mo
ariTitles.txt	110 Ko
frTitles.txt	80 Ko
friTitles.txt	20.2 Mo
eniTitles.txt	19.3 Mo

Pour l'obtention d'un alignement par phrase adéquat, nous avons développé un programme en *Perl* qui supprime les lignes vides dans les fichiers textes résultants. Cette opération est effectuée une seule fois avant le lancement de la traduction des requêtes.

6.2 Segmentation de la requête

Pour la segmentation de la requête, nous avons proposé une nouvelle méthode qui tient compte de la complexité morphologique de la langue arabe. Elle est générique, et peut donc être appliquée à d'autres langues sans aucune modification, comme par exemple pour l'anglais, le français, l'espagnol et l'allemand. Néanmoins, pour le chinois une légère adaptation est nécessaire, car il n'y a pas d'utilisation de l'espace entant que séparateur entre deux mots ou groupe de mots.

Notre méthode (Figure 6.1), consiste à décomposer une requête en plusieurs unités lexicales (ou *Syntagmes*). Pour une requête de n mots, on peut générer 2^{n-1} segmentations possibles qui entraine une complexité exponentielle de notre application quand la valeur de n croit. À cet effet, dans le but de faciliter le traitement des requêtes arabes, notre méthode effectue la décomposition en faisant correspondre les différentes segmentations à une séquence de nombres, qui représentent les tailles des syntagmes dans l'ordre de leur apparition dans la requête.

De plus, afin de réduire davantage le temps de segmentation des requêtes longues, nous avons utilisé deux seuils, *sQ* (la taille de la requête maximale permise) et *sUlex* (la taille maximale des unités lexicales à segmenter). Pour accélérer le temps de traitement des requêtes, les seuils *sQ* et *sUlex* sont fixés par défaut à *20* et *10* respectivement, par rapport à la capacité d'un ordinateur de *4 Go* de *RAM*, *300 Go* de disque dur, et avec un processeur *Dual-Core 2.20 Ghz*. Cependant, ils peuvent être modifiés par l'utilisateur en fonction de la configuration matérielle de sa machine. Ainsi, deux formules de segmentation sont appliquées en fonction de la taille de la requête en nombre de mots (soit n) :

1- Requête courte : si $n < = sQ$ et $n <= sUlex$, alors la segementation est effectuée selon la formule 2^{n-1} (qui est le nombre de segmentations possibles générées).

2- Requête longue : si $n <= sQ$ et $n > sUlex$, alors la requête est décomposée en petites requêtes de tailles $sUlex$, dont une est de taille $n \equiv sUlex$, et la segmentation s'applique selon la formule $(n/sUlex)*2^{sUlex-1} + 2^{(n \equiv sUlex)-1}$ (les segmentations possibles) au lieu de 2^{n-1}. Le symbole « ≡ » désigne l'opérateur *modulo*.

Comparativement à la première formule, la deuxième formule nous a permis de réduire considérablement le temps de traitement des requêtes longues, car la segmentation est effectuée uniquement pour la taille s*Ulex* au lieu de *n*. De plus, et comme on manipule des nombres au lieu des chaines de caractères pour segmenter la requête, l'opération de segmentation est effectuée une fois pour la taille $n \equiv sUlex$, et une fois pour la taille s*Ulex* au lieu de *n/sUlex* fois.

Dans notre méthode, les segmentations obtenues sont triées par ordre décroissant de tailles des unités lexicales, et nous traitons d'abord celles qui apparaissent au début de la requête et puis celles qui apparaissent à la fin, car pour la RIT arabe-anglais, les premiers ou les derniers mots de la requête peuvent être les premiers mots dans la requête anglaise, en raison des caractéristiques de la langue arabe où les mots ont un ordre libre contrairement à l'anglais et au français. Cependant, l'approche de Gaillard et al. (2010) applique une décomposition de la requête en plusieurs syntagmes et privilégie seulement les unités lexicales qui apparaissent au début de la requête, et celles dont la taille est la plus élevée, étant donné que leur étude porte sur la paire de langue français-anglais.

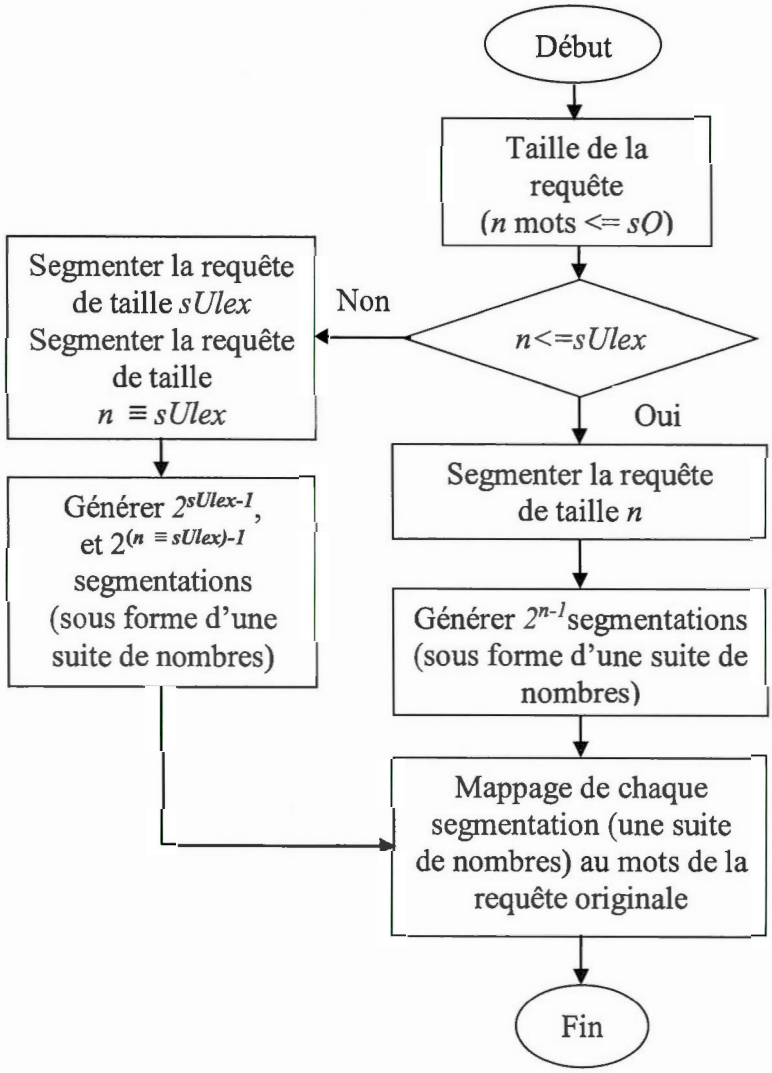


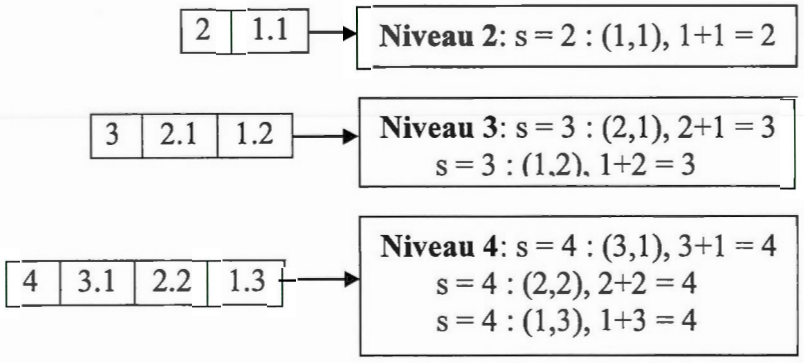
Figure 6.1 Méthode de segmentation de la requête.

Dans les deux exemples suivants, nous allons décrire de façon détaillée le principe sur lequel est basé notre méthode de segmentation.

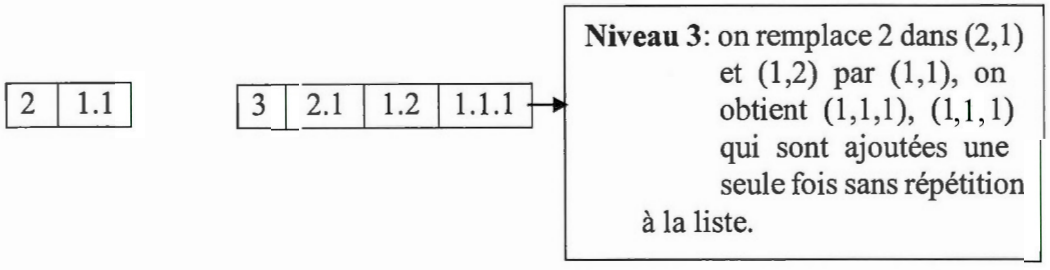
Exemple 1 : Soit la requête « *A B C D* » de *4* mots, *sQ = 10*, *sUlex = 4*, le programme de segmentation effectue les étapes suivantes :

- La taille de la requête est < = sQ et < = *sUlex*.
- Alors, segmenter la requête avec la formule 2^{n-1} segmentations possibles (*n =4)*.
- Les segmentations générées sont sous forme d'une suite de nombres qui représentent les tailles des unités lexicales dans l'ordre de leur apparition dans la requête.

  - ***Étape 1***: Le programme commence d'abord par construire pour chaque niveau, les différentes combinaisons des paires de nombres dont la somme est égale à *s* , où *s* varie de *2* à *n*. Pour cet exemple: *s* varie de *2* à *4*. Un niveau représente une unité lexicale de taille *s*, et toutes les paires de nombres générées sont les segmentations possibles de chaque niveau en deux unités dont la somme de leurs tailles est égale à *s*.



  - ***Étape 2*** : Pour chaque niveau *k* allant de *s+1* à *n* et pour chaque paire *(i,j)* tel que *i+j=k*, on remplace *i* puis *j* par toutes les combinaisons correspondantes aux niveaux précedents *i* et *j*.



4	3.1	2.2	1.3	2.1.1	1.2.1	1.1.1.1	1.1.2

Niveau 4: on remplace 3 dans (3,1) par (2,1), (1,2) et (1,1,1), on obtient : (2,1,1),(1,2,1), (1.1.1.1).

On remplace 2 dans (2,2) par (1,1), on obtient (1,1,2), (2,1,1). Les combinaisons obtenues sont rajoutées à la liste sans répétition.

- ***Étape 3*** : Les éléments du dernier niveau sont triés par ordre décroissant de tailles des unités lexicales.

4	3.1	2.2	1.3	2.1.1	1.2.1	1.1.2	1.1.1.1

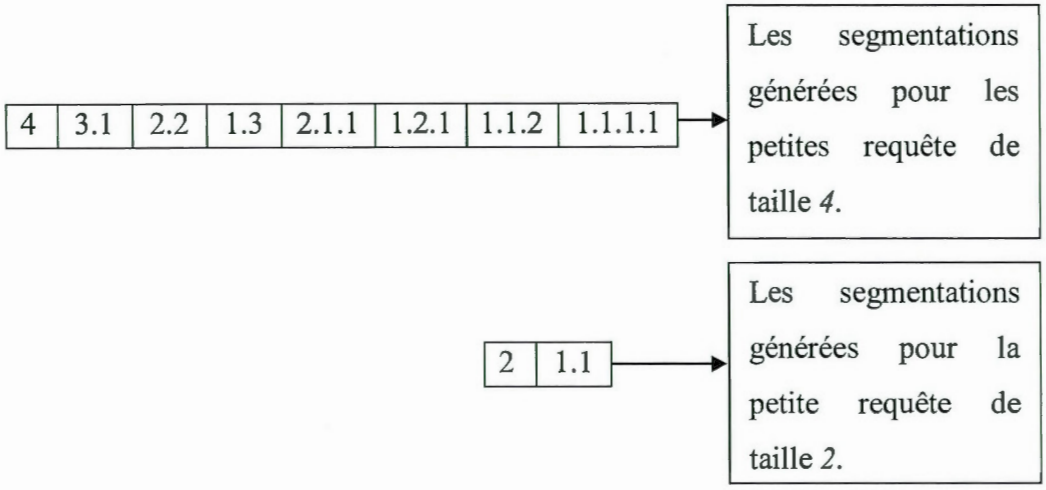
- ***Étape 4*** : cette dernière étape consiste à faire le mappage des différentes segmentations, en d'autres termes, remplacer les nombres par les mots qui leur correspondent dans la requête originale.

ABCD	ABC.D	AB.CD	A.BCD	AB.C.D	A.BC.D	A.B.CD	A.B.C.D

Exemple 2 : Soit la requête « *A B C D E F G H I J* » de *10* mots , *sQ = 10*, *sUlex = 4*.

- La taille de la requête est < = sQ et > *sUlex*.
- Alors, la requête est décomposée en deux (*2 = 10/sUlex*) petites requêtes de tailles *sUlex (4)*, et une requête de taille deux (*2 = 10 ≡ 4*).
- La segmentation sera effectuée une seule fois pour les petites requêtes de taille 4 et pour la requête de taille 2.

- ***Étape 1, Étape 2, Étape 3*** : Les même traitements que dans le premier exemple (ces étapes sont effectuées une seule fois).



- ***Étape 4*** : La première liste des segmentations de l'étape précedente pour la taille *4,* sont utilisées pour le mappage des quatres premiers mots, et des quatres mots suivants de la requête. La deuxième, celle qui correspond à la taille *2,* est uitlisée pour le mappage des deux derniers mots de la requête.

ABCD	ABC.D	AB.CD	A.BCD	AB.C.D	A.BC.D	A.B.CD	A.B.C.D

EFGH	EFG.H	EF.GH	E.FGH	EF.G.H	E.FG.H	E.F.GH	E.F.G.H

IJ	I.J

6.3 Prétraitement de la requête

Durant le processus de traduction de la requête, un stemming léger peut être effectué et qui consiste à tokeniser chaque mot de la requête en supprimant les préfixes suivants : « الـ », « بالـ », « والـ », « فالـ », « كالـ ». En outre, la conjonction «و » est supprimée quand elle n'est pas attachée directement aux mots de la requête.

Le stemming léger a été intégré au processus de traduction, contrairement à la boite à outils MADA+TOKAN qui a été utilisée pour l'analyse morphologique de toutes les requêtes de notre collection de données. Pour l'outil TOKAN, nous avons choisi le

schéma de tokenisation « *w+f+l+k+b+s+Al+REST+P* » qui, en se basant des résultats d'analyse retournés par MADA, il sépare les conjonctions (*w* :و, *f* :ف), les prépositions (*b* :بـ, *k* :كـ, *l* :لـ), les particules verbales (*s* :سـ), l'article défini (*Al* :الـ), les enclitiques pronominaux, ajoute le *POS-tag* de base à la forme du mot, et spécifie que les diacritiques sont générés. Un exemple plus détaillé est présenté dans le chapitre suivant. Notre expérience avec ces outils, nous a permis d'étudier l'effet d'un tel prétraitement sur les performances de notre système de recherche d'information translinguistique.

Exemple : Voici ci-dessous, deux exemples de requêtes arabes de notre collection prétraitées en utilisant le stemming léger et les outils MADA+TOKAN.

Première requête arabe (sans prétraitement)

ما هي حوسبة المستخدم النهائي والذي يقوم بذلك

*mA hy Hwsbp Almstxdm AlnhA}y wAl*y yqwm blk*
(translittération arabe de Buckwalter)
Qu'est-ce que l'informatisation de l'utilisateur final et qui le fait
(traduction en français)
What is The End User Computing and Who's Doing It
(traduction en anglais)

Première requête arabe prétraitée (stemming léger / MADA+TOKAN)

ما + هي + حوسبة + الـ + مستخدم + الـ + نهائي + و+ الذي + يقوم + بذلك

Deuxième requête arabe (sans prétraitement)

بدائل لبوست سكريبت

bdA}l lbwst skrybt (translittération arabe de Buckwalter)
Alternatives au Postscript (traduction en français)
Alternatives to Postscript (traduction en anglais)
Deuxième requête arabe prétraitée (stemming léger)

بدائل + لبوست + سكريبت

Deuxième requête arabe prétraitée (*MADA+TOKAN*)

بدائل + لـ + بوست + سكريبت

On peut remarquer que la première requête a obtenu le même résultat avec les deux façons de prétraitement. Néanmoins, pour la deuxième le résultat est différent, et le nombre de tokens fournis par MADA+TOKAN est supérieur à celui obtenu avec le stemming léger, car conformément au schéma de tokenization, TOKAN sépare ce type de préposition «لـ : *l*» qui ne peut pas être traité avec le stemming léger.

6.4 Méthode de traduction en exploitant *Wikipédia*

La traduction des requêtes arabes vers l'anglais est effectuée en faisant une recherche dans les lexiques bilingues extraits. Le lexique *arabe-anglais* est utilisé pour la traduction directe, et ceux de l'*arabe-français* et *français-anglais* sont exploités pour une traduction transitive. Les titres de la langue source (l'*arabe*), sont triés par ordre alphabétique afin d'accélérer l'opération de recherche. Les étapes qui suivent vont décrire de façon plus détaillée le processus de traduction de la requête illustré par la Figure 6.2 :

- Vérifier si la taille de requête ne dépasse pas le seuil *sQ* ;
- Si oui, elle est rejetée (pas de traduction), à cause du temps de son traitement supposé être long par rapport aux caractéristiques matérielles de l'ordinateur. Néanmoins, si la valeur du seuil *sQ* est bien choisie, toutes les requêtes courtes ou longues seront traitées dans un temps raisonnable ;
- Sinon, la traduction de la requête entière (non prétraitée) est recherchée dans le lexique bilingue *arabe-anglais* ;
- Si aucune traduction n'a été trouvée, on recherche la traduction de la requête entière (prétraitée) dans le même lexique que précédemment;

- Si l'étape préceédente echoue, et si $n<=sUlex$ alors on segmente selon la formule 2^{n-1}, et puis on recherche les traductions pour chaque segmentation. Si $n>sUlex$ alors la requête est décomposée en $n/sUlex$ petites requêtes, chacune de tailles $sUlex$ et une autre petite requête de taille $n \equiv sUlex$;
- La traduction est recherchée pour chaque petite requête, et sa segmentation ne sera effectuée que si sa traduction est vide ;
- On recherche la traduction de chaque segmentation (non prétraitée, puis prétraitée) ;
- Si aucune traduction n'est obtenue, une traduction transitive est lancée ;
- Le meilleur résultat de traduction choisis, est celui dont le nombre de ses unités lexicales traduites est le plus élevé.

Exemple : Voici ci-dessous deux requêtes en arabe de la collection Trec-2002- AP88-90, suivie chacune d'une translitération *Buckwalter*, et des traductions en français et en anglais correspondantes. On donnera également leurs traductions avec notre méthode de traduction.

065 <الرقم>
<المجال> العلوم والتكنولوجيا
<الموضوع> نظم استرجاع المعلومات
<الوصف> الوثيقة سوف تحدد نوع من نظام استرجاع المعلومات

065 <Alrqm>
AlElwm w Altknwlwjyā <AlmjAl>
nZm {strjAE AlmElwmAt <AlmwDwE>
Alwvyqp swf tHdd nwE mn nZAm {strjAE AlmElwmAt <AlwSf>
<num> Numéro : 065
<dom> Domaine : Science et technologie
<titre> Topic : Les systèmes de recherche d'information
<desc> Description : Document permettra d'identifier un type de système de recherche d'information.

<num> Number: 065

<dom> Domain: Science and Technology

<title> Topic: Information Retrieval Systems

<desc> Description:

Document will identify a type of information retrieval system.

<الرقم> 132

<المجال> عسكري

<الموضوع> "الشبح" مركبة جوية

<الوصف> سوف توفر الوثيقة بيانات التكاليف، التقنية و/أو الأداء على مشاريع "الشبح" المركبة الجوية للولايات المتحدة

132 <Alrqm>

Eskry <AlmjAl>

Al$bH mrkbp jwyp <AlmwDwE>

swf twfr Alwvyqp byAnAt AltkAlyf, Altqnyp w/>w <Alwsf>

Al>dA' ElY m$AryE Al$bH Almrkbp Aljwyp llwlAyAt AlmtHdp

<num> Numéro : 132

<dom> Domaine : Militaire

<titre> Topic : Avions «Furtifs»

<desc> Description : Le document fournira les données du coût, techniques et / ou de performance sur les projets d'avions "furtifs" des États-Unis.

<num> Number: 132

<dom> Domain: Military

<title> Topic: "Stealth" Aircraft

<desc> Description :

Document will provide cost, technical, and/or performance data on U.S. "stealth" aircraft projects.

À travers les deux requêtes arabes précédentes, *N°: 065* «نظم استرجاع المعلومات» et *N°: 132* «مركبة جوية "الشبح"» nous allons présenter clairement les différentes étapes de notre méthode de traduction.

Avec notre approche, la traduction de la première requête est «*Information Retrieval*», aucune segmentation n'est effectuée, car selon notre méthode, on commence d'abord par une recherche de la requête entière dans le lexique arabe extrait. Si la requête existe, le titre anglais correspondant est retourné. Pour la deuxième requête, par contre, qui ne figure pas entièrement dans le lexique, on procède à sa segmentation de la façon suivante :

- Suppression des signes de ponctuation de la requête.
- Segmenter la requête dont le nombre de termes est de trois, le nombre de segmentation possibles est égale à *4*, générées par application de la formule 2^{n-1} (car $n<=10$).

Les segmentations possibles = {الشبح مركبة جوية}, {الشبح مركبة, جوية}, {الشبح, مركبة جوية},

{مركبة, جوية, الشبح}

- Rechercher les traductions pour chaque segmentation, pour la première segmentation (un syntagme de 3 mots), aucune traduction n'est retournée. De même pour la deuxième (composée de deux syntagmes de 2 et 1 mot(s) respectivement), aucune traduction n'est trouvée pour les deux. Pour la troisième segmentation, une traduction («*Aircraftest*») est retournée pour l'unité lexicale "مركبة جوية", et pour "الشبح" aucune traduction n'est obtenue. La traduction de la dernière segmentation de trois unités lexicales (de un mot chacune) est vide.
- Dans notre approche, les segmentations sont triées par ordre décroissant des tailles de leurs unités lexicales afin de privilégier la traduction de celles dont la

taille des unités lexicales est la plus élevée (qu'elles soient au début ou à la fin de la requête). Parmi les traductions des différentes segmentations possibles, la meilleure traduction choisie est celle dont la taille des syntagmes traduits est la plus élevée, ou celle qui a le plus de syntagmes traduits.

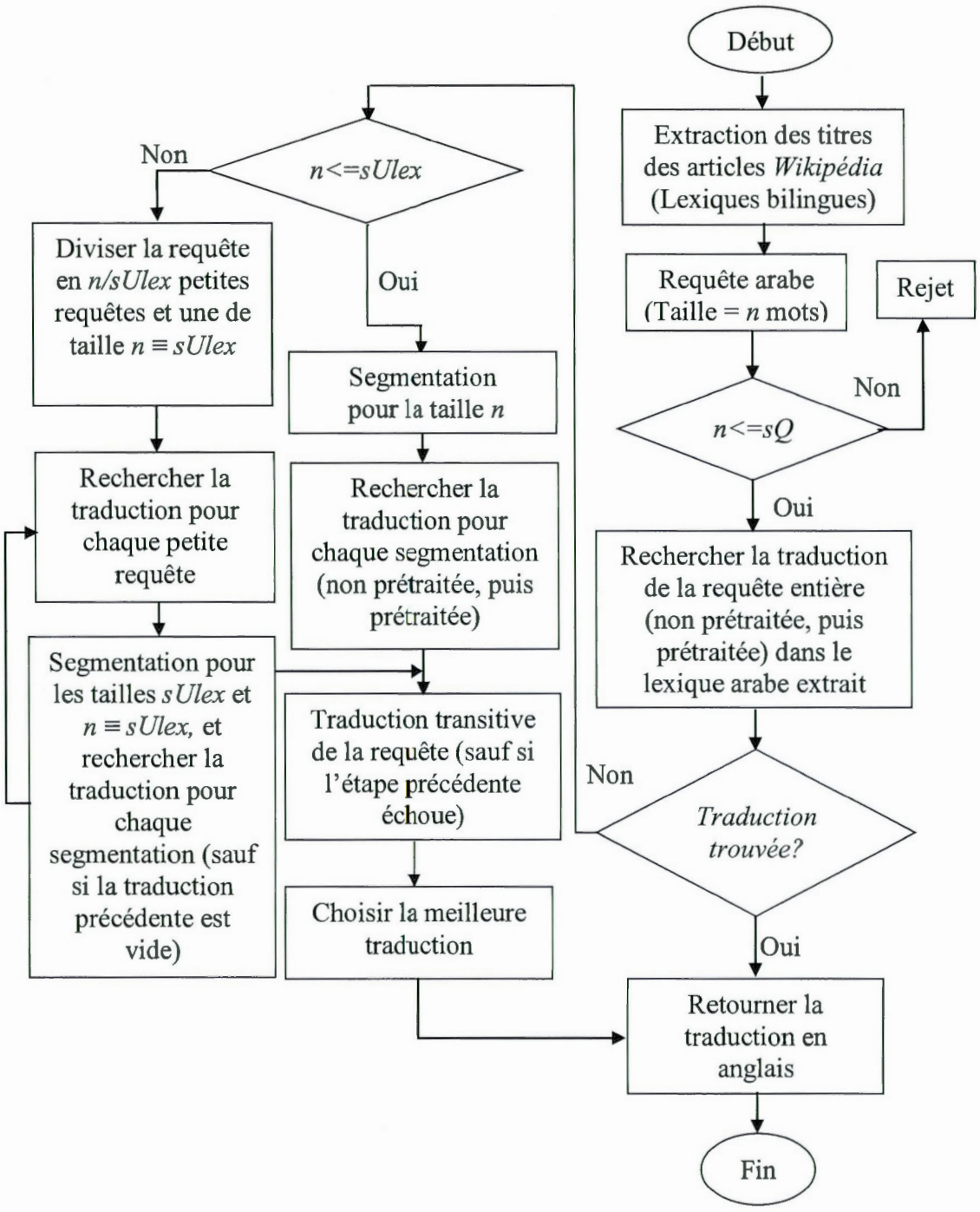


Figure 6.2 Méthode de traduction en exploitant *Wikipédia*.

6.5 Implémentation

Pour l'implémentation de notre approche de traduction exploitant *Wikipédia*, nous avons développé un programme *Java* sous *NetBeans,* qui réalise les fonctionnalités principales suivantes :

- Extraction des titres bilingues à partir des archives *Wikipédia* : Peut être effectuée pour les langues *arabe, anglais, français, espagnol, allemand,* et *chinois*.
- Recherche de traductions dans les lexiques bilingues extraits, d'une langue source vers une langue cible (qui peuvent être parmi les six langues citées précédemment).
- Segmentation de la requête : Cette méthode ne se limite pas uniquement à la langue *arabe,* elle peut être appliquée également à plusieurs autres langues.

Pour chaque paire de langue, le lexique bilingue extrait est enregistré dans des fichiers textes séparés, ces données sont transférées durant le processus de traduction vers des collections afin d'accélérer l'opération de recherche. Les structures de données utilisées pour le développement de notre application sont les suivantes :

- Le lexique de la langue source est transféré dans un *TreeMap* :

  TreeMap <Titre_Source, Position>

  Titre_Source : de type *String* qui représente le titre en langue source (trié).

  Position : de type *Integer* qui représente la position du titre (numéro de la ligne) en langue source dans le fichier texte.

- Le lexique de la langue cible est transféré dans un vecteur < *Titre_Cible* > :

  Vector <Titre_cible>

Titre_Cible : de type *String* qui représente le titre en langue cible (non trié).

De même, les différentes segmentations générées par le programme sont stockées dans un vecteur dont la structure est la suivante :

Vector<Segmentation>

Segmentation : de type *String*, est une suite de nombres séparés par un point qui représente la taille des différentes unités lexicales composant la requête.

De plus, pour les besoins de la présente étude nous avons développé un programme qui intègre les opérations d'indexation et de recherche du moteur *Lucene* (*version 4.1.0*) selon le schéma de pondération *Par défaut*. Les APIs suivantes ont été utilisées :

- *Wikixmlj-r43.jar* : Est une interface qui offre un accès facile aux archives *Wikipédia* (au format XML), elle a été téléchargée du site http://code.google.com/p/wikixmlj/downloads/list .
- *Lucene-core-4.1.0.jar* : C'est la bibliothèque de *Lucene* qui permet d'ajouter des fonctionnalités de recherche plein-texte à nos application.
- *Lucene-demo- 4.1.0.jar* : Est un package qui offre de simples exemples de codes pour montrer les caractéristiques de *Lucene*.
- *Lucene-analyzers-common-4.1.0.jar* : Regroupe les packages des analyseurs qui peuvent être utilisés pour l'indexation du contenu dans différentes langues et domaines.
- *Lucene-queryparser-4.1.0.jar* : Est un package qui offre l'ensemble des analyseurs de la requête.

Les APIs de *Lucene* ont été téléchargées du site http://lucene.apache.org/. Le code de quelques programmes de notre application sont fournis en Annexes.

CHAPITRE VII

ÉVALUATIONS ET RÉSULTATS

Pour les besoins de l'évaluation de notre méthode de traduction et le développement de notre système de recherche translingue, nous avons utilisé la boite à outils MADA+TOKAN pour l'analyse morphologique et la tokenization des requêtes arabes. Le moteur de recherche *Lucene,* a été choisi pour la recherche et l'indexation de notre collection de données. En outre, les mesures de précision et de rappel des résultats de recherche retournés par *Lucene* sont obtenues à l'aide du programme Trec_eval de TREC[22]. Dans les paragraphes qui suivent, nous allons décrire de façon plus claire les ressources utilisées et nous allons présenter également les résultats d'évaluation ainsi que leur interprétations.

7.1 Ressources utilisées

7.1.1 Moteur de recherche *Lucene*

Lucene[23], est une bibliothèque puissante de recherche d'information écrit en *Java.* En raison de sa simplicité et sa capacité de traiter un grand volume de données, il est devenu la bibliothèque ouverte, la plus largement utilisée en recherche d'information [28]. Il est aussi disponible dans d'autres langages de programmation (*C/C++, C #, Ruby, Perl, Python* et *PHP*). *Lucene*, a été créé par Doug Cutting, il a joint la fondation

[22] http://trec.nist.gov/trec_eval/

[23] http://lucene.apache.org/

des logiciels *Apache* («*Apache Software Foundation*») en *Septembre 2001* et est devenu son propre projet *Apache* de haut niveau en *Février 2005* [28].

En effet, cette bibliothèque permet d'ajouter des fonctionnalités de recherche à nos applications, qui permettent d'indexer et de rechercher toute donnée à partir de laquelle on peut extraire du texte. La source des données peut être dans différentes langues, et sous différents formats (pages web, documents conservés dans les systèmes de fichiers, fichiers textes, documents *Word, XML, HTML, PDF*, ...etc.). La recherche d'une requête est le processus qui consiste à consulter l'index, et rechercher les documents correspondant à la requête triés dans l'ordre de tri demandé. L'approche de recherche adoptée par *Lucene* combine le modèle vectoriel et le modèle booléen (voir section 1.3.1 et 1.3.2), et retourne la liste des documents résultants à l'utilisateur pour exploitation [28].

Grâce à sa puissance et à sa rapidité dues à l'indexation, *Lucene* est utilisé principalement par les bio-informaticiens qui manipulent des banques de séquences biologiques. Par ailleurs, de nombreux groupes tels que *Wikipédia* ou *Eclipse* y recourent également pour le moteur de recherche de leur site *Web* [54].

7.1.1.1 Les classes de recherche et d'indexation

Les classes de base de l'*API Lucene*, pour l'indexation et la recherche sont les suivantes [28]:

- Les classes d'indextaion : Pour effectuer la procédure d'indexation la plus simple, on a besoin des classes, *IndexWriter, Directory, Analyzer, Document,* et *Field* :

  - *IndexWriter* : cette classe est l'élément central du processus d'indexation. Elle permet de créer un nouvel index (ou ouvrir un index existant), ajouter, supprimer ou mettre à jour les documents dans l'index.

- *Directory* : elle représente l'emplacement de l'index. C'est une classe abstraite, qui permet à ses sous-classes de stocker l'index à l'endroit désiré. *Lucene* comprend un certain nombre d'implémentations intéressantes de la classe *Directory*. Par exemple, l'implémentation «*FSDirectory.open*» permet de stocker les fichiers dans un répertoire sur le système de fichiers.

- *Analyzer* : est une classe abstraite livrée avec plusieurs implémentations. Avant l'indexation, le texte est passé à travers l'analyseur spécifié dans le constructeur *IndexWriter*. Par exemple, l'analyseur «*SimpleAnalyzer*» divise un texte en faisant la conversion des tokens en lettres minuscules, «*StopAnalyzer*» permet en plus, la suppression des mots vides (*le, la, les*,..etc.), et «*StandardAnalyzer*» en plus des traitements précedents, elle permet de créer des tokens basé sur une grammaire sophistiquée, qui reconnaît les adresses e-mail, les acronymes, les caractères alphanumériques, et autres.

- *Document* : cette classe est une collection de champs qui représentent le document, ou les métadonnées associées à ce document. *Lucene* traite le texte extrait des documents et l'ajoute comme une instance de la classe *Field*. Les métadonnées (par exemple : auteur, titre, thème, et date de modification) sont indexés et stockés séparément comme des champs d'un document.

- *Field* : dans un index, chaque document contient un ou plusieurs champs nommés de cette classe. Chaque champ a un nom et une valeur, ainsi qu'un tas d'options qui contrôlent précisément comment *Lucene* peut indexer la valeur du champ.

• Les classes de recherche : De même que pour l'indexation, quelques classes sont nécessaires pour effectuer la recherche, *IndexSearcher, Term, Query,* et *TopDocs*.

- *IndexSearcher* : est une classe qui ouvre un index en lecture seule, elle nécessite une instance *Directory* (tenant l'index crée), puis elle offre quelques méthodes de recherche dont certaines sont implémentées dans la classe abstraite *Searcher*. La plus simple, prend comme paramètres l'objet *Query* et un nombre entier *topN*, et retourne un objet *TopDocs*. Les lignes de code suivantes montrent une utilisation typique de cette méthode :

```
Directory dir = FSDirectory.open(new File("/tmp/index"));
IndexSearcher searcher = new IndexSearcher(dir);
Query q = new TermQuery(new Term("contents", "lucene"));
TopDocs hits = searcher.search(q, 10);
searcher.close();
```

- *Term* : similaire à l'objet *Field*, il est constitué de deux éléments, le nom du champ et le mot (valeur du texte) de ce champ. Au cours de la recherche, on peut construire des objets *Term* et les utiliser avec *TermQuery*:

```
Query q = new TermQuery(new Term("contents", "lucene"));
TopDocs hits = searcher.search(q, 10);
```

Les deux lignes de code précédentes, indiquent à *Lucene* de trouver les *10* premiers documents qui contiennent le mot «*Lucene*» dans un champ nommé «*contents*», et de trier les documents par ordre décroissant de pertinence.

- *Query* : c'est la classe abstraite, livrée avec un certain nombre de sous-classes de requêtes concrètes (*TermQuery, BooleanQuery, PhraseQuery, PrefixQuery, PhrasePrefixQuery, TermRangeQuery, NumericRangeQuery, FilteredQuery, et SpanQuery*). *TermQuery,* est la requête *Lucene* la plus élémentaire. La méthode la plus intérressante de cette classe est *setBoost(float)*, qui permet d'indiquer à *Lucene* que certaines sous-requêtes doivent avoir une plus forte contribution au score de pertinence final que les autres sous-requêtes.

- *TermQuery* : comme mentionné ci-dessus, *TermQuery* est le type de requête le plus basique pris en charge par *Lucene*. Il est utilisé pour faire correspondre les documents qui contiennent des champs ayant des valeurs spécifiques.

- *TopDocs* : cette classe est un simple conteneur de pointeurs vers les *N* premiers documents des résultats de recherche, qui correspondent à une requête donnée.

7.1.1.2 La formule de similarité de *Lucene*

Au cours de la recherche, si un document correspond à la requête, alors un score lui est assigné. Ce score mesure le degré de similarité entre le document et la requête, un score élevé indique une forte correspondance entre eux. Le score est calculé pour chaque document «*d*» correspondant à chaque terme «*T*» dans une requête «*q*». La formule du score utilisée par *Lucene*, est la suivante [28]:

$$\sum_{t\ in\ q} \left(tf(t\ in\ d)x\ idf(t)^2\ \ boost(t.field\ in\ d)\ lengthNorm(t.field\ in\ d)\right) coord(q,d)\ queryNorm(q) \quad (1)$$

Les différents facteurs utilisés dans la formule (1) sont détaillés comme suit [28]:

- *tf(t in d)* : est la fréquence du terme *t* dans le document *d.*
- *idf(t)* : est la fréquence de document inverse du terme *t*. Les termes rares ont une *idf* élevée, contrairement aux termes trés courants.
- *boost(t.field in d)* : le *boost,* est utilisé pour stimuler statiquement certains champs et certains documents sur les autres.
- *lengthNorm(t.field in d)* : c'est la valeur de normalisation d'un champ, étant donné le nombre de termes dans le champ. Les champs plus courts (ayant peu de tokens) obtiendront le *boost* le plus élévé.
- *coord(q,d)* : c'est un facteur de coordinnation, basé sur le nombre des termes de la requête que contient le document.
- *queryNorm(q)* : est la valeur de normalisation de la requête *q*, compte tenu de la somme quadratique des poids de chacun des termes de la requête.

La plupart des facteurs de la formule ci-dessus, sont contrôlés et mis en œuvre comme une sous-classe de la classe abstraite *Similarity*. *DefaultSimilarity*, est l'implémentation utilisée sauf indication contraire [28].

Le code des programmes concernant l'implémentation des fonctionnalités d'indextation et de recherche du moteur *Lucene* est donné en Annexes.

7.1.2 Analyseur morphologique MADA et TOKAN

MADA et TOKAN sont des boites à outils polyvalentes, hautement personnalisées et gratuitement disponibles pour la langue *arabe.* Ils fournissent un outil excellent de prétraitement pour les principales applications de TALN (*NLP : Natural Language Preprocessing)*, telles que la traduction automatique (*MT* : *Machine Translation*), la reconnaissance automatique de la parole (*ASR : Automatic Speech Recognition*), et la reconnaissance des entités nommées (*NER : Named Entity Recognition*) [21].

MADA, est un utilitaire qui ajoute des informations lexicales et morphologiques, tout en levant l'ambiguïté en une seule opération. Cet analyseur est basé sur une approche

qui combine plusieurs tâches différentes en un seul coup, tokenisation, diacritisation, désambiguïsation morphologique complète, étiquetage de la position du discours (*POS Tagging*) et lemmatisation. Ceci, est un trait important qui distingue MADA des autres analyseurs qui séparent la tâche de tokenisation et de stemming du *POS Tagging*, ce qui peut conduire à plus d'érreurs. En effet, une grande partie des analyseurs disponibles ont tendance à cibler une application spécifique ou un *POS Tagging* qui n'est pas suffisamment général pour différentes applications. MADA+TOKAN, par contre, montrent que différentes représentations de la morphologie (*tokenisation, POS Tagging*) peuvent être effectués différemment sur la même tâche, ce qui permettra aux chercheurs d'explorer rapidement et facilement un grand espace d'étiquettes et annotations possibles [21]. C'est pour cela que cette boite à outils trouve son utilisation dans plusieurs travaux de recherches comme dans [42] [48] [49] [51] [55].

TOKAN est un tokeniseur général pour l'*arabe*, il permet de faire la segmentation d'un texte arabe désambiguïsé de MADA dans un ensemble large de schémas de tokenisation possibles (*D1, D2, TBold, TB, D3, EN* ... ect). Par exemple, pour le schéma «*w+f+b+k+l+s+Al+REST+ / + POS+P :+O :+DIAC*», TOKAN sépare les conjonctions (*w :* و *f :*ف), les prépositions (*b :* ب, *k :* ك, *l :* ل), les particules verbales (*s :* س), l'article défini (*Al :* ال), les enclitiques pronominaux, ajoute le *POS-tag* de base à la forme du mot, et spécifie que les diacritiques sont générés. Le Tableau 7.1, montre un exemple de segmentation de l'outil TOKAN selon les deux schémas *D1* et *D3* de la phrase arabe : «وسينهي الرئيس جولته بزيارة الى تركيا», qui signifie en français «*Le président va terminer sa tournée par une visite en Turquie*» [21].

On peut observer que les résultats de tokenisation de chaque mot de la phrase, peut différer conformément au schéma choisi. Le nombre de tokens obtenu avec le schéma *D3* est supérieur à celui de *D1*.

Tableau 7.1 Exemple de segmentation avec les shémas *D1* et *D3* de TOKAN.

Phrase en Arabe	وسينهي الرئيس جولته بزيارة الى تركيا
Translitération arabe de *Buckwalter*	wsynhy Alr}ys jwlth bzyArp AlY trkyA
Traduction en Anglais	The president will finish his tour with a visit to Turkey.
Schéma : D1 = (w+, f+), (و, ف)	و+ سينهي الرئيس جولته بزيارة الى تركيا w+ synhy Alr}ys jwlth bzyArp AlY trkyA
Schéma : D3 = (l+, k+, b+, s+, Al+), (لـ, كـ, بـ, سـ, الـ) et les enclitiques pronominaux	و+س+ينهي ال+رئيس جولة+ه ب+زيارة الى تركيا w+ s+ ynhy Al+ r}ys jwlp + h b+ zyArp AlY trkyA

Pour l'exemple de la phrase arabe précedente et par application du schéma *D1*, TOKAN divise le premier mot en deux (و, سينهي), car il contient la conjonction (*w:* و) , le reste des mots est inchangé, car aucune des conjonction (*w :* و *f :* ف) n'est attachée à l'un d'eux. Cependant, avec le schéma *D3* tous les mots sont tokenisés sauf les deux derniers. Le premier mot est divisé en trois mots (و, سـ, ينهي), la conjonction (*w:* و), le particule verbale (*s :* سـ), et le mot (ينهي). Le deuxième est divisé en deux (الـ, رئيس), l'article défini (الـ) et le mot (رئيس). De même pour le troisième (ه ,جولة), le mot (جولة) et l'enclitique pronominal (*h :* ه), et le quatrième (بـ, زيارة), la préposition (*b :* بـ) et le mot (زيارة). Au total, on a onze (11) tokens, au lieu de sept (7) obtenus avec le shéma *D1*.

Pour notre application, cette façon de faire, nous permettra de rechercher les traductions pour les différentes formes de mots, et donc augmenter les possibilités d'obtention des traductions dans les lexiques bilingues extraits.

Pour son exécution, MADA nécessite :

- *SVMTOOLS*: Est un générateur de tagueurs (étiqueteurs) séquentiels basé sur les SVMs, il a été téléchargé du site *http://www.lsi.upc.es/~nlp/SVMTool/*, et la ligne du code «*my $SVMTagger="/nlp/tools/SVMTool/SVMTool-*

1.2.2/bin/SVMTagger"» dans le programme «*MADA-SVMTOOLS.pl*» écrit en *Perl,* doit être remplacée par «*my $SVMTagger=" /SVMTool-1.2.2/bin/SVMTagger"*», où *SVMTool-1.2.2* est le nom du répertoire dans lequel il est installé.

- L'analyseur *BUCKWALTER* et *ARAGEN* (qui doivent être dans le même repertoire que l'analyseur MADA).

7.1.3 Évaluation avec *trec_eval*

Trec_eval[24] est un outil standard utilisé par la communauté TREC pour l'évaluation d'une exécution de recherche, étant donné un fichier des résultats de recherche et un ensemble de résultats jugés. La version de ce programme peut être exécutée sous un environnement *Unix* ou *Cygwin*. La ligne de commande qui permet de lancer l'exécution de ce programme avec les options les plus courantes est la suivante :

trec_eval [-q][-a][-M<num>] Qrels Results > Output

Où, l'option *–q* : signifie que les évaluations seront détaillées pour chaque requête, *-a* : résume les évaluations de toutes les requêtes, *-M<num>* : indique qu'au maximum *num* documents seront utilisés dans l'évaluation (le reste sera écarté), *Qrels*: c'est le nom du fichier contenant les jugements de pertinence, *Results* : est le nom du fichier contenant les résultats de recherche, et *Output* : est le nom du fichier dans lequel seront enregistrés les résultats retournés par *trec_eval* (ce parametre est optionnel).

La commande suivante, illustre les résultats des expérimentations effectuées lors de l'évaluation de notre méthode de traduction (voir chapitre VII) :

trec_eval –a –M1050 Qrels Results > Output

[24] http://trec.nist.gov/trec_eval/

Cette commande, indique que les résultats d'évaluation seront résumés pour toutes les requêtes, et au maximum 1050 documents seront considérés dans le calcul. Le Tableau 7.2, montre un exemple de quelques lignes du fichier contenant les documents jugés. La structure de ce fichier est composée de quatre (4) champs, la première colonne représente le numéro de la requête, la deuxième colonne est toujours à 0, la troisième colonne est l'identificateur du document, et enfin, la dernière colonne est la pertinence qui est égale à 1, si le document est pertinent pour la requête en question et 0 sinon.

Pour être conforme au format requis par *trec_eval*, la structure du fichier *Qrels* a été adaptée en utilisant un programme écrit en *Perl*.

Tableau 7.2 Quelques lignes du fichier *Qrels* (jugements de pertinence).

1	0	AP880212	0
1	0	AP880216	1
1	0	AP880217	0
1	0	AP880218	0
1	0	AP880219	0
1	0	AP880220	1
1	0	AP880221	0
1	0	AP880222	0
1	0	AP880223	0
1	0	AP880224	1

Les lignes listées dans le Tableau 7.3, représentent une petite partie du fichier résultat (*Results*) retourné par le moteur *Lucene* lors de la recherche de la requête numéro 1 de notre collection de test. Ce fichier, contient six (6) champs, la première colonne est le numéro de la requête, la deuxième colonne est toujours à 0, la troisième colonne contient l'identificateur du document, la quatrième colonne est le rang du document dans la liste, la cinquième colonne est le score du document. Et enfin, la dernière colonne est un champ à 0.

Un exemple des résultats d'évaluation retournés par le programme *trec_eval* conformément aux options choisies dans la ligne de commande, sont représentés par les premières lignes du Tableau 7.4. Les cinq premières lignes par exemple, sont respectivement, le nombre de requête traitées, le nombre de documents retrouvés, le nombre de documents pertinents dans la collection de données, le nombre de documents pertinents retrouvés, et la précision moyenne (*MAP*) correspondante.

D'autres mesures d'évaluation sont également fournies à la suite de ces résultats, comme les mesures *P5, P15, P20, P30, P100, P200, P500, P1000*, et *R5, R15, R20, R30, R100, R200, R500, R1000*, qui sont respectivement les précisions et le rappel quand *5, 10, 20, 30, 100, 200, 500*, ou *1000* documents sont retrouvés.

Tableau 7.3 Quelques lignes du fichier *Results* (résultats de recherche).

1	0	AP891103	1.	score=0,0309 0
1	0	AP890720	2.	score=0,0289 0
1	0	AP890323	3.	score=0,0289 0
1	0	AP900514	4.	score=0,0288 0
1	0	AP900621	5.	score=0,0278 0
1	0	AP890726	6.	score=0,0268 0
1	0	AP881007	7.	score=0,0255 0
1	0	AP890308	8.	score=0,0254 0
1	0	AP890417	9.	score=0,0254 0
1	0	AP880613	10.	score=0,0253 0

Tableau 7.4 Un exemple du fichier *Output* (résultats de *trec_eval*).

num_q	all	150
num_ret	all	147368
num_rel	all	17573
num_rel_ret	all	17121
map	all	0.1408

7.2 Évaluations

Pour l'évaluation de notre méthode de traduction, nous avons utilisé la collection de données gratuite de Trec-2002- AP88-90 (*1050* fichier textes, *242 918* documents, *728* Mo, *150* requêtes), un ensemble d'articles de nouvelles publiés par l'*Associated Press* en *1988-1990*. Néanmoins, les documents et les requêtes de cette collection sont fournis en *anglais,* et comme la langue source de notre système de recherche translingue est l'*arabe*, nous avons traduit manuellement toutes les requêtes anglaises vers l'*arabe* pour être considérées comme étant des traductions de références et utilisées dans le système monolingue pour l'*anglais*. De plus, afin d'étudier la RIT *arabe-anglais,* nous avons traduit l'ensemble des requêtes avec *Wikipédia*, *Google Translate* et *Mymemory*.

Lucene a été le moteur de recherche choisi dans le cadre de cette étude, pour effectuer les opérations de recherche et d'indexation de la collection de données selon le schéma de pondération *Par défaut*. Par ailleurs, les outils MADA+TOKAN ont été utilisés pour l'analyse morphologique et la segmentation des 150 requêtes arabes.

À cet effet, plusieurs expérimentations ont été effectuées pour l'évaluation des performances de notre méthode de traduction pour la RIT *arabe-anglais*. La mesure de la précision moyenne (*MAP*) du système monolingue a été calculée sur les résultats de recherche retournés par *Lucene* pour les 150 requêtes anglaises. Pour le système translingue, les MAPs ont été déterminées à partir des résultats de recherche fournis par *Lucene* pour les 150 requêtes traduites de façon directe de l'*anglais* vers l'*arabe*, et transitive de l'*arabe* vers le *français* et du *français* vers l'*anglais*. La traduction transitive n'a été effectuée que pour les requêtes qui n'ont pas obtenu des traductions directes. Tout d'abord, la requête entière (sans puis avec prétraitement) est traduite de façon transitive. Ensuite, en cas d'échec de l'étape précédente, si la requête est longue elle est juste divisée en petites requêtes, sinon elle est segmentée en 2^{n-1} segmentations,

et enfin une traduction transitive est lancée pour chaque petite requête ou pour chaque segmentation.

En outre, afin d'étudier l'effet du prétraitement sur les performances de notre système de recherche translingue, nous avons réalisé des évaluations pour la traduction directe selon quatre variantes, sans aucun prétraitement de requêtes, avec un prétraitement (stemming léger), une analyse morphologique avec MADA+TOKAN, et leur combinaison. Les mesures de présision moyennes des traductions réalisées avec *Google Translate* et *Mymemory* ont été aussi calculées, pour une étude comparative de la dite méthode avec de tels systèmes de traduction automatiques. Les résultats obtenus sont résumés par les Tableaux 7.5, et Tableau 7.6.

Les résultats du Tableau 7.5, montrent que les précisions moyennes obtenues avec les variantes de prétraitement de la requête (stemming léger, MADA+TOKAN) ou de leur combinaison, sont meilleures qu'avec les requêtes non prétraitées. Néanmoins, on peut observer que la précision moyenne du système monolingue avec la collection de données de Trec-2002- AP88-90 est faible. Et suite à nos recherches à propos de la dite collection, nous avons trouvé des résultats presque similaires obtenus avec la même collection de test dans les références suivantes :

- Le livre «*Charting a new course: Naturel Processing and Information Retrieval*» , à la page 75, les précisions sont de 13.91% et 23.71% pour les modèles *TF-IDF* et *BM25* respectivement [52].

- L'article «*Query Dependent Pseudo-Relevance Feedback based on Wikipédia*». Pour la même collection, et pour toutes les requêtes la précision est de 14.28% (avec le modèle *query-likelihood language*) [53].

De plus, on peut remarquer que pour la traduction transitive, les précisions moyennes obtenues à la suite de cette expérience sont presque similaires à ceux de la traduction directe (sans prétraitement), ceci s'explique par la faible taille du lexique bilingue *Arabe-Français* qui représente un taux de 2.42% par rapport à la taille du lexique *arabe-anglais* et 0.53% par rapport à celui du *français-anglais* (voir Tableau 6.1). Par conséquent, une précision meilleure peut être obtenue une fois ce lexique sera plus enrichi.

Par ailleurs, notre méthode de traduction en exploitant *Wikipédia* a obtenu une précision moyenne inférieure à celles à de *Google Translate* et *Mymemory* de 3.73%, mais juste de 1,64% dans le cas où un stemming léger est effectué. Ainsi, cet écart peut être réduit à nul en exploitant également le contenu des articles *Wikipédia,* au lieu se limiter qu'aux titres seulement.

Tableau 7.5 Comparaison des *MAPs* de la méthode de traduction en exploitant *Wikipédia* selon le schéma de pondération *Par défaut* de *Lucene.*

Système de Recherche	MAP
Système Monolingue	14.08%
Système Translingue (traduction directe)	
Sans Prétraitement	8.65%
Prétraitement (*Stemming leger*)	11.54%
Analyse morphologique (*MADA+TOKAN*)	10.34%
Combinaison (des trois variantes précédentes)	9.62%
Système Translingue (combiné avec une traduction transitive via la langue pivot le «français»)	8.37%

Tableau 7.6 Comparaison des *MAPs* de la méthode de traduction en exploitant *Wikipédia* avec *Google Translate* et *MyMemory*.

Méthode de Traduction	MAP	%/Monolingue	Gain/Perte
En Exploitant *Wikipedia* *(traduction directe)* *Stemming leger* *Sans prétaritement*	 11.54% 8.65%	 81.96% 61,43%	 -18.04% -38.57%
GoogleTranslate	13.80%	98.01%	-1.99%
Mymemory	13.81%	98.08%	-1.92%

Le Tableau 7.7, illustre un exemple de quatre (4) requêtes arabes de la collection de test, traduites vers l'*Anglais* selon les quatre variantes : sans aucun prétraitement, *stemming* léger, analyse morphologique avec MADA+TOKAN, et traduction transitive. Les traductions de références correspondantes à ces requêtes sont repectivement, «*Information Retrieval Systems*», «*Alternatives to Postscript*», «*Debt Rescheduling*», et «*Natural Language Processing*».

Par application de notre méthode de traduction, la première requête « نظم استرجاع المعلومات» («*nZm {strjAE AlmElwmAt*», «*Systèmes de Recherche d'Information*»)[25] a obtenu la traduction «*Information retrieval*» avec la première variante seulement. La deuxième requête «بدائل لبوست سكريبت» («*bdA}l lbwst skrybt*», «*Alternatives à Postscript*») a obtenu uniquement la traduction «*PostScript*» avec la troisième variante. Pour la troisième requête «إعادة جدولة الديون» («*<EAdp jdwlp Aldywn*», «*Rééchelonnement de la dette*») la traduction est «*Protectionism*» retournée par la deuxième et la troisième variante. Et enfin, la quatrième variante retourne «*Process*» comme traduction pour la quatrième requête «معالجة اللغات الطبيعية» («*mEAljp AllgAt AlTbyEyp*», «*Traitement du langage naturel*»).

[25] Translittération arabe de *Buckwalter* suivie de la traduction en français.

Le but de cet exemple est de montrer l'effet du prétraitement de la requête et de la traduction transitive sur le résultat de la traduction. En d'autres termes, si la traduction de la requête non prétraitée ne peut être trouvée, une segmentation des mots de la requête selon des schémas de tokenization variés peut améliorer considéralement les résultats de traduction. Par conséquent, l'integration d'une analyse morphologique au processus de traduction devient inévitable. De même pour la traduction transitive qui peut contribuer à l'amélioration des performances relativement à la taille du lexique pivot disponible.

Tableau 7.7 Les résultats de traduction pour quatre requêtes de la collection de données selon quatres variantes.

Requête	Sans prétraitement	Stemming léger	MADA+TOKAN	Traduction transitive
En arabe	نظم استرجاع المعلومات	نظم استرجاع معلومات	نظم استرجاع معلومات	
	بدائل لبوست سكريبت	بدائل لبوست سكريبت	بدائل بوست سكريبت	
	إعادة جدولة الديون	إعادة جدولة ديون	إعادة جدولة ديون	
	معالجة اللغات الطبيعية	معالجة لغات طبيعية	معالجة لغات طبيعية	
Traduction en anglais	Information retrieval	-	-	-
	-	-	PostScript	-
	-	Protectionism	Protectionism	-
	-	-	-	Process

Dans ce qui suit, nous allons montrer nos analyses réalisées sur les mesures de précision et rappel à différents niveaux : *P5, P10, P15, P20, P30, P100, P200, P500, P1000* et *R5, R10, R15, R20, R30, R100, R200, R500, R1000* qui mesurent respectivement, les présisions et les rappels après que *5, 10, 15, 20, 30, 100, 200, 500, 1000* documents ont été retrouvés par le moteur de recherche. Le Tableau 7.8 illustre les mesures relatives aux système monolingue et translingue pour la variante avec prétraitement de requêtes

(stemming leger). Les représentations graphiques correspondantes des précisions et rappels pour les deux systèmes sont montrés par la Figure 7.1, et la Figure 7.2.

Bien que le système translingue (*avec prétraitement des requêtes)* a obtenu une précision moyenne (*MAP*) inférieure à celle du monolingue, les précisions à différents niveaux obtenues sont encourageantes, car comme le montre le Tableau 7.8, les valeurs des précisions sont meilleures que celles du monolingue, et les rappels sont presques égales jusqu'à *R200.*

Tableau 7.8 Précisions et rappels des systèmes monolingue et translingue (variante *avec prétraitement des requêtes*).

Système Monolingue				Système Translingue (*stemming léger*)			
Précision à	% MAP	Rappel à	% Rappel	Précision à	% MAP	Rappel à	% Rappel
P5	*12%*	*R5*	*0%*	*P5*	*12%*	*R5*	*0%*
P10	*11%*	*R10*	*1%*	*P10*	*12%*	*R10*	*1%*
P15	*12%*	*R15*	*1%*	*P15*	*15%*	*R15*	*1%*
P20	*13%*	*R20*	*2%*	*P20*	*16%*	*R20*	*2%*
P30	*14%*	*R30*	*3%*	*P30*	*15%*	*R30*	*2%*
P100	*13%*	*R100*	*9%*	*P100*	*15%*	*R100*	*9%*
P200	*13%*	*R200*	*18%*	*P200*	*14%*	*R200*	*17%*
P500	*12%*	*R500*	*45%*	*P500*	*11%*	*R500*	*34%*
P1000	*11%*	*R1000*	*91%*	*P1000*	*9%*	*R1000*	*67%*

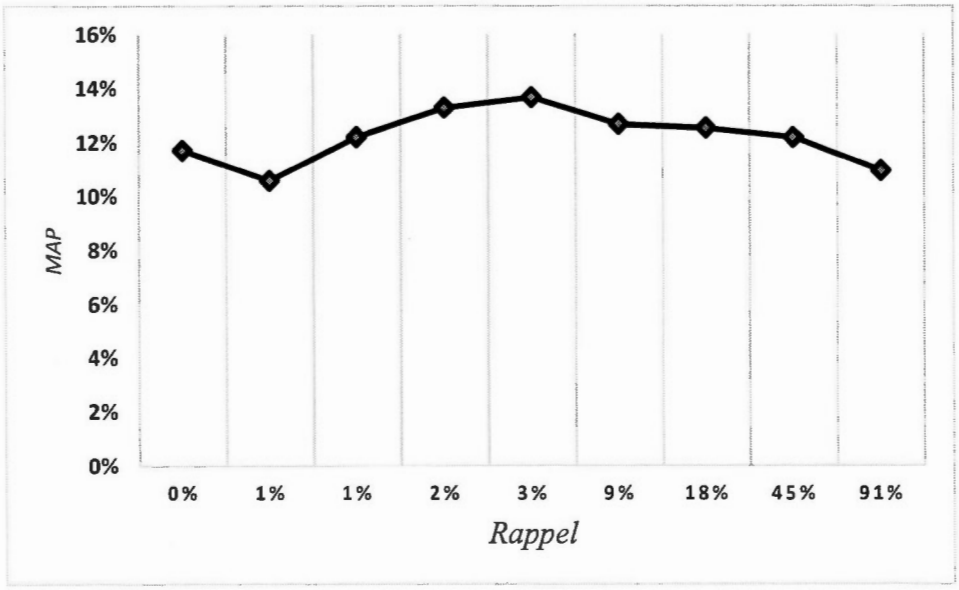


Figure 7.1 Graphe des précisions moyennes et rappels du système monolingue.

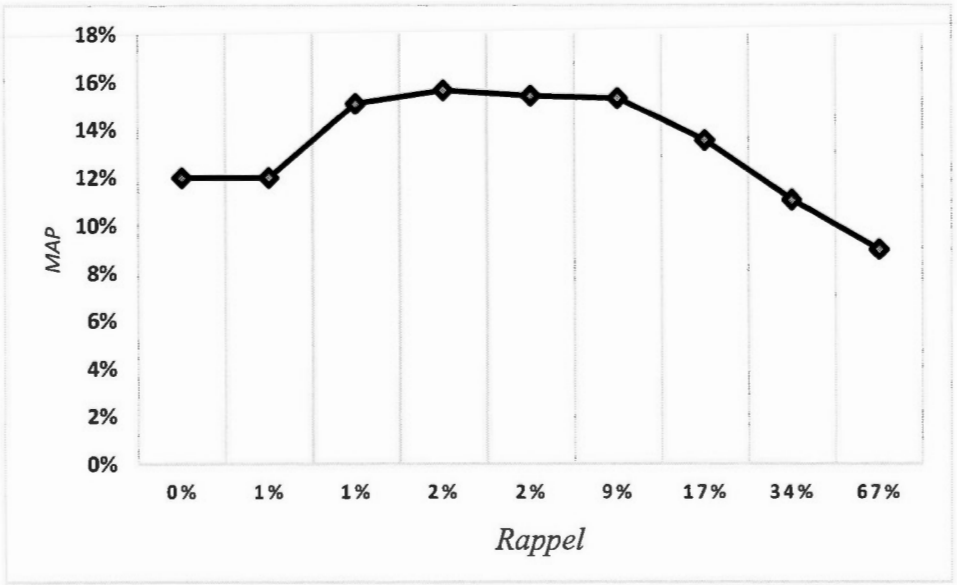


Figure 7.2 Graphe des précisions moyennes et rappels du système translingue (variante *avec prétraitement des requêtes*).

CONCLUSION

Le long de cette étude, nous avons présenté une méthode de traduction des requêtes pour la RIT *arabe-anglais* en exploitant l'encyclopédie *Wikipédia*. Notre méthode est basée sur l'extraction des traductions à partir des titres des articles *Wikipédia* qui ont un lien inter-langue. Le lexique bilingue *arabe-anglais* est extrait pour la traduction directe, et les lexiques *arabe-français* et *français-anglais* sont extraits pour la traduction transitive. Si la traduction de la requête entière ne peut être trouvée dans le lexique bilingue *arabe-anglais*, un stemming léger est d'abord effectué avant d'entammer la segmentattion et la traduction transitive considérées comme des opérations très coûteuses.

Comme la langue source de notre système RIT est l'*arabe*, une langue à morphologie très riche et complexe, nous avons proposé une méthode de segmentation de la requête en plusieurs unités lexicales qui s'adapte au type de structure de la phrase arabe. En outre, afin de réduire la complexité de notre application, nous avons paramétré notre méthode de segmentation à l'aide de deux formules différentes pour les requêtes longues et courtes, en fixant deux seuils dont les valeurs peuvent être modifiés et adaptés aux performances matérielles de l'ordinateur. Ceci, nous a permis de réduire considérablement le temps de traduction des requêtes longues.

De plus, nous avons montré l'effet positif du prétraitement des requêtes sur les performances du système de recherche translingue. Nous avons conclu que le stemming léger, ou une analyse morphologique plus complexe des requêtes étaient très utiles, où différentes formes des mots de la requête sont fournies et qui peuvent conduire à l'obtention de traductions dans les lexiques bilingues.

De même pour le résultat concernant la traduction transitive, où l'enrichissement du lexique peut être facilement réalisé en faisant une traduction indirecte par triangulation, c'est-à-dire, utiliser plus d'une langue pivot et qui peuvent être l'*espagnol*, l'*allemand* et le *chinois* dont l'extraction est possible avec notre application actuelle. De plus, étant donné que notre méthode de segmentation n'est pas limitée à la langue *arabe*, elle peut être également appliquée à d'autres langues choisies comme pivots, pour la traduction intermédiaire de la langue pivot vers la langue cible.

Au même titre que d'autres études de recherche, les résultats d'évaluations obtenus sont encourageantes, la meilleure précision moyenne (*MAP*) de notre système de recherche translingue (11.54%) représente 81.96% de celle du système monolingue. Nos résultats ont confirmé également la pertinence de l'encyclopédie *Wikipédia* comme une ressource pour la traduction. Néanmoins, le taux de précision moyenne de la méthode proposée reste faible et peut se justifier par le fait que dans la présente étude, l'exploitation de *Wikipédia* a été limitée seulement aux titres des articles. Par conséquent, ces résultats peuvent être meilleurs, car *Wikipédia* est une encyclopédie constamment mise à jour par les contributeurs. En effet, en perspective et dans le but d'améliorer la qualité des traductions obtenues, il serait utile de se pencher sur :

- L'intégration d'une analyse morphologique plus complexe au processus de traduction.
- L'exploitation du contenu des articles, des pages de redirection et des catégories des articles *Wikipédia*, afin d'obtenir une précision meilleure.
- Le filtrage des traductions extraites, en d'autres termes, écarter les titres qui ne sont pas rigoureux (non parallèles) extraits des articles *Wikipedia*, car les articles sont édités par des volontaires.
- Détection des translittérations dans les titres parallèles qui ne sont pas les bonnes traductions.

ANNEXES

APPENDICE A

PROGRAMME JAVA : *EXTRACTION DES TITRES DES ARTICLES WIKIPEDIA*

Ce programme se charge de l'extraction des lexiques bilingues à partir des archives *Wikipédia* (au format XML) de la langue source désirée. La version actuelle permet l'extraction des titres *arabe-anglais*, *arabe-français* et *français-anglais*. De plus, les lexiques dont la langue source et cible sont : l'*arabe*, l'*anglais*, le *français*, l'*allemand*, l'*espagnol*, ou le *chinois* peuvent être également extraits. La classe *«ExtractForm»* de ce programme permet à l'utilisateur de choisir l'archive *Wikipédia* à exploiter, la langue cible à traiter, et le chemin de sauvegarde ainsi que les noms des fichiers textes qui vont contenir les titres extraits de la langue source et cible. Après un parcours des pages *Wikipédia,* la classe *«Extraire»* transfère les titres extraits de la langue source et cible dans les fichiers correspondants.

Programme Java : «*Extraction des titres des articles Wikipédia*»

```
// La classe ExtractForm

package wikitranslate;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

/* @author Habiba Chakour

 * Programme d'extraction des titres bilingues à partir des articles Wikipédia
 * ayant un lien inter-langue, l'archive Wikipédia de la langue source à exploiter
 * doit être téléchargée avant le lancement de ce programme.
 * Les archives Wikipédia arabe et français ont été téléchargées des sites :
 * http://dumps.wikimedia.org/arwiki/latest/,
 * http://dumps.wikimedia.org/frwiki/latest/
 * Une fois téléchargées, les archives sont décompressées dans un répertoire, et
 * on aura les archives au format XML qui seront exploiter par le programme. */

public class ExtractForm extends javax.swing.JFrame {

    // Constructeur de la classe ExtractForm
    public ExtractForm() {
    initComponents();
  }
  @SuppressWarnings("unchecked")

  private void jToggleButton3ActionPerformed(java.awt.event.ActionEvent evt) {

    /* Lancement de l'opération d'extraction des titres bilingues.
     * Les quatre paramètres choisis par l'utilisateur, et qui sont utilisés par la classe
     * Extraire : l'archive à exploiter, le répertoire de sauvegarde et le nom du
     * fichier texte (de la langue source), le répertoire de sauvegarde et le nom du
     * fichier texte (de la langue cible), et la langue cible à traiter.*/

    Extraire Ext = new Extraire();
    Ext.Extraire(jTextField1.getText(),jTextField2.getText(),jTextField3.
      getText(),jComboBox1.getSelectedItem().toString().substring(0,2));
    try {
       Ext.ExtractTitle();
```

```
        } catch(Exception e) {
          JOptionPane.showMessageDialog(null, "Erreur lors d'ouverture de fichiers: "
                                                    + e.getMessage());
        }
        JOptionPane.showMessageDialog(null, "Extraction des titres bilingue
                                                terminée avec succés!.");
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        // Ouverture de l'explorateur de fichiers pour choisir
        // l'archive Wikipédia de la langue source à exploiter.

        jFileChooser1=new JFileChooser();
        jFileChooser1.setFileSelectionMode(JFileChooser.FILES_ONLY);
        jFileChooser1.setMultiSelectionEnabled(false);
        jFileChooser1.setDialogTitle("Choisir un Fichier");
        int result=jFileChooser1.showOpenDialog(null);
        if(result==JFileChooser.APPROVE_OPTION){
            String dirName=jFileChooser1.getSelectedFile().getAbsolutePath();
            jTextField1.setText(dirName);
        }
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

        // Ouverture de l'explorateur de fichiers pour choisir le répertoire
        // de sauvegarde et le nom du fichier texte (de la langue source).

        jFileChooser1=new JFileChooser();
        jFileChooser1.setFileSelectionMode(JFileChooser.FILES_ONLY);
        jFileChooser1.setMultiSelectionEnabled(false);
        jFileChooser1.setDialogTitle("Choisir un Fichier");
        int result=jFileChooser1.showOpenDialog(null);
        if(result==JFileChooser.APPROVE_OPTION){
            String dirName=jFileChooser1.getSelectedFile().getAbsolutePath();
            jTextField2.setText(dirName);
        }
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

        // Ouverture de l'explorateur de fichiers pour choisir le répertoire
```

```
    // de sauvegarde et le nom du fichier texte (de la langue cible).

    jFileChooser1=new JFileChooser();
    jFileChooser1.setFileSelectionMode(JFileChooser.FILES_ONLY);
    jFileChooser1.setMultiSelectionEnabled(false);
    jFileChooser1.setDialogTitle("Choisir un Fichier");
    int result=jFileChooser1.showOpenDialog(null);
    if(result==JFileChooser.APPROVE_OPTION){
       String dirName=jFileChooser1.getSelectedFile().getAbsolutePath();
       jTextField3.setText(dirName);
    }
}
/*@param args the command line arguments */

public static void main(String args[]) {

    /* Création et ouverture de la forme  ExtractForm à partir de laquelle
    * l'utilisateur peut saisir les paramètres nécessaires à l'exécution de
    * l'opération d'extraction des titres bilingues. */

    java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
          new ExtractForm().setVisible(true);}});
}
// Variables declaration - do not modify
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JFileChooser jFileChooser1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JToggleButton jToggleButton3;
// End of variables declaration
```

```
}
```

Programme Java : «La classe Extraire»

```
package wikitranslate;
import edu.jhu.nlp.language.Language;
import edu.jhu.nlp.wikipedia.PageCallbackHandler;
import edu.jhu.nlp.wikipedia.WikiPage;
import edu.jhu.nlp.wikipedia.WikiXMLParser;
import edu.jhu.nlp.wikipedia.WikiXMLParserFactory;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.util.Vector;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

/* @author Habiba Chakour */

public class Extraire {
    // Les quatre attributs de la classe Extraire correspondants aux quatres
    // paramètres saisis par l'utilisateur à partir de la forme ExtractForm.

    public static String dumpWiki;
    public static File outS;
    public static File outT;
    public static String Slang;

    public void Extraire(String dw,String s,String t,String Sl){
          // Constructeur de la classe Extraire
          dumpWiki = dw;
          outS  = new File (s);
          outT  = new File (t);
          Slang = Sl.toString();
    }

    public void ExtractTitle() throws IOException
    {
       // Création des deux fichiers textes qui vont contenir les titres
       // extraits de la langue source et cible respectivement.
```

```
final BufferedWriter br_ar = new BufferedWriter (new FileWriter (outS));
final BufferedWriter br_en = new BufferedWriter (new FileWriter (outT));

/* Analyse de l'archive Wikipedia de la langue source (fichier XML)
* avec l'analyseur SAXParser qui utilise moins de memoire vive que
 * le DOM, pour notre système RIT arabe-anglais nous avons utilisé :
 * les archives Wikipédia arabe du 10 Mars 2013 , et française du 09
 * Mars 2013. */

if (dumpWiki != null){
   WikiXMLParser wikip1 =
        WikiXMLParserFactory.getSAXParser(dumpWiki);
   try {
      // Parcours des pages Wikipédia avec l'analyseur SAXParser
      // et extraction des titres de la langue source et cible s'il y a
      // un lien inter-langue

       wikip1.setPageCallback(new PageCallbackHandler() {
       public void process(WikiPage page1) {
            String titleS=page1.getTitle().toString();
            String titleT=null;
            titleT=page1.getTranslatedTitle(Slang);
            if (titleS != null && titleT != null){
                 try {
                     br_ar.write(titleS);
                       br_ar.newLine();
                     br_en.write(titleT);
                       br_en.newLine();
                  } catch(IOException e) {
                        e.printStackTrace();
                  }
            }
       }
   });
   wikip1.parse();
   } catch(Exception e) {
        e.printStackTrace();
   }
 }
 br_ar.close();
 br_en.close();
}
}
```

APPENDICE B

PROGRAMME JAVA : *SEGMENTATION DE LA REQUÊTE*

Pour la segmentation de la requête de taille n mots, la classe «*SegmentQuery*» de ce programme utilise les méthodes «*Segmenter*», et «*MapSegments*». Le seul paramètre requis par la méthode «*Segmenter*» est le nombre n. Elle retourne comme résultat les 2^{n-1} segmentations possibles sous forme d'une suite de nombre séparés par un point. La méthode «*MapSegments*» s'occupe du mappage de toutes les segmentations générées aux unités lexicales qui leur correspondent dans la requête originale (les nombres à mapper représentent les tailles des unités lexicales en mots).

Programme Java : «La classe SegmentQuery»

```
package wikitranslate;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;
import java.util.Vector;
import java.util.TreeSet;
import java.util.Iterator;

/* @author Habiba Chakour
 * Programme de segmentation de la requête de taille n mots
 * en 2^(n-1) segmentations possibles. */

public class SegmentQuery {
    /* Les attributs de cette classe sont la requête originale, le vecteur
    * qui contient les tokens de la requête, et un autre vecteur dont les
    * éléments sont les segmentations générées (sous forme d'une suite
    * de nombres séparés par un point). */

    public static String Query;
    public static Vector<String> Tokens = new Vector<String>();
    public static Vector<String> Segments = new Vector<String>();

    public void SegmentQuery(String qr){
        // Tokenisation de la requête originale de type String
        if (qr.length() != 0){
            Query = qr;
      StringTokenizer tokens = new StringTokenizer(qr," .?!,:;&()[]'\"",false);
            try{ // Transfert des tokens de la requête dans le vecteur Tokens
              while (tokens.hasMoreElements()){
                Tokens.add(tokens.nextToken().toString());
              }
            } catch (NoSuchElementException exception){
              System.out.println ("Erreur lors de la lecture: " +
                                        exception.getMessage());
```

```
        }
    }
}

public void Segmenter(int m){
  /* Segmentation de la requête de taille m mots en 2^(m-1) segmentations.
   * Cette méthode utilise une matrice de dimension m-1 lignes et 2^(m-1)
   * colonnes, les lignes de la matrice sont appelés «niveaux» dans
   * notre programme, ils représentent les tailles des unités lexicales
   * qui varient de 2 à m. */

  if (m>2){
   TreeSet ts = new TreeSet();
   String[][] Mlevel = new String[m-1][(int)Math.pow(2, m-1)];

         /* Le processus de segmentation est itératif, le programme commence
          * par mémoriser dans la matrice toutes les décompositions possibles
          * de chaque niveau n en deux nombres (i,j), tel que i+j = n, un point
          * est rajouté entre ces deux nombres (i.j) avant de transferer cet element
          * construit dans la matrice. */

   for(int n=2;n<=m;n++) {
      for(int i=n-1;i>=1;i--){
         for(int j=1;j<=n-1;j++){
            if(i+j==n){
              Mlevel[n-2][j-1]= Integer.toString(i)+"."+Integer.toString(j) ;
            }
         }
      }
   }
         /* Maintenant on revient sur la matrice et à partir de la 2ème ligne,
          * pour chaque élément (i.j) de la matrice, on remplace i, puis j par
          * toutes les décompositions possibles qui lui correspondent aux
          * niveaux i et j précédents. */

   for (int n=1;n<=m-2;n++) {
     ts.clear();
```

```
        for (int i=0;i<=n/2;i++){
            String[] s =  Mlevel[n][i].toString().split("\\.");
            int level = Integer.parseInt(s[0]);
            if ( level != 1) {
               for (int k=0;k<(int)Math.pow(2, level-1)-1;k++) {
                  ts.add(Mlevel[level-2][k] + "." + s[1] );
                  ts.add(s[1] + "." + Mlevel[level-2][k]);
               }
            }
         }
               // Transferer l'ensemble des segmentations sans repetition
               // du niveau n dans la matrice
               Iterator tsIt = ts.iterator();
         for(int k=(int)Math.pow(2, n+1)-2;k>=n+1;k--){
            Mlevel[n][k] = tsIt.next().toString();
         }
       }

       // Transfert de la derniere ligne de la matrice vers le vecteur Segments
        for (int k=0;k<(int)Math.pow(2,m-1)-1;k++){
        Segments.add(Mlevel[m-2][k]);
       }
      }
    }
    public void MapSegments(){
       /* Pour produire des segmentations formées à partir des mots de la
       * requête originale, cette méthode travaille avec le contenu des
       * vecteurs Tokens et Segments et fait le mappage des différentes
       * segmentations générées aux unités lexicales qui leur correspondent. */

      String[] s;
      String chunk;
      String chunks;
      if (Tokens.size() == 2){
          Segments.add(Tokens.elementAt(0)+"."+Tokens.elementAt(1));
```

```
        } else {
              for(int i=0;i<Segments.size();i++){
                int l=0;
                chunks = "";
                s = Segments.elementAt(i).split("\\.");
                for (int j=0;j<s.length;j++){
                    chunk = "";
                    int k=1;
                    while(k<=Integer.parseInt(s[j])){
                        chunk = chunk+Tokens.elementAt(l).trim()+" ";
                         k++; l++;
                    }
                    chunks = chunks+chunk+".";
                }
                chunks = chunks.substring(0, chunks.length()-2);
                Segments.set(i, chunks);
             }
         }
      }
}
```

APPENDICE C

PROGRAMME JAVA: *TRADUCTION DE LA REQUÊTE*

Dans ce programme, le processus de traduction de la requête est assuré par la classe «*TranslateForm*». Cette classe utilise la classe «*Requête*» qui est chargée de la tokenisation et du prétraitement de la requête originale. Avant le lancement de la traduction, si les fichiers textes de la langue source et cible (traduction directe), et ceux de la langue source, cible et pivot (traduction transitive) ne sont pas fournis par l'utilisateur, alors, les fichiers «*arTitles.txt*», «*enTitles.txt*» (de la traduction directe), et «*ariTitles.txt*», «*frTitles.txt*», «*friTitles.txt*», «*eniTitles.txt*» (de la traduction transitive) seront utilisés.

Programme Java : «*La classe Requête*»

```
package wikitranslate;
import java.util.Vector;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;

/* @author Habiba Chakour
* Programme de traduction de la requête de la langue source vers la langue
* cible pour une traduction directe, et de la langue source vers la langue
* pivot, et puis de la langue pivot vers la langue cible pour une traduction
* transitive.
* Les lexiques bilingues arabe-anglais (les fichiers textes de la langue
* source et cible) de la traduction directe, et arabe-français, et français-
* anglais (les fichiers textes de la langue source, pivot, et cible) de la
* traduction transitive, doivent être extraits avant le lancement de ce
* programme. */

public class Requete {
   /* Les attributs de cette classe sont la requête originale, la
   * requête prétraitée, le vecteur des tokens de la requête
   * originale, le vecteur des tokens de la requête prétraitée,
   * et les seuils sQ et sUl. */

   public static String Query;
   public static String sQuery;
   public static Vector<String> Tokens = new Vector<String>();
   public static Vector<String> sTokens = new Vector<String>();
   public static int seuilQ;
   public static int seuilUlex;

   public void Requete(String qr, int sQ, int sUl){
            // Constructeur de la classe Requête
      seuilQ = sQ;
      seuilUlex = sUl;
      // Tokenisation de la requête originale et prétraitée
      if (qr.length() != 0){
```

```
            Query = qr;
            sQuery = "";
        StringTokenizer tokens = new StringTokenizer(qr," .?!,:;&()[]'\"",false);
             try{
                String token = null;
                while (tokens.hasMoreElements()){
                   token = tokens.nextToken().toString();
                    Tokens.add(token);
                   // Prétraitement de la requête (stemming léger)
                    if (token.startsWith("ال")){
                       sTokens.add(token.substring(2));
                       sQuery = sQuery +token.substring(2)+" ";
                    } else {
                          if (token.startsWith("بال") || token.startsWith("فال") ||
                            token.startsWith("وال") || token.startsWith("كال")){
                            sTokens.add(token.substring(3));
                            sQuery = sQuery +token.substring(3)+" ";
                       } else {
                             sTokens.add(token);
                             sQuery = sQuery +token+" ";
                       }
                    }
                }
             } catch (NoSuchElementException exception){
                System.out.println ("Erreur lors de la lecture: " +
exception.getMessage());
             }
          }
    }

public String[] DiviseRequete(Vector<String> tokens){
       // Division de la requête de taille n mots en petites requêtes, dont
       // n/sUl requêtes sont de taille sUl et une autre de taille n ≡ sUl

       int x = tokens.size()/seuilUlex;
```

```
        int y = tokens.size()%seuilUlex;
        int i = 0;
        int taille = x;
        if (y!=0){
           taille = x+1;
        }
        String[] q = new String[taille];
        String s = null;
        while(i<x*seuilUlex){
           s = "";
           for(int j=i;j<i+seuilUlex;j++){
                s = tokens.elementAt(j) + " ";
           }
           q[i/seuilUlex] = s;
           i+=seuilUlex;
        }
        if (y!=0){
          s = "";
          for(int j=i;j<tokens.size();j++){
               s = tokens.elementAt(j) + " ";
          }
          q[i/seuilUlex] = s;
        }
        return q;
    }
}
```

Programme Java : «*La classe TranslateForm*»

```
package wikitranslate;
import java.io.IOException;
import java.util.Date;
import java.util.Vector;
import javax.swing.JOptionPane;
```

```
/** @author Habiba Chakour
* La classe TranslateForm traduit une requête de la langue source vers la
* langue cible, en proposant plusieurs options : traduction combinée à la
* traduction transitive, traduction directe avec ou sans prétraitement de la
* requête, et traduction transitive, juste le code de la première option est
* fourni dans ce programme. */

public class TranslateForm extends javax.swing.JFrame {
   // Constructeur de la classe TranslateForm
   public TranslateForm() {
      initComponents();
   }
   @SuppressWarnings("unchecked")

 private void jToggleButton9ActionPerformed(java.awt.event.ActionEvent evt)
 {
    // Traduction directe combinée à la traduction transitive
    long start = new Date().getTime();
    try {
       // Les valeurs par défaut des seuils sQ et sUl
       int sQ  = 20;
       int sUl = 10;
       // Modification des seuils sQ et sUl par l'utilisateur
       try{
            if(!jTextField3.getText().isEmpty()) {
            sQ = Integer.parseInt(jTextField3.getText());
          }
          if(!jTextField4.getText().isEmpty()){
            sUl= Integer.parseInt(jTextField4.getText());
          }
       } catch (NumberFormatException ex){
          JOptionPane.showMessageDialog(null,"Il faut saisir un nombre
             !!","Probléme de format",JOptionPane.ERROR_MESSAGE);
       }
       // Création de l'objet Requête avec les paramètres nécessaires
```

```
Requete Q = new Requete();
Q.Requete(jTextField1.getText(), sQ, sUl);
// Vérifier si la taille de la requête ne dépasse pas le seuil sQ fixé
if(Q.Tokens.size()>Q.seuilQ){
  JOptionPane.showMessageDialog(null, "Attention: La taille de la
                    requête dépasse le seuil permis!!!.","Probléme de
                   dépassement",JOptionPane.ERROR_MESSAGE);
 } else {
     // Création de l'objet de traduction (les fichiers textes de la langue
     // source et cible choisis par l'utilisateur peuvent être utilisés, sinon
     // les fichiers «arTitles.txt» et «enTitle.txt» seront chargés).

     WikiTranslate wT = new WikiTranslate();
     if (jTextField8.getText().isEmpty() ||
       jTextField9.getText().isEmpty()){
         wT.WikiTranslate("arTitles.txt", "enTitles.txt");
     } else {
          wT.WikiTranslate(jTextField8.getText(),
           jTextField9.getText());
     }
     // Traduction directe de la requête entière sans prétraitement
     String Trad = wT.TraduireQuery(Q.Query);
     System.out.println("Traduction directe (requête entière sans
                                          prétraitement): "+Trad);
     if (Trad == null){
       System.out.println("Traduction directe (requête entière avec
                                             prétraitement): "+Trad);
       // Traduction directe de la requête entière avec prétraitement
       Trad = wT.TraduireQuery(Q.sQuery);
     }
     if (Trad != null){
       jTextField2.setText(Trad);
     } else {
         // Si aucune traduction n'a été trouvée pour la requête entière
         // alors segmenter
```

```
System.out.println("Traduction directe (requête segmentée):
                                                    "+Trad);
SegmenteRequete segQ = new SegmenteRequete();
segQ.Segmenter(Q.Tokens.size(),Q.seuilUlex);
Vector<String> vTrad  = new Vector<String>();
// Si la taille de la requête ne dépasse pas le seuil sUl alors
// traduire 2^n-1 segmentations avec ou sans prétraitement
if (Q.Tokens.size()<=Q.seuilUlex){
   Trad  = wT.TraduireSegments(segQ.MapSegments(Q.
                                        Tokens), sQ, sUl);
    segQ.Segments.clear();
} else {
      // Si la taille de la requête dépasse sUl diviser la
      // requête en petites requêtes de tailles égales à sUl
      // sauf une de taille r
      String[] dQ  = Q.DiviseRequete(Q.Tokens);
      if (Q.Tokens.size()>Q.seuilUlex){
         int r = Q.Tokens.size()%Q.seuilUlex;
         for(int k=0;k<dQ.length;k++){
            String sTrad = wT.TraduireQuerySquery(dQ[k], sQ,
                                                    sUl);
            if (sTrad != null){
               // Une Traduction a été trouvé pour la petite requête
               Trad = Trad + sTrad;
            } else {
                  // Segmenter la petite requête et traduire les
                  // segmentations possibles (2^sUl-1 ou 2^r-1) avec
                  // ou sans prétraitement
                  Q.Tokens.clear();
                  Q.sTokens.clear();
                  Q.Requete(dQ[k], sQ, sUl);
                  if (k == dQ.length-2 && r>2){
SegmenteRequete segY = new  SegmenteRequete();
                     segY.Segmenter(r,Q.seuilUlex);
                     Trad = Trad +
```

```
wT.TraduireSegments(segY.MapSegments(Q.Tokens), sQ, sUl);
                                segY.Segments.clear();
                            } else {
                                    Trad = Trad +
wT.TraduireSegments(segQ.MapSegments(Q.Tokens), sQ, sUl);
                            }
                        }
                    }
                }
            }
            wT.sT.clear();
            wT.tT.clear();
            // Si aucune traduction n'est issue de la traduction directe
            // alors entamer une traduction transitive
            if (Trad.isEmpty()){
                System.out.println("Traduction Transitive: "+Trad);
                // Charger les fichiers textes de la langue source et pivot
                // choisis par l'utilisateur, sinon on utilise ariTitles.txt et
                // frTitles.txt.
                    if (jTextField5.getText().isEmpty() ||
                        jTextField7.getText().isEmpty()){
                        wT.WikiTranslate("ariTitles.txt", "frTitles.txt");
             } else {
                        wT.WikiTranslate(jTextField5.getText(),
                                        jTextField7.getText());
              }
                // Traduction de la requête entière avec et sans prétraitement
                // de la langue source vers la langue pivot
                Q.Tokens.clear();
                Q.sTokens.clear();
                Q.Requete(jTextField1.getText(), sQ, sUl);
                Trad = wT.TraduireQuery(Q.Query);
                if (Trad == null){
                    Trad = wT.TraduireQuery(Q.sQuery);
                }
```

```
wT.sT.clear();
wT.tT.clear();
// Traduction de la requête de la langue pivot
// vers la langue cible
WikiTranslate wTT = new WikiTranslate();
if (Trad != null){
 // Charger les fichiers textes de la langue pivot et cible
 // choisis par l'utilisateur, sinon on utilise friTitles.txt et
 // eniTitles.txt
  if (jTextField7.getText().isEmpty() ||
      jTextField6.getText().isEmpty()){
     wTT.WikiTranslate("friTitles.txt", "eniTitles.txt");
   } else {
          wTT.WikiTranslate(jTextField7.getText(),
                            jTextField6.getText());
   }
   jTextField2.setText(wTT.TraduireQuery(Trad));
   wTT.sT.clear();
   wTT.tT.clear();
 } else {// Segmenter la requête et traduire transitivement
      // chaque segment
     segQ.Segments.clear();
     segQ.Segmenter(Q.Tokens.size(),Q.seuilUlex);
     wT.WikiTranslate("ariTitles.txt", "frTitles.txt");
     vTrad.clear();
     if (Q.Tokens.size()<=Q.seuilUlex){
        // Si la taille de la requête ne dépasse pas sUl alors
        // segmenter et traduire les différentes segmentations
        // possibles
        vTrad  =
wT.TraduireSegmentsVect(segQ.MapSegments(Q.Tokens), sQ, sUl);
      } else { // Si la taille de la requête dépasse sUl alors diviser
           // la requête en petites requêtes de tailles égales à
         // sUl sauf la dernière petite requête est de taille r
         String[] dQ  = Q.DiviseRequete(Q.Tokens);
```

```
                        // Traduction des petites requêtes
                    vTrad  = wT.TraduireArrayQuery(dQ, sQ, sUl);
                }
                wT.sT.clear();
                wT.tT.clear();
                // Si les traductions ne sont pas vides
                if (!vTrad.isEmpty()) {
                   wTT.WikiTranslate("friTitles.txt", "eniTitles.txt");
                   String[] S  = new String[vTrad.size()];
            System.arraycopy(vTrad.toArray(), 0, S, 0, vTrad.size());
                   Vector<String> vTTrad =
                                wT.TraduireArrayQuery(S);
                   Trad = "";
                   for (int i=0; i<vTTrad.size(); i++){
                      if (!vTTrad.elementAt(i).isEmpty()) {
                         Trad = Trad + vTTrad.elementAt(i)+ " ";
                      }
                   }
                }
                jTextField2.setText(Trad);
                System.out.println("Traduction transitive: "+Trad);
                wTT.sT.clear();
                wTT.tT.clear();
                segQ.Segments.clear();
             }
          }
          jTextField2.setText(Trad);
          Q.Tokens.clear();
          Q.sTokens.clear();
          // Fin du processus de traduction
       }
    }
 } catch (IOException exception) {
    JOptionPane.showMessageDialog(null, "Erreur lors de la lecture: " +
```

```
                                                    exception.getMessage());
    }
    long end = new Date().getTime();
    JOptionPane.showMessageDialog(null, "Traduction transitive combinée à la
traduction directe terminée: "+(end-start)+" Milisecondes");
    }

public static void main(String args[]) {
  /* Creation et ouverture de la forme TranslateForm */
    java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
        new TranslateForm().setVisible(true);
      }
    });
  }
  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel jLabel2;
  private javax.swing.JLabel jLabel3;
  private javax.swing.JLabel jLabel4;
  private javax.swing.JLabel jLabel5;
  private javax.swing.JLabel jLabel6;
  private javax.swing.JLabel jLabel7;
  private javax.swing.JLabel jLabel8;
  private javax.swing.JTextField jTextField1;
  private javax.swing.JTextField jTextField2;
  private javax.swing.JTextField jTextField3;
  private javax.swing.JTextField jTextField4;
  private javax.swing.JToggleButton jToggleButton1;
  private javax.swing.JToggleButton jToggleButton2;
  private javax.swing.JToggleButton jToggleButton4;
  private javax.swing.JToggleButton jToggleButton5;
  private javax.swing.JToggleButton jToggleButton6;
  private javax.swing.JToggleButton jToggleButton7;
  // End of variables declaration
}
```

APPENDICE D

PROGRAMME JAVA : *MOTEUR DE RECHERCHE*

Ce programme intègre les fonctionnalités de recherche et d'indexation du moteur de recherche *Lucene* (version 4.1.0). La classe «*Index*» permet d'indexer une collection de données selon trois modèles de pondération (*par défaut, Tf.Idf, BM25*). En raison de la limite du temps imparti, nos expérimentations ont été effectuées uniquement avec le modèle *par défaut*. Le choix du modèle de pondération, du répertoire da sauvegarde de l'index, et de la collection de données sont fournis dans la classe «*IndexForm*». La classe «*Research*» retourne à l'utilisateur la liste triée des documents retrouvés conformément au modèle de pondération choisi.

Programme Java : «*La classe IndexForm*»

```
package wikitranslate;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
/**@author Habiba Chakour
* Programme d'indexation de la collection de données
* Pondération selon les trois modèles du moteur de
* recherche Lucene : Par éfaut, Tf.Idf, BM25 */

public class IndexForm extends javax.swing.JFrame {
   /** Constructeur de la classe IndexForm */
   public IndexForm() {
      initComponents();
   }
   @SuppressWarnings("unchecked")

   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // Indexation de la collection de données selon les trois modèles de
      // pondération de Lucene : Par défaut, Tf.Idf, et BM25

      char s = '0';  // modèle de pondération par défaut
     // Création de l'objet de type Index
      Index ind = new Index();
      if (buttonGroup1.isSelected(jRadioButton2.getModel())) {
         s = '1';  // modèle de pondération Tf.Idf
      } else {
         if (buttonGroup1.isSelected(jRadioButton3.getModel())) {
            s = '2'; // modèle de pondération BM25
         }
      }
     // Lancement de l' opération d'indexation en utilisant
     // les paramètres introduits par l'utilisateur
      try{
         ind.Index(jTextField1.getText(),jTextField2.getText(),s);
```

```
            ind.IndexFileOrDirectory();
        } catch (Exception e) {

JOptionPane.showMessageDialog(null, "Erreur lors d'ouverture de fichiers: " +
e.getMessage());
        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // Ouverture de l'explorateur de fichiers pour choisir
        // le repertoire de sauvegarde de l'index créé
        jFileChooser1=new JFileChooser();
jFileChooser1.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        jFileChooser1.setMultiSelectionEnabled(false);
        jFileChooser1.setDialogTitle("Choisir un Fichier");
        int result=jFileChooser1.showOpenDialog(null);
        if(result==JFileChooser.APPROVE_OPTION){
            String dirName=jFileChooser1.getSelectedFile().getAbsolutePath();
            jTextField1.setText(dirName);
        }
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // Ouverture de l'explorateur de fichiers pour
        // choisir la collection de données à indexer
        jFileChooser1=new JFileChooser();
jFileChooser1.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        jFileChooser1.setMultiSelectionEnabled(false);
        jFileChooser1.setDialogTitle("Choisir un Fichier");
        int result=jFileChooser1.showOpenDialog(null);
        if(result==JFileChooser.APPROVE_OPTION){
            String dirName=jFileChooser1.getSelectedFile().getAbsolutePath();
            jTextField2.setText(dirName);
        }
    }
```

```
  public static void main(String args[]) {
  /* Création et ouverture de la forme IndexForm */
    java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
        new IndexForm().setVisible(true);
      }
    });
  }
  // Variables declaration - do not modify
  private javax.swing.ButtonGroup buttonGroup1;
  private javax.swing.JButton jButton1;
  private javax.swing.JButton jButton2;
  private javax.swing.JButton jButton3;
  private javax.swing.JFileChooser jFileChooser1;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel jLabel2;
  private javax.swing.JLabel jLabel3;
  private javax.swing.JLabel jLabel4;
  private javax.swing.JLabel jLabel6;
  private javax.swing.JRadioButton jRadioButton1;
  private javax.swing.JRadioButton jRadioButton2;
  private javax.swing.JRadioButton jRadioButton3;
  private javax.swing.JTextField jTextField1;
  private javax.swing.JTextField jTextField2;
  // End of variables declaration
}
```

Programme Java : «*La classe Index*»

```
import java.io.IOException;
import java.util.Date;
import javax.swing.JOptionPane;
import org.apache.lucene.search.similarities.Similarity;
/** @author Habiba Chakour */
```

```
public class Index {
   // Les attributs de la classe Index : le répertoire de l'Index,
   // la collection de données, et le modèle de pondération
   // choisi.
   public static File IndexLocation;
   public static FSDirectory FS;
   public static File DC;
   public static char Sm;
   private ArrayList<File> queue = new ArrayList<File>();

   public void Index(String il, String dc, char S) throws IOException{
       // Constructeur de la classe Index
       IndexLocation = new File(il);
       FS = FSDirectory.open(IndexLocation);
       DC = new File(dc);
       Sm = S;
   }

   public void IndexFileOrDirectory()throws IOException {
       long start = new Date().getTime();
       // Configurartion de l'analyseur de Lucene,
       // l'analyseur StandardAnalyzer est choisi
       StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_41);
        IndexWriterConfig config  = new IndexWriterConfig(Version.LUCENE_41,
analyzer);
       // Configuration de la formule de similarité de Lucene
       // conformément au modèle de pondération choisi par
       // l'utilisateur
       if (Sm == '1'){
         TFIDFSimilarity Sim = new DefaultSimilarity() {
                        public float coord(int overlap, int maxOverlap) {
                            return 1;
                        }
                        public float queryNorm(float sumOfSquaredWeights){
```

```
                    return 1;
                }
                public float tf(float freq) {
                    return
Float.parseFloat(Double.toString(Math.log(Double.parseDouble(Float.toString(1+fre
q)))));
                }
            };
        config.setSimilarity(Sim);
    } else {
        if (Sm == '2'){
            Similarity Sim = new BM25Similarity();
            config.setSimilarity(Sim);
        }
    }
    // Création de l'index
    IndexWriter w = new IndexWriter(FS, config);

    // Transfert de la liste des fichiers de la
    // collection de données dans la liste queue

    addFiles(DC);
    int originalNumDocs = w.numDocs();
    // Parcours de la liste des documents à indexer
    for (File f : queue) {
      FileReader fr = null;
      try {
        Document doc = new Document();
        // Ajouter le document à l'index
        fr = new FileReader(f);
        doc.add(new TextField("contents", fr));
        doc.add(new StringField("path", f.getPath(), Field.Store.YES));
        doc.add(new StringField("filename", f.getName(), Field.Store.YES));
        w.addDocument(doc);
        System.out.println("Ajouté: " + f);
```

```
            } catch (Exception e) {
                System.out.println("Non Ajouté: " + f);
            } finally {
            fr.close();
            }
        }
        int newNumDocs = w.numDocs();
        System.out.println("");
        System.out.println("**************************");
        System.out.println((newNumDocs - originalNumDocs) + " Documents
                                                        Ajoutés.");
        System.out.println("**************************");
        queue.clear();
        w.close();
        long end = new Date().getTime();
        JOptionPane.showMessageDialog(null, " Documents Ajoutés=
"+(newNumDocs - originalNumDocs)+" Temps= " + (end - start) + " milliseconds");
    }
    private void addFiles(File file) {
    if (!file.exists()) {
       JOptionPane.showMessageDialog(null, " Fichier Inexistant ! ");
     }
     if (file.isDirectory()) {
       for (File f : file.listFiles()) {
         addFiles(f);
       }
     } else {
       String filename = file.getName().toLowerCase();
        queue.add(file);
     }
   }
}
```

Programme Java : «*La classe ResearchForm*»

```
package wikitranslate;
import java.io.BufferedWriter;
import java.util.Vector;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.nio.file.Files;
import java.nio.charset.Charset;
import java.nio.file.Paths;
import javax.swing.JFileChooser;

/**@author Habiba Chakour */

public class ResearchForm extends javax.swing.JFrame {
   /** Constructeur de la classe ResearchForm */
   public ResearchForm() {
      initComponents();
   }
   @SuppressWarnings("unchecked")
   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

      // Choix du modèle de pondération de Lucene
      char s = '0'; // modèle par Défaut
      if (buttonGroup1.isSelected(jRadioButton2.getModel())) {
         s = '1'; // modèle Tf.Idf
      } else {
         if (buttonGroup1.isSelected(jRadioButton3.getModel())) {
            s = '2'; // modèle BM25
         }
      }
      try{
      // Transfert du fichier des requêtes à traiter dans le vecteur
            Vector<String> Queries = new
            Vector<String>(Files.readAllLines(Paths.get(jTextField3.getText()),
```

```
                                                    Charset.defaultCharset()));
        // Création du fichier texte qui va contenir la liste des
        // documents trouvés (les résultats de la recherche)
            PrintWriter resRech = new PrintWriter(new BufferedWriter(new
                                        FileWriter(jTextField4.getText())));
        for(int i=0;i<Queries.size();i++){
          // Recherche de la requête en cours
          Research res = new Research();
          res.Research(jTextField1.getText(),Queries.elementAt(i), i+1, s);
      for(String line: Files.readAllLines(Paths.get(res.ResearchDocs().getName()),
      Charset.defaultCharset())){
                    // Transfert du résultat de la recherche dans le fichier texte
            resRech.println(line);
          }
        }
        resRech.close();
      } catch (Exception e) {
        System.out.println(e.getMessage());
      }
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
      // Ouverture de l'explorateur de fichiers pour choisir l'index à ouvrir
      jFileChooser1=new JFileChooser();
jFileChooser1.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
      jFileChooser1.setMultiSelectionEnabled(false);
      jFileChooser1.setDialogTitle("Choisir un Fichier");
      int result=jFileChooser1.showOpenDialog(null);
      if(result==JFileChooser.APPROVE_OPTION){
        String dirName=jFileChooser1.getSelectedFile().getAbsolutePath();
        jTextField1.setText(dirName);
      }
    }
```

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // Ouverture de l'explorateur de fichiers pour choisir le fichier des requêtes
    jFileChooser1=new JFileChooser();
    jFileChooser1.setFileSelectionMode(JFileChooser.FILES_ONLY);
    jFileChooser1.setMultiSelectionEnabled(false);
    jFileChooser1.setDialogTitle("Choisir un Fichier");
    int result=jFileChooser1.showOpenDialog(null);
    if(result==JFileChooser.APPROVE_OPTION){
        String dirName=jFileChooser1.getSelectedFile().getAbsolutePath();
        jTextField3.setText(dirName);
    }
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // Ouverture de l'explorateur de fichiers pour choisir le répertoire
    // de sauvegarde du fichier résultat
    jFileChooser1=new JFileChooser();
    jFileChooser1.setFileSelectionMode(JFileChooser.FILES_ONLY);
    jFileChooser1.setMultiSelectionEnabled(false);
    jFileChooser1.setDialogTitle("Choisir un Fichier");
    int result=jFileChooser1.showOpenDialog(null);
    if(result==JFileChooser.APPROVE_OPTION){
        String dirName=jFileChooser1.getSelectedFile().getAbsolutePath();
        jTextField4.setText(dirName);
    }
}
/** @param args the command line arguments */

public static void main(String args[]) {
/* Creation et ouverture de la forme ResearchForm*/
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new ResearchForm().setVisible(true);
        }
    });
```

```
    }
    // Variables declaration - do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JFileChooser jFileChooser1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JRadioButton jRadioButton3;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    // End of variables declaration
}
```

Programme Java : «*La classe Research*»

```
package wikitranslate;
import org.apache.lucene.document.Document;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.TopScoreDocCollector;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.search.similarities.Similarity;
import org.apache.lucene.search.similarities.DefaultSimilarity;
```

```
import org.apache.lucene.search.similarities.BM25Similarity;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.search.Query;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.util.Version;
import java.io.File;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.BufferedWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import javax.swing.JOptionPane;
import org.apache.lucene.search.similarities.TFIDFSimilarity;

/** @author Habiba Chakour */

public class Research {
   public static String fQuery;
   public static int nQuery;
   public static File IndexLocation;
   public static FSDirectory FS;
   public static char Sm;

   public void Research(String il, String fQ, int nR, char S) throws IOException{
       // Les attributs de la classe Research : l'index à ouvrir, le fichier des
       // requêtes, numero de la requête, et la similarité
       IndexLocation = new File(il);
       FS = FSDirectory.open(IndexLocation);
       fQuery = fQ;
       nQuery = nR;
       Sm = S;
   }
   public File ResearchDocs(){
      try {
```

```
        // Création de l'analyseur et ouverture de l'index
      StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_41);
        IndexReader reader = DirectoryReader.open(FS);
        IndexSearcher searcher = new IndexSearcher(reader);
    TopScoreDocCollector collector = TopScoreDocCollector.create(242918, true);
        if (Sm == '1'){
          TFIDFSimilarity Sim = new DefaultSimilarity() {
                        public float coord(int overlap, int maxOverlap) {
                            return 1;
                        }
                        public float queryNorm(float sumOfSquaredWeights){
                            return 1;
                        }
                        public float tf(float freq) {
                            return
Float.parseFloat(Double.toString(Math.log(Double.parseDouble(Float.toString(1+fre
q)))));
                        }
                    };
          searcher.setSimilarity(Sim);
        } else {
            if (Sm == '2'){
                Similarity Sim = new BM25Similarity();
                searcher.setSimilarity(Sim);
            }
        }
        // Création du fichier des résultats fRes
        File fRes   = new File("res.txt");
        PrintWriter resRech;
        resRech =  new PrintWriter(new BufferedWriter(new FileWriter(fRes)));
        // Limiter le nombre des chiffres significatifs du Score
        DecimalFormat f = new DecimalFormat();
        f.setMaximumFractionDigits(4);
        try{
            // Création et analyse de la requête
```

```
                Query q = new QueryParser(Version.LUCENE_41, "contents",
                                                    analyzer).parse(fQuery);
            // Rechercher les documents pour chaque requête
              searcher.search(q, collector);
            // Collecter les documents trouvés
              ScoreDoc[] hits = collector.topDocs().scoreDocs;
              // Afficher les résultats
              System.out.println("Found " + hits.length + " hits.");
              //resRech.println("Query: "+fQuery+"  "+hits.length);
              for(int i=0;i<hits.length;++i) {
                 int docId = hits[i].doc;
                 Document d = searcher.doc(docId);
                 resRech.println(nQuery + "\t0\t" +d.get("filename")+ "\t\t"+(i + 1) +
".\t"+"score=" + f.format(hits[i].score)+"\t0");
              }
            } catch (Exception e) {
JOptionPane.showMessageDialog(null, "Erreur lors de la recherche: " +
e.getMessage());
            }
        resRech.close();
        reader.close();
        analyzer.close();
        return fRes;
       } catch (Exception e) {
      JOptionPane.showMessageDialog(null, "Erreur lors de l'ouverture des fichiers: "
+ e.getMessage());
      }
      return null;
    }
}
```

BIBLIOGRAPHIE

[1] Lehtokangas, R., Airio, E., et Järvelin, K., *Transitive dictionary translation challenges direct dictionary translation in CLIR.* Information Processing & Management, 2004, 40(6), p. 973-988.

[2] Kishida, K. et Kando, N., *A hybrid approach to query and document translation using a pivot language for cross-language information retrieval.* Dans Proceedings of the 6th international conference on Cross-Language Evalution Forum: accessing Multilingual Information Repositories, 2006, p. 93-101.

[3] Gollins, T. et Sanderson, M., *Improving cross language retrieval with triangulated translation.* Dans Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, 2001, p. 90-95.

[4] Purwarianti, A., Tsuchiya, M., et Nakagawa, S., *Query transitive translation using IR score for indonesian-japanese CLIR.* Dans Proceedings of the Second Asia conference on Asia Information Retrieval Technology, 2005, p. 565-570.

[5] Kadri, Y., *Recherche d'Information Translinguistique sur les Documents en Arabe.* Thése de Phd, Université de Montréal, Québec, Canada, 2008, p. 165

[6] Kadri, Y. et Nie, J.-Y., *Combining resources with confidence measures for cross language information retrieval.* Dans Proceedings of the ACM workshop in CIKM, 2007, p. 131-137.

[7] Fautsch, C. et Savoy, J., *Evaluation de diverses stratégies de désambiguîsation lexicale.* Dans Proceedings of the 6th French Information Retrieval Conference of CORIA, 2009, p. 19-31.

[8] Baziz, M., Boughanem, M., et Nassr, N., *La recherche d'information multilingue : désambiguïsation et expansion de requêtes basées sur WordNet (regular paper).* Dans International Symposium On Programming and Systems (ISPS), Alger, 2003, p. 175-186.

[9] Saravanan, K., Udupa, R., et Kumaran, A., *Crosslingual Information Retrieval System Enhanced with Transliteration Generation and Mining.* Fire 2010, Microsoft Research India Bangalore, India, 2010, p. 1-8.

[10] Su, C.-Y., Lin, T.-C., et Wu, S.-H., *Using Wikipedia to translate OOV terms on MLIR.* Dans Proceedings of NTCIR-6 Workshop Meeting, Tokyo, Japan, 2007, p. 109-115.

[11] AbdulJaleel, N. et Larkey, L.S., *Statistical transliteration for english-arabic cross language information retrieval.* Dans Proceedings of the twelfth international conference on Information and knowledge management, 2003, p. 139-146.

[12] Nguyen, D. et al., *WikiTranslate: query translation for cross-lingual information retrieval using only Wikipedia.* Dans Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access, 2009, p. 58-65.

[13] Peters, C., Braschler, M. et Clough, P., *Multilingual Information Retreival: From Research To Practice.* Springer-Verlag Berlin Heidelberg, 2012, p. 232.

[14] Sadat, F., *Notes du cours psycholinguistique et traitement des langues naturelles: Modèles de recherche d'information.* Université de Québec à Montréal, 2013, p. 1-8.

[15] Lashkari, A.H., Mahdavi, F., et Ghomi, V., *A Boolean Model in Information Retrieval for Search Engines.* Dans Proceedings of International Conference on Information Management and Engineering, 2009, p. 385-389.

[16] Nie, J.-Y., *Modèles plus avancés: Modèle probabiliste.* [cited 2014 27 Mai 2014]; Available from: http://www.iro.umontreal.ca/~nie/IFT6255/Modeles_Probabilistes.html.

[17] Boughanem, M., Kraaij, W., et Nie, J.-Y., *Modèles de langue pour la recherche d'information* [cited 2014 27 Mai 2014]; Available from: http://www.iro.umontreal.ca/~nie/IFT6255/modele_langue.pdf.

[18] Zhou, D. et al., *Translation techniques in cross-language information retrieval.* ACM Computing Surveys, 2012, 45(1), p. 1-44.

[19] Carl, M., *Introduction à la traduction guidée par l'exemple (Traduction par analogie).* TALN, Batz-sur-Mer, 2003.

[20] Yen, C., *Évaluation de la production de quatre systèmes traduction automatique*. Master of Arts, Dalhousie University Halifax, Nova Scotia, 2013, p. 156.

[21] Habash, N., Rambow, O., et Roth, R., *MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization.* Dans Proceedings of the Second International Conference on Arabic Language Resources and Tools, 2009, p. 22-23.

[22] Gaillard, B., Collin, O., et Boualem, M., *Traduction de requêtes basée sur Wikipédia.* Dans Proceedings of TALN, 2010.

[23] Collin, O., Gaillard, B., et Bouraoui, J.-L., *Constitution d'une ressource sémantique issue du treillis des catégories de Wikipédia.* Dans Proceedings of TALN, 2010.

[24] Sadat, F. et Terrasa, A., *Exploitation de Wikipédia pour l'Enrichissement et la Construction des Ressources Linguistiques.* 17e Conférence sur le traitement automatique des langues naturelles, 2010.

[25] Yao, J.-M. et al., *Study on Wikipedia for translation mining for CLIR.* Dans Proceeding of the Ninth International Conference on Machine Learning and Cybernetics (ICMLC), 2010, p. 3374 -3379.

[26] Tufis, D. et al., *Wikipedia as an SMT Training Corpus.* Dans Proceedings of Recent Advances in Natural Language Processing, Hissar, Bulgaria, 2013, p. 702–709.

[27] Kadri, Y. et Nie, J.-Y., *Effective Stemming for Arabic Information Retrieval.* The Challenge of Arabic for NLP/MT, International Conference at the British Computer Society (BCS), 2006, p. 68-74.

[28] McCandless, M., Hatcher, E., et Gospodnetic, O., *Lucene in action*, M. Greenwich, Editor. Manning Publications, 2010, p. 528.

[29] Sellami, R., Sadat, F., et Belguith, L.H., *Extraction de lexiques bilingues à partir de Wikipédia.* 19e Conférence sur le traitement automatique des langues naturelles, JEP-TALN RECITAL, 2012.

[30] Sadat, F., *Notes du cours psycholinguistique et traitement des langues naturelles: Introduction à la Traduction Automatique (TA)*. Université de Québec à Montréal, 2013, p. 1-51.

[31] Sadat, F., *Notes du cours psycholinguistique et traitement des langues naturelles: Introduction au TALN - Traitement Automatique du Langage Naturel* -. Université de Québec à Montréal, 2013, p. 1-60.

[32] Rebout, L., *L'extraction de phrases en relation de traduction dans Wikipédia.* Mémoire de maitrise, Faculté des arts et des sciences, Université de Montréal, 2012, p. 90.

[33] Nakayama, K., Hara, T., et Nishio, S., *A Thesaurus Construction Method from Large ScaleWeb Dictionaries.* 21st International Conference on Advanced Networking and Applications(AINA'07), 2007, p. 109-115.

[34] Sadat, F., *Notes du cours psycholinguistique et traitement des langues naturelles: Introduction à la Recherche d'Information (RI).* Université de Québec à Montréal, 2013, p. 1-63.

[35] Brkić, M., Vičić, T., et Seljan, S., *Evaluation of the Statistical Machine Translation Service for Croatian-English.* INFuture2009: Digital Resources and Knowledge Sharing, 2009, p. 319-332.

[36] Crego, J.M. et al., *Micro-adaptation lexicale en traduction automatique statistique.* TAL, 2010, 51 (2), p. 65 à 93.

[37] Guyon, A., *Abrégé de traduction automatique.* 2014, Available from: http://www.btb.termiumplus.gc.ca.proxy.bibliotheques.uqam.ca:2048/tpv2guides/guides/chroniq/index-fra.html?lang=fra&lettr=indx_autr8o7vhUcC4c0s&page=9i-_Sdw_Ak98.html.

[38] Kit, C., Pan, H. et Webster J.J., *Example-Based Machine Translation: A New Paradigm.* Translation and Information Technology, Chinese U of HK Press, 2002, p. 57-78.

[39] Peng, L., *A Survey of Machine Translation Methods.* TELKOMNIKA, 2013, 11(12), p. 7125 – 7130.

[40] Dilekh, T., *Implémentation d'un outil d'indexation et de recherche des textes en arabe*, in *Faculté des Sciences*. Université Hadj Lakhdar – Batna, 2011, p. 98.

[41] Ghoul, D., *Outils génériques pour l'étiquetage morphosyntaxique de la langue arabe : segmentation et corpus d'entraînement.* Mémoire de maitrise, UFR Sciences du Langage, Université Stendhal Grenoble, 2011, p. 98.

[42] Attia, M., et al., *Automatic Extraction of Arabic Multiword Expressions.* Proceedings of the Workshop on Multiword Expressions: from Theory to Applications (MWE), 2010, p. 18–26.

[43] Sadat, F., *Extracting the Multilingual Terminology from a Webbased Encyclopedia.* Défis de la recherche en sciences de l'information (ERIC), Cinquième conférence internationale, 2011, p. 1-5.

[44] Rahimi, Z., et Shakery, A., *Creating a Wikipedia-based Persian-English Word Association Dictionary.* 5th International Symposium on Telecommunications (IST), 2010, p. 562-567.

[45] Erdmann, M. et al., *Improving the Extraction of Bilingual Terminology from Wikipedia.* ACM Transactions on Multimedia Computing, Communications and Applications, 2009, 5 (4), p. 31:1-31:17.

[46] Alqudsi, A., Omar, N., et Shaker, K., *Arabic machine translation: a survey.* Springer Science+Business Media B.V., 2012, p. 1-24.

[47] Hebresha, H.A. et Aziz, M.J.A., *Classical Arabic English Machine Translation Using Rule-based Approach.* Journal of Applied Sciences, 2013, 13, p. 79-86.

[48] Ayed, R. et al., *Evaluation d'une approche de classification possibiliste pour la désambiguïsation des textes arabes.* 21ème Traitement Automatique des Langues Naturelles, Marseille, 2014, p. 316-327.

[49] Hamdi, A., *Apport de la diacritisation dans l'analyse morphosyntaxique de l'arabe.* JEP-TALN-RECITAL, 2012, 3, p. 247–254.

[50] Mars, M., Antoniadis, G., et Zrigui, M., *Nouvelles ressources et nouvelles pratiques pedagogiques avec les outils tal.* TICEMED 08, Journal Information Sciences for Decision Making (Journal ISDM), ISDM32, 2008, p. 1-9.

[51] Shaalan, K., *A Survey of Arabic Named Entity Recognition and Classification.* Computational Linguistics, 2014, 40(2), p. 469-510.

[52] Tait, J.I., *(Ed.), Charting a New Course: Natural Language Processing and Information Retrieval..* Springer Pays-Bas (Essays in Honour of Karen Spärck Jones), 2005, 16, p. 283.

[53] Xu, Y., Jones, G.J.F., et Wang, B., *Query Dependent Pseudo-relevance Feedback Based on Wikipedia.* Dans Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2009, p. 59-66.

[54] Ludivine, M., *Moteur de recherche et d'indexation.* Université Charles De Gaulle-Lille 3, 2008, p. 1-33.

[55] Zbib, R. et al., *Methods for integrating rule-based and statistical systems for Arabic to English machine translation.* Machine Translation, 2012, 26, p. 67–83.

[56] Déjean, H., et Gaussier, É., *Une nouvelle approche à l'extraction de lexiques bilingues à partir de corpus comparables.* Lexicometrica, Alignement lexical dans les corpus multilingues, 2002, p. 1-22.

[57] Brown, P. et al., *A statistical approach to machine translation.* Computational Linguistics, 1990, 16(2), p. 79-85.

[58] Sellami, R., Sadat, F., et Belguith, L.H., *Traduction Automatique Statistique à partir de corpus comparables : Application au couple de langues arabe-français.* Dans Proceedings of CORIA, 2013, p. 431-440.