MINISTERE DE L'ENSEIGNEMENT
SUPERIEUR ET DE LA RECHERCHE
SCIENTIFIQUE
UNIVERSITE SAAD DAHLAB - BLIDA 1
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

MEMOIRE DE MASTER

EN INFORMATIQUE

Spécialité **:** Traitement Automatique de la Langue

CONCEPTION ET REALISATION D'UN SYSTEME DE TRADUCTION DE CONTEXTE DE ET VERS LA LANGUE ARABE

Présenté par :

NADJEMI Abdelouahab et ZENATI Mourad

Encadré par :

Mme : N.BENBLIDIA	Promoteur
M. ABBAS Mourad	Encadreur
M. LICHOURI Mohamed	Co-Encadreur

Jury :

Mme L.OUAHRANI	Présidente
Mme. H.DAOUD	Examinatrice

2020-2021

Résumé

Ce mémoire a pour objectif de proposer un système capable de réaliser une traduction de contexte de et vers la langue arabe. Ce système est basé sur la Traduction Automatique Statistiques, et est considéré comme un moteur de recherche qui cherche dans un grand ensemble de données appelé « corpus parallèle ». Les corpus que nous avons utilisés sont des ensembles de paires de langues, chaque paire contenant l'arabe et une autre langue. Dans notre cas, nous avons concentré sur la paire de langues arabe-anglais et nous nous sommes appuyés sur deux approches de recherche : la première utilisant ElasticSearch et la seconde utilisant Tf-Idf de Spark. Nous avons obtenu une traduction contextuelle avec des résultats plutôt satisfaisants, malgré les difficultés que nous avons rencontrées, notamment avec l'arabe, qui présente des particularités différentes des autres langues.

Mots clés : Traduction de contexte, moteur de recherche, langue Arabe, corpus parallèle, ElasticSearch, Spark TF-IDF.

Abstract

The objective of this not is to propose a system capable of carrying out a context translation from and into the Arabic language. This system is based on statistical machine translation, and is considered as a search engine that searches a large data set called a “parallel corpus”. The corpora we used are sets of language pairs, each pair containing Arabic and another language. In our case, we focused on the Arabic-English language pair and we relied on two search approaches: the first using ElasticSearch and the second using Spark's Tf-Idf. We obtained a contextual translation with rather satisfactory results, despite the difficulties we encountered, especially with Arabic, which has different particularities from other languages.

Keywords: Context translation, search engine, Arabic language, parallel corpus, ElasticSearch, Spark TF-IDF.

ملخص

تهدف هذه المذكرة إلى اقتراح نظام قادر على تنفيذ ترجمة بالسياق من وإلى اللغة العربية. هذا النظام يعتمد على الترجمة الآلية الإحصائية، ويعتبر بمثابة محرك بحث يبحث في مجموعات كبيرة من البيانات تسمى "المجموعة المتوازية". المجموعات التي استخدمناها هي مجموعات أزواج من اللغات، كل زوج يحتوي على اللغة العربية ولغة أخر.في حالتنا، ركزنا على الزوج اللغوي العربي-الإنجليزي واعتمدنا على طريقتين للبحث : الأولى بإستخدام ElasticSearch والثانية باستخدام Tf-Idf لـ Spark. تحصلنا إلى حد ما، على نتائج مرضية للترجمة بالسياق على الرغم من الصعوبات التي واجهناها، لا سيما مع اللغة العربية، التي تتميز بخصائص مختلفة عن اللغات الأخرى.

الكلمات المفتاحية: ترجمة بالسياق، محرك بحث، اللغة العربية، النص المتوازي، ElasticSearch، Spark's TF-IDF.

Dédicaces

Je dédie ce travail

À mes chers parents,

À ma chère famille,

Je vous souhaite tous le bonheur d'El Firdaous

Abdelouaheb

Dédicaces

Je dédie ce travail

À ma mère Nouna ellah yarhameha, à mon père Tayeb,

À ma chère épouse et mes chers enfants : Salsabil, Souhaieb, Oussama et ma petite Nour El Houda

À mes frères.

À mes sœurs.

À tous mes amis.

Je vous souhaite tout le bonheur du monde

Mourad

Remerciements

A l'issue de ce travail, nous remercions, en premier lieu, le bon **Dieu** de nous avoir donné la force et le courage de le mener à terme.

Nous adressons notre gratitude et nos sincères remerciements à notre encadreur Dr. **ABBAS Mourad**, directeur de recherche au niveau du Centre de Recherche Scientifique et Technique pour le Développement de la Langue Arabe (CRSTDLA), qui nous a ouvert ses portes durant la période du stage pratique. Merci de nous avoir encadrés, éclairés, si bien orientés et surtout d'avoir veillé à l'aboutissement de notre projet.

Nous tenons à exprimer nos sincères remerciements à notre Promoteur Pr. **N.BENBLIDIA**, notre Co-encadreur Dr. **M.LICHOURI** et le Dr. **O.Nadjemi** pour leurs conseils et intérêts incontestables.

Dans l'impossibilité de citer tous les noms, nos sincères remerciements vont à tous ceux et celles, qui ont contribué par leurs conseils au bon aboutissement de ce travail.

Nous remercions les membres du jury de leur attention et intérêt portés envers notre travail. Merci de nous avoir honorés par votre présence.

Enfin, nous n'oserions oublier de remercier tous les enseignants du TAL pour le travail énorme qu'ils effectuent afin de nous créer les conditions favorables du bon déroulement de nos études, notamment pendant la pandémie de COVID-19.

Table des matières

Liste des figures

Liste des Tableaux

Liste des Acronymes

API	Application Programming Interface
ASM	Arabe Standard Moderne
CAT	Computer-Aided Translation
CSV	Coma Separated Values
ELK	Elasticserch, Logstash, Kibana
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IDF	Inverse Document Frequency
JSON	JavaScript Object Notation
NoSQL	Not only SQL
OPUS	Open Parallel corpUS
LSI	Latent Semantic Indexing
SQL	Structured Query Language
SRI	système de recherche d'information
TAL	Traitement Automatique de Langue
TALN	Traitement Automatique du Langage Naturel
TA	Traduction Automatique
TAS	Traduction Automatique Statistique
TMX	Translation Memory Exchange
TF	Term Frequency
XML	eXtensible Markup Language

Introduction Générale

Introduction générale

Il est difficile d'ignorer la prévalence croissante de la traduction automatique (TA), car la technologie dans l'industrie de la traduction est en constante optimisation. Il faut également admettre que la traduction automatique est désormais vitale pour le secteur de la traduction et représente une grande partie de son avenir. Cela est logique compte tenu des innombrables traductions qui doivent être effectuées quotidiennement au niveau commercial, mais quelles sont les limites de la traduction automatique ?

La technologie d'aujourd'hui, malgré ses progrès remarquables, est toujours incapable de résoudre de nombreux problèmes que seul un traducteur humain professionnel peut les résoudre - les services de traduction automatique sur Internet peuvent ressembler davantage à des dictionnaires qu'à des traducteurs. Cela peut signifier qu'il est imprudent de se fier uniquement à eux, car une intuition innée de la langue est perdue, ce qui peut être particulièrement vrai dans un environnement commercial mondial, par exemple lors de la traduction d'articles tels que des CV, des contrats et des documents commerciaux.

Pour cela, passer à la réalisation d'un système de traduction de contexte pourrait résoudre ce problème.

Problématique

Les systèmes de traductions automatiques actuels ont plusieurs limites, dont l'une est l'absence de prise en charge du contexte par les systèmes de traduction automatique, ce qui peut conduire à la création de phrases inintelligibles en permanence. De plus, les outils de traduction automatique produisent très fréquemment des erreurs dans des domaines spécialisés qui impliquent une rédaction technique dans certaines professions. Il est très fréquent que le mot en question ne soit pas reconnu, ou qu'une traduction très approximative soit faite, ce qui pourrait produire un sens complètement différent du texte.

Afin de résoudre ce type de problème, on a vu l'apparition de deux services de traduction automatique de contexte : Reverso Context et Linguee. Ces deux services permettent aux utilisateurs d'obtenir toutes les traductions possibles du texte introduit selon plusieurs contextes. La limite actuelle de ces deux services est qu'ils ne contiennent pas beaucoup de paires de traduction de et vers l'Arabe. C'est pourquoi l'introduction de ce projet.

Objectifs du travail

L'objectif de ce travail est de réaliser un système de traduction automatique de contexte comparable aux performances de Reverso et Linguee, mais dédié à la traduction de et vers la langue Arabe. Ce système est basé sur la Traduction Automatique Statistiques, et est considéré comme un moteur de recherche, qui cherche dans un grand ensemble de données appelé « corpus parallèle ».

Le présent mémoire est organisé comme suit :

- ✓ Le chapitre 1, pour donner des notions sur la traduction et traiter les principaux problèmes que l'on doit résoudre lors du développement d'un système de traduction automatique, ainsi que l'évolution avec les différentes approches possibles et les principales tendances observées dans ce système ;
- ✓ Le chapitre 2, pour donner un aperçu général sur le Traitement Automatique du Langage Naturel, quelques particularités et caractéristiques propres de la langue arabe ;
- ✓ Le chapitre 3, pour présenter les mécanismes de la Recherche d'Information, les concepts clés et les modèles de base sur lesquels repose la recherche ;
- ✓ Le chapitre 4, pour se consacrer à la réalisation du système de traduction de contexte étudié ;
- ✓ Le chapitre 5, pour se pencher sur les différents aspects relatifs à mise en œuvre de ce système ;
- ✓ La conclusion générale pour terminer le mémoire et inclure quelques recommandations et perspectives de recherche.

Chapitre I

Notion et évolution de la traduction automatique

Chapitre I :

Notion et évolution de la traduction automatique

1. Introduction

Avant de parler de traduction automatique et de traduction automatique statistique, il est important d'étudier la notion de traduction en elle-même, ses dimensions et quand on peut évaluer la traduction et dire qu'elle est bonne ou pas.

Au cours de ce chapitre, nous verrons que ces préoccupations sont difficiles à aborder et ont déjà donné lieu à une abondante littérature. Puis, nous examinerons pourquoi la compréhension d'une phrase est l'une des choses les plus difficiles à faire avec les ordinateurs, malgré leur incroyable puissance de calcul. Par la suite, nous allons introduire un bref historique et les différentes approches possibles et les principales tendances observées dans le domaine de la traduction automatique depuis ses débuts. Enfin une idée sur les principaux enjeux et les principales évolutions du domaine.

2. Que signifie traduire ?

La réponse à cette question peut paraître évidente : traduire, c'est transposer un texte en langue source dans un texte en langue cible. Cependant, on peut facilement voir que cette réponse d'une simplicité trompeuse se réfère en fait à un problème d'une complexité dramatique. Que signifie « transposer un texte » ? Comment passer d'une langue source à une langue cible ? Comment trouver des expressions équivalentes entre deux langues ? La traduction devrait-elle être basée sur des mots, des morceaux de mots ou même des phrases ? Et, plus fondamentalement, comment déterminer quelle est la signification d'un texte ou d'une expression ? Tout le monde a-t-il la même compréhension d'un texte ? Sinon, comment ce problème peut-il être traité dans le processus de traduction ?

Comme il ressort du paragraphe précédent, la traduction est liée à un grand nombre de questions traitant de la linguistique, mais aussi de la psychologie ou même de la philosophie lorsque la nature du sens est en jeu. Au lieu d'aborder ces questions très complexes (sans réponse claire !), il est probablement plus utile de faire un pas de côté et d'essayer de déterminer quelles sont les caractéristiques d'une « bonne » traduction.

3. Qu'est-ce qu'une bonne traduction ?

Une première question cruciale en ce qui concerne la traduction est que personne ne sait comment définir formellement ce qui constitue une « bonne » traduction. Il ne faut donc pas s'attendre à faire beaucoup de progrès dans cette perspective, mais au moins certains critères peuvent être trouvés dans la littérature [1].

3.1. Critères pour une « bonne » traduction

La traduction d'un texte doit être fidèle au texte original : elle doit respecter les principales caractéristiques du texte original, le ton et le style, les détails des idées ainsi que sa structure générale. Le résultat doit être facile à lire dans la langue cible, et il doit également être linguistiquement correct, ce qui signifie qu'un processus subtil de reformulation doit être utilisé. Idéalement, le lecteur ne devrait pas se rendre compte qu'il lit une traduction s'il ne connaît pas l'origine du texte, ce qui implique que toutes les expressions formulées et idiomatiques doivent être rendues de manière appropriée.

En conséquence, le traducteur doit parfaitement comprendre le texte qu'il doit traduire, mais il doit également avoir une connaissance encore meilleure de la langue cible. C'est la raison pour laquelle les traducteurs professionnels ne traduisent généralement que dans leur langue maternelle afin d'avoir une compréhension et une connaissance parfaites des expressions à utiliser pour rendre le texte source avec précision.

La subjectivité inhérente à ces critères est cependant indéniable. Ce qui est considéré comme une « bonne » traduction par certains lecteurs peut être mauvaise selon une autre personne. Cette situation se présente fréquemment lorsque des traducteurs professionnels travaillent avec des auteurs qu'ils ne connaissent pas ou lorsque le traducteur ne sait pas dans quel contexte sa traduction sera utilisée.

Ce que l'on attend d'une traduction peut varier radicalement en fonction des clients, de l'époque, de la nature du texte, de son usage, voire du contexte. Les textes techniques ne sont pas traduits de la même manière que les textes littéraires. Une adaptation spécifique du texte original est nécessaire lorsque le texte concerne un monde éloigné du monde du lecteur dans la langue cible [1].

Le traducteur doit choisir entre rester proche du texte original ou recourir à la paraphrase pour assurer la compréhension (en particulier avec des contextes historiques, des événements

inconnus, etc.). Le ton et le style d'un texte sont également des notions très subjectives largement liées à la langue considérée.

Comme on peut facilement le voir à partir de cet aperçu rapide, toutes ces caractéristiques subjectives font de l'évaluation de la tâche un problème difficile.

Certains écueils sont cependant bien connus et fréquemment abordés dans la littérature sur le sujet. La traduction mot à mot n'est pas une bonne pratique, car le résultat est souvent difficile à comprendre et non idiomatique dans la langue cible.

Les faux-amis (deceptive cognate) et les doublons syntaxiques doivent être bannis car ils conduisent au non-sens (par exemple, le mot français « achèvement » doit être traduit par «completion» en anglais, et non par «achievement») [1]. Il est également bien connu qu'un traducteur doit d'abord lire tout le texte, ou au moins une grande partie du texte, à traduire afin d'éviter les erreurs de traduction locales. Une bonne connaissance des clients, du contexte et de l'utilisation future du texte traduit peuvent également aider à ajuster la traduction à la cible.

3.2. Conséquences pour la traduction automatique

Il est clair que la traduction est un processus complexe impliquant des capacités cognitives et linguistiques de haut niveau. Un traducteur doit être à l'aise avec les deux langues concernées, et il doit avoir des compétences particulières pour reformuler une langue source dans une langue cible qui n'a pas le même libellé ou la même structure.

Ces types de compétences ne sont pas directement accessibles aux machines. Les systèmes artificiels en sont encore à leurs balbutiements de ce point de vue et sont très éloignés des capacités d'un être humain en matière de raisonnement, de déduction et de reformulation. Pour pouvoir reformuler une phrase, il faut bien sûr avoir une bonne maîtrise de la langue elle-même, mais il faut aussi maîtriser la recherche d'analogie entre concepts, ce qui est beaucoup plus compliqué que juste des équivalences entre mots et expressions.

Les développeurs de systèmes artificiels sont conscients de ces limites. Très peu de chercheurs ont essayé de développer des systèmes de traduction automatique pour les textes littéraires : presque tout le monde convient que la traduction automatique est une tâche difficile qui est loin d'être résolue et que seuls les textes banals devraient être traités. L'idée n'est pas de remplacer les traducteurs humains qui sont les seuls capables de traduire des romans ou de la poésie. Même les textes techniques posent des difficultés spécifiques car ils emploient un vocabulaire très technique qui doit d'abord être introduit dans le système afin d'obtenir des

traductions pertinentes. Le but de la traduction automatique est désormais considéré principalement comme celui de fournir une aide à l'utilisateur et, dans certains contextes professionnels, de lui permettre de décider si un traducteur humain doit être appelé ou non [1].

La qualité globale réalisable par la traduction automatique a également fait l'objet de nombreux débats. Le but ultime est d'obtenir une qualité de traduction équivalente à celle d'un être humain. Les gens conviennent que c'est très difficile et aussi difficile à formaliser, car la qualité d'une traduction est liée à la nature et à la complexité du texte à traduire.

Pendant longtemps, la traduction automatique a utilisé des techniques locales qui pourraient être comparées, dans une certaine mesure, à un processus de traduction mot à mot, même si la plupart des systèmes prennent désormais également en considération des expressions plus complexes. Les informations au niveau du texte sont rarement prises en compte, même s'il est bien connu que le texte peut fournir des informations importantes pour le processus de traduction. La tonalité ou le style d'un texte, par exemple, est toujours ignoré : ce type d'information est en fait trop difficile à formaliser pour les systèmes automatiques.

D'une certaine manière, même le niveau de la phrase est trop complexe pour la plupart des systèmes actuels. On suppose généralement que ces systèmes effectuent une traduction phrase par phrase, ce qui est vrai dans une certaine mesure, mais le processus de traduction implique généralement, en fait, des fragments de phrases. La traduction d'une phrase complète consiste alors à assembler les traductions de ces fragments locaux. Il n'est donc pas surprenant que la traduction automatique donne parfois des résultats étranges et souvent des absurdités. La morphologie (l'analyse de la structure des mots) et la syntaxe (l'analyse de la structure des phrases) sont rarement prises en compte, ce qui a des conséquences particulièrement dramatiques pour certaines langues.

Par exemple, certains mots sont dits très flexionnels, ce qui signifie que leurs formes des mots peuvent changer en fonction de la fonction grammaticale du mot dans la phrase (sujet, complément, etc.). Dans ce contexte, il est clair qu'un processus automatique ne sera pas en mesure de fournir la bonne forme de mot dans la langue cible sans une analyse syntaxique appropriée.

Enfin, il faut comprendre pourquoi le traitement des langues avec des ordinateurs est difficile, même lorsqu'il s'agit de tâches plus faciles que la traduction automatique.

Une langue a des milliers de mots, avec différentes formes de surface ("to dance", « danced », « dancing »), des significations différentes et des structures différentes. Composés

(par exemple « round table », qui désigne généralement un événement et non un objet), des verbes légers (par exemple, « to take a shower », où « take » a peu de contenu sémantique), et des idiomes ou des expressions figées (par exemple, «kick the bucket», dont le sens n'a rien à voir avec «kick» ou «bucket») rend la tâche encore plus complexe, puisqu'il faut alors repérer des expressions complexes et pas seulement des mots isolés [1].

4. Difficultés d'analyser du langage naturel avec des ordinateurs

Outre le manque d'informations sur le client, le contexte ou le style du texte à traduire, le principal problème est lié à la tâche elle-même. Le traitement des langages naturels est difficile en soi, principalement parce qu'au cœur du langage naturel se trouvent l'imprécision et l'ambiguïté.

4.1. Langages naturels et l’ambiguïté

Les linguistes ainsi que les informaticiens s'intéressent depuis la création des ordinateurs au traitement du langage naturel, un domaine également appelé linguistique informatique. Le traitement du langage naturel est difficile car, par défaut, les ordinateurs n'ont aucune connaissance de ce qu'est une langue. Il est donc nécessaire de préciser la définition d'un mot, d'une phrase et d'une expression. Dans ce cas, les problèmes peuvent ne pas sembler trop difficiles et pas si différents des langages formels, qui sont également constitués de mots.

Cependant, pensez à des expressions comme : «isn’t», «won’t», «U.S.», «$80»: ce n'est pas toujours clair ce qu'est un mot et comment les mots sont impliqués dans de telles expressions. La principale différence réside dans le fait que chaque mot et chaque expression d'une langue naturelle donnée peuvent être ambigus [1].

Considérons des exemples célèbres tels que "the chicken is ready to eat" ou "there was not a single man at the party". Ce sont des exemples de manuels et peuvent sembler impossibles à réaliser. Toutefois, ils illustrent certains problèmes bien connus du traitement du langage : dans le premier exemple, faut-il donner à manger au poulet ou est-ce le poulet qui est prêt à être mangé ? Dans le deuxième exemple, est-ce que l'orateur veut dire qu'il n'y avait pas d'hommes à la fête ou veut dire que tous les hommes étaient mariés ? Ces exemples sophistiqués ne doivent pas masquer le fait que l'ambiguïté est en fait omniprésente et fait également partie des mots et expressions les plus banals. Rien que dans ces deux exemples, on peut remarquer que «chicken» peut désigner un animal ou une sorte de viande, mais aussi un lâche. « A party » peut désigner (selon WordNet) : « une organisation pour gagner le pouvoir politique », « un groupe de

personnes associées temporairement à une activité », « une occasion à laquelle les gens peuvent se rassembler pour interaction sociale et divertissement », voire « une personne impliquée dans une procédure judiciaire »). « Party » peut également être un verbe pour «avoir ou participer à une fête », etc. [1]

Une réponse à ce problème est juste d'enregistrer d'une certaine manière toutes ces différentes significations dans un dictionnaire, ce qui existe déjà comme nous l'avons mentionné. Par exemple, WordNet, constitue une base de données lexicale qui peut être utilisée aussi bien par les humains que par les ordinateurs, comme nous le verrons plus loin dans la partie « indexation sémantique », chapitre « Recherche d'Information ». Cependant, on se rend vite compte que ce n'est pas une solution de travail, puisqu'une fois toutes ces significations stockées dans le dictionnaire, il y a un problème de trouver un moyen de choisir le bon sens de chaque occurrence, c'est-à-dire pour chaque mot utilisé dans le contexte.

4.2. Systèmes artificiels et naturels

Une question très débattue dans le domaine de la traduction automatique est de savoir dans quelle mesure les systèmes artificiels devraient reproduire les stratégies utilisées par les humains pour traduire. En d'autres termes, pouvons-nous apprendre quelque chose en observant les pratiques de travail des traducteurs professionnels ?

C'est une autre question difficile, et la première chose à souligner est que nous ne savons pas grand-chose sur les processus cognitifs impliqués dans la tâche de traduction. De plus, les stratégies de traduction varient probablement largement d'un traducteur à l'autre. Il est clair que traduire nécessite d'aller au-delà de la simple approche mot pour mot, comme nous l'avons déjà souligné, mais on peut se demander si les traducteurs professionnels font systématiquement une analyse syntaxique et sémantique approfondie de la phrase à traduire. Il est clair, par exemple, que les interprètes professionnels (qui font des traductions sur place) traduisent souvent des groupes de mots semi-autonomes sans avoir entendu la phrase complète, en particulier dans le cas d'une phrase longue.

Cette approche peut être comparée à celle des systèmes statistiques qui n'effectuent pas une analyse approfondie de la phrase à traduire, mais identifient des groupes de mots qui fonctionnent ensemble. Le parallèle n'est pas tout à fait exact, car les interprètes sélectionnent toujours des groupes de mots relativement autonomes dans la phrase sélectionnent toujours des groupes de mots relativement autonomes dans la phrase, alors qu'une analyse statistique extraira tout groupe de mots régulier sans trop s'appuyer sur des contraintes syntaxiques.

Cependant, comme déjà dit, les systèmes statistiques sont très bons pour reconnaître les expressions multi-mots (composés, idiomes, etc.) qui sont perçues comme des unités uniques même par les humains, comme le montre la psychanalyse du langage.

Des expériences récentes ont renforcé ce point de vue, puisque même des structures syntaxiques de haut niveau peuvent correspondre à des modèles réguliers. Ces structures sont parfois appelées « constructions » (structures syntactico-sémantiques spécifiques inscrites comme telles dans notre cerveau) ou « préfabriquées » (comme une maison faite d'éléments préfabriqués qui peuvent être rapidement assemblés pour obtenir une construction modulaire).

Dans ce cadre, la syntaxe n'est pas aussi importante que dans les approches traditionnelles : la phrase est vue comme un assemblage « d'unités préfabriquées », ou, en d'autres termes, un assemblage de séquences complexes stockées comme telles dans le cerveau. L'analyse est donc plus simple, puisque, si cette hypothèse est correcte, le cerveau n'a pas vraiment à prendre en compte chaque mot individuel mais a un accès direct aux unités de niveau supérieur, réduisant à la fois l'ambiguïté globale et la complexité du processus de compréhension de la phrase [1].

4.3. Conséquences pour la traduction automatique

Nous avons vu dans ce chapitre que le principal problème du traitement du langage naturel est l'ambiguïté : il est simplement difficile de déterminer le sens d'un mot. Le sens dépend du contexte, mais la notion de contexte est elle-même aussi vague et ambiguë.

Il faut ajouter que la détermination du nombre de sens par mot est également une question ouverte, car d'un dictionnaire à l'autre, le nombre de sens des mots diffère : certains dictionnaires sont plus précis et contiennent une description plus fine du sens, tandis que d'autres préfèrent limiter le nombre de sens par mot, en fonction de leurs choix conceptuels et de leur lectorat visé.

Malgré ces problèmes, il est généralement admis que pour produire des traductions de haute qualité, il faut d'abord fournir une description précise et exacte de la signification des phrases. Les progrès de la traduction automatique ont alors été liés aux progrès réalisés dans la compréhension du texte, qui ont largement conduit le domaine pendant plusieurs années. Cependant, ces hypothèses sont remises en question : les approches statistiques peuvent utiliser des quantités de texte disponibles sur le Web et calculer les équivalences possibles entre les langues sans utiliser de dictionnaires prédéfinis ou de formalismes de haut niveau.

5. Traduction automatique

5.1. Bref aperçu historique

L'histoire de la traduction automatique peut être résumée comme suit :

- ✓ Jusque dans les années 1940, certains chercheurs se sont penchés sur le problème de l'automatisation de la traduction, mais ces études pionnières n'ont pas été reprises car il n'était pas possible de les développer et de les tester en pratique.
- ✓ De 1940 au milieu des années 1960, avec l'avènement des premiers ordinateurs, plusieurs équipes ont développé des systèmes opérationnels de traduction automatique. Les attentes concernant le développement du domaine étaient élevées. Les approches utilisées étaient parfois assez naïves, mais certains chercheurs ont déjà imaginé des approches plus substantielles, dont certaines ont été fondamentales et redécouvertes quelques années plus tard.
- ✓ En 1965-1966, le rapport ALPAC[1], commandé par les agences de financement américaines, eut un impact dramatique sur le terrain. Les conclusions du rapport étaient très négatives, qualifiant les recherches effectuées jusqu'alors de viciées et inutiles. Le financement qui diminuait déjà considérablement s'est progressivement évanoui. Cependant, il faut noter que le rapport a également souligné la nécessité de développer plus de recherche fondamentale combinant informatique et linguistique, afin de permettre au domaine de progresser davantage dans la compréhension du texte (et plus fondamentalement dans l'analyse syntaxique et l'analyse sémantique automatique).
- ✓ Les années suivantes, jusqu'à la fin des années 80, n'ont pas été très productives pour la traduction automatique, en particulier dans le monde anglo-américain. De nouveaux groupes ont néanmoins émergé en Europe et dans d'autres pays. En revanche, les recherches en linguistique computationnelle fleurissaient à la même période pour la parole comme pour le texte écrit : les années 1960 et 1970 ont vu des développements majeurs dans l'analyse syntaxique (parsing), la sémantique et la compréhension du texte à titre d'exemple, comme suggéré dans le rapport ALPAC de 1966.
- ✓ Les années 1990 ont vu l'avènement d'une nouvelle approche basée sur les statistiques et de très grands corpus bilingues. Cette tendance découle clairement d'une série d'articles fondateurs rédigés par un groupe de recherche travaillant chez IBM à la fin des années 80 et au début des années 90. Ces articles ont eu un impact considérable, ainsi que le

[1]ALPAC: Automatic Language Processing Advisory Committee.

développement d'approches statistiques et empiriques dans le traitement du langage naturel. Les systèmes de traduction les plus populaires de nos jours (traduction Google, Bing, Reverso) reposent tous sur une variante de cette approche.

Récemment, la très forte demande de traduction automatique sur le web a également eu pour effet de remettre la traduction automatique au cœur du domaine de la linguistique computationnelle, après plusieurs décennies au purgatoire. Une nouvelle approche basée sur le Deep Learning est également en train de révolutionner complètement le domaine depuis le milieu des années 2010 [1].

5.2. Approches de traduction automatiques

Différentes approches et différentes techniques ont été utilisées pour la traduction automatique. Par exemple, la traduction peut être directe, d'une langue à l'autre (c'est-à-dire sans représentation intermédiaire), ou indirecte, lorsqu'un système essaie d'abord de déterminer une représentation plus abstraite du contenu à traduire.

Cette représentation intermédiaire peut également être indépendante de la langue afin de permettre de traduire directement un texte source dans différentes langues cibles.

Chaque système est unique et met en œuvre une approche plus ou moins originale du problème. Cependant, dans un souci de clarté et de simplicité, les différentes approches peuvent être regroupées en trois catégories différentes, comme le font la plupart des manuels sur le sujet.

5.2.1. Systèmes de traduction directe : sont des systèmes qui essaient de produire une traduction directement d'une langue source vers une langue cible sans représentation intermédiaire. Ces systèmes sont généralement basés sur des dictionnaires : un dictionnaire fournit une traduction mot à mot, puis des règles plus ou moins sophistiquées tentent de réordonner les mots cibles de manière à obtenir un ordre des mots aussi proche que possible de ce qui est requis par la langue cible. Il n'y a pas d'analyse syntaxique dans ce type de système et les règles de réorganisation s'appliquent directement aux formes de surface.

5.2.2. Systèmes de transfert : sont plus complexes que les systèmes de traduction directe, car ils intègrent une sorte d'analyse syntaxique. Le processus de traduction est alors capable d'exploiter la structure de la phrase source fournie par le composant d'analyse syntaxique, évitant la limitation mot à mot de la traduction directe. Le résultat est donc supposé être plus idiomatique qu'avec la traduction directe, à condition que la composante syntaxique fournisse des informations précises sur la source et sur la langue cible.

5.2.3. Systèmes interlinguaux : sont les plus ambitieux, qui sont des représentations plus ou moins formelles du contenu à traduire. Des recherches approfondies ont été menées sur la notion d'interlingua. Des questions fondamentales se sont immédiatement posées telles que : à quel point l'interlingua doit-il être profond et précis pour fournir une représentation solide de la phrase à traduire ? Au lieu de développer une langue complètement artificielle, qui est connue pour être une tâche très complexe, l'anglais est souvent utilisé comme interlingua, mais c'est en fait assez trompeur, car la représentation n'est alors ni formelle ni indépendante de la langue. Il vaut donc mieux parler de « langue pivot », ou simplement de «pivot», lorsque l'interlingua est une langue naturelle spécifique. Dans ce contexte, lors de la traduction de la langue 'A' vers la langue 'B', le système essaie d'abord de transférer le contenu de 'A' vers la langue pivot avant de procéder à la traduction du pivot vers la langue cible 'B'.

Ces trois types d'approches peuvent être considérés comme formant un continuum, allant d'une stratégie très proche de la surface du texte (une traduction mot à mot) jusqu'à des systèmes essayant de développer une représentation entièrement artificielle et abstraite indépendante de tout Langue. Ces différentes stratégies ont été résumées dans une figure très frappante appelée « le triangle de Vauquois », du nom d'un célèbre chercheur français en traduction automatique dans les années 1960 (figure 1).

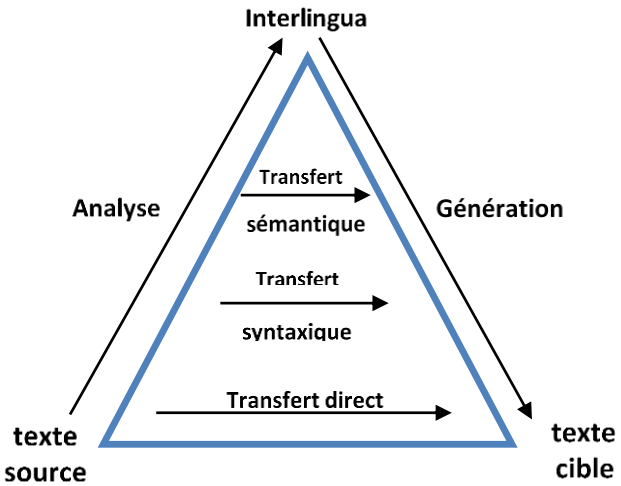


Figure 1 : Triangle de Vauquois, modèle pour les fondements de la traduction automatique[2]

Dès le début, les chercheurs ont également tenté dès le départ, de développer des stratégies plus sophistiquées pour prendre en compte la structure des langues en jeu. La notion de « règles de transfert » est apparue dans les années 1950 : pour passer d'une langue source à

[2]https://fr.wikipedia.org/wiki/Fichier:Triangle_de_vauquois.svg

une langue cible, il faut avoir des informations sur la manière de traduire des groupes de mots qui forment une unité linguistique (un idiome ou même une phrase). La structure des phrases est trop variable pour être prise en compte directement dans son ensemble, mais les phrases peuvent être divisées en fragments (ou blocs) qui peuvent être traduits à l'aide de règles spécifiques. Par exemple, les adjectifs en français sont généralement placés après le nom, alors qu'ils sont avant le nom en anglais. Cela peut être spécifié à l'aide des règles de transfert.

6. Révolution des systèmes de Traduction Automatique Statistique

La classification des systèmes de traduction automatique est remise en question par les nouvelles approches apparues depuis le début des années 1990. La disponibilité d'énormes quantités de texte, notamment sur Internet, et le développement de la capacité des ordinateurs ont révolutionné le domaine.

La plupart des systèmes de traduction automatique industriels actuels, et notamment les plus courants (traduction Google, traduction Bing, Reverso, etc.), reposent sur une approche statistique qui ne rentre pas complètement dans la classification précédente. Ces systèmes ne sont pas principalement basés sur de grands dictionnaires bilingues et des ensembles de règles artisanales. Les premiers systèmes statistiques ont mis en œuvre une sorte d'approche de traduction directe, puisqu'ils ont tenté de trouver des équivalences de mots entre deux langues différentes en regardant directement de très grandes quantités de données bilingues, provenant principalement d'institutions gouvernementales, comme le parlement canadien Hansard ou Européen Europarl, ou de l'Organisation des Nations Unies (ONU) et, plus récemment, pour la plupart, récoltées sur le web (ex. projet OPUS, que nous avons utilisé dans notre travail). Ces données représentent sous ce que l'on appelle *bitexte* ou un corpus bilingue parallèle (ou corpus parallèle) où les liens de traduction entre les phrases sont explicites.

Les approches statistiques sont désormais beaucoup plus précises. Ils ne traitent plus de mots isolés mais sont désormais capables de repérer des séquences de mots (tels que des composés, des idiomes, des expressions figées ou simplement des séquences régulières de plusieurs mots) qui doivent être traduites dans leur ensemble.

Les approches les plus récentes tentent même de s'attaquer au problème directement au niveau de la phrase. Il est à noter que ces systèmes ont leurs propres représentations internes qui ne sont généralement pas directement lisibles par un être humain.

Le succès de ces systèmes réside dans le fait qu'ils sont capables de saisir des équivalences régulières entre les langues, mais aussi des séquences plus ou moins figées dans

un texte, sur la base d'une analyse purement statistique. Puisque, comme nous l'avons vu, le sens n'est pas quelque chose de formellement défini mais correspond à la manière dont les mots sont utilisés, une approche purement statistique peut être assez puissante pour découvrir des régularités et des contextes d'utilisation spécifiques.

7. Conclusion

Nous avons vu dans ce chapitre des notions relatives à la traduction, les critères pour une bonne traduction et leurs conséquences sur la traduction automatique. Nous avons présenté les difficultés rencontrées pour analyser le langage naturel avec des ordinateurs et dans quelle mesure les systèmes artificiels devraient reproduire les stratégies utilisées par les humains pour traduire. Nous avons présenté un bref historique de la traduction automatique, ensuite on a mis l’accent sur les différentes architectures linguistiques de la traduction. Nous conclurons ce premier chapitre par la révolution des systèmes de traduction automatique statistique (TAS).

Chapitre II

Traitement Automatique du Langage Naturel « TALN » et la langue arabe

Chapitre II :

Traitement Automatique du Langage Naturel et la langue Arabe

1. Introduction

Aujourd'hui, un volume de données de plus en plus important sur la toile est composé, en très grande partie, de données textuelles qui peuvent être analysées et exploitées dans différents buts. L'informatique a permis le développement d'outils pour traiter l'information et établir des solutions. Parmi les outils et techniques informatiques liées à ce domaine, on trouve le Traitement Automatique du Langage Naturel (TALN).

Certaines langues ont été privilégiées comme l'anglais et le français, où les recherches sont centralisées pour donner des outils pour des applications TALN. D'autres, telle que la langue Arabe, poursuivent les recherches et les travaux afin de proposer des outils robustes de traitement permettant de proposer des applications TALN pour ces langues.

Ce chapitre présente un aperçu général sur le Traitement Automatique du Langage Naturel ainsi que les particularités et les problèmes rencontrés lors du traitement de la langue Arabe.

2. Traitement Automatique du Langage Naturel

2.1. Définition du TALN

Définition 1 : Le Traitement Automatique du Langage Naturel (TALN) désigne l'ensemble de recherches et méthodes qui visent à permettre à une machine de comprendre automatiquement le langage humain [2]. Le TALN a pour objectif de comprendre le sens des phrases, les idées qui s'en dégagent et ce, de manière la plus optimale et la plus naturelle d'un point de vue humain, et de fournir une réponse adaptée sans qu'aucune intervention extérieure ne soit nécessaire.

Définition 2 : C'est le sous-domaine de l'informatique, en particulier de l'intelligence artificielle (IA), qui vise à permettre aux ordinateurs de comprendre et de traiter le langage

humain. Techniquement, le TALN aurait pour tâche principale de programmer des ordinateurs pour l'analyse et le traitement d'une énorme quantité de données en langage naturel[3].

2.2. Bref historique du TALN

Le traitement automatique des langues naturelles est né à la fin des années quarante du siècle dernier, dans un contexte scientifique et politique très précis [3] [4] [5].

- ✓ Entre 1951 et 1954 : Zelling Harris[4] publie ses travaux les plus importants de la linguistique (linguistique distributionnaliste) ;
- ✓ 1954 : La mise au point du premier traducteur automatique (très rudimentaire) quelques phrases russes, sélectionnées à l'avance, furent traduites automatiquement en anglais ;
- ✓ 1956 : L'école d'été de Dartmouth, voit la naissance de l'intelligence artificielle ;
- ✓ 1957 : N. Chomsky[5] publie ses premiers travaux importants sur la syntaxe des langues naturelles, et sur les relations entre grammaires formelles et grammaires naturelles ;
- ✓ 1962 : Première conférence sur la traduction (Bar-Hillel[6]) rapport ALPAC ;
- ✓ 1966 : Le système ELIZA[7] (Weizenbaum) ;
- ✓ 1968 : Le premier (vrai) système de traduction (Systran, russe → anglais) ;
- ✓ 1971 : Un système intelligent en mode fermé (SHRDLU[8]) ;
- ✓ 1976 : Le système de traduction METEO mis au point à l'Université de Montréal ;
- ✓ 80s : Système de reconnaissance statistique multi locuteur ;
- ✓ 90s : Premiers corpus, approches statistiques en apprentissage automatique. Applications utilisent des corpus de grande taille et méthodes statistiques ;
- ✓ 2000s : Utilisation du World Wide Web comme corpus.

2.3. Niveaux du TALN

Le processus du traitement automatique des données linguistiques nécessite différents niveaux d'analyses. On parle dans la littérature d'analyse morphologique, d'analyse syntaxique, d'analyse sémantique et d'analyse pragmatique (figure 2). Dans ce qui suit nous allons décrire brièvement les différents niveaux d'analyse d'un texte en langue naturelle :

[3] Support de cours : Dr. M.MEZZI, Module : Syntaxe, Lexique et Sémantique, Université de Blida, 2021.
[4] Zellig Sabbetai Harris : linguiste américain.
[5] Noam Chomsky : linguiste américain.
[6] Yehoshua Bar-Hillel : linguiste et chercheur dans la traduction automatique.
[7] Un programme qui simule un entretien avec un psychiatre.
[8] Un programme qui permet d'interagir avec un robot dans un monde de blocs.

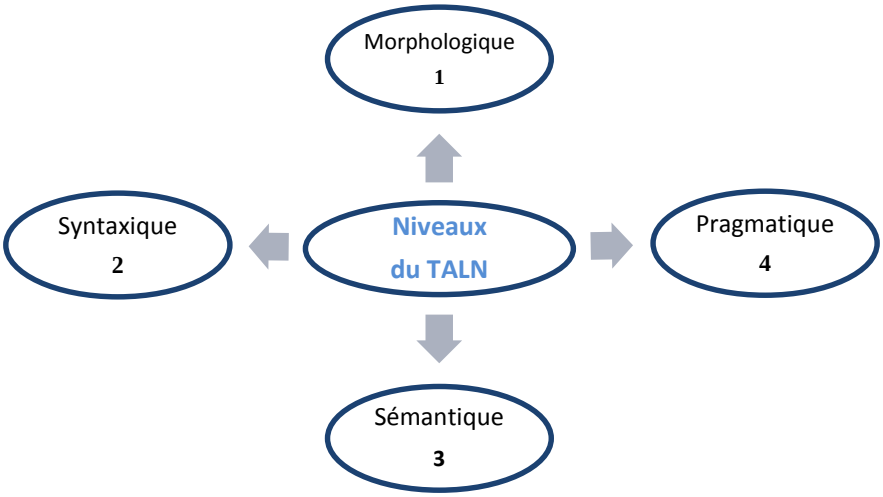


Figure 2 : Niveaux de TALN

2.3.1. Analyse morphologique

L'analyse morphologique est nécessaire pour tout système de traitement automatique de la langue naturelle, cette analyse permet de rassembler les mots en classes morphologiques qui seront utilisées par les autres niveaux d'analyse. L'interprétation de ces classes est relative, et diffère en fonction des traitements souhaités. L'analyse morphologique permet de reconnaître une chaîne de caractère comme étant un mot de la langue. A chaque chaîne est associé : une classe lexicale (grammaticale) décrivant la fonction syntaxique du mot (Exemple : verbe, substantif, adjectif, etc.) ainsi que certaines variables, qui sont des compléments de la description de la fonction syntaxique du mot (Exemple : le genre, le nombre).

L'analyse morphologique implique la construction d'un ensemble de règles que nous appelons une grammaire et un dictionnaire. La grammaire comporte des règles qui justifient la composition des formes à partir des éléments contenus dans le dictionnaire. Cette approche permet de séparer la grammaire de la lexicographie, d'où l'occasion d'enrichir le dictionnaire lorsqu'il est utilisé. Le dictionnaire, commun à la plupart des systèmes TALN, est un facteur majeur dans la stratégie et donc la qualité du système. Il comprend généralement le "vocabulaire" du langage naturel traité, permettant ainsi l'identification des différents éléments constituant un texte [6].

2.3.2. Analyse syntaxique

Un langage formel est défini par sa grammaire, tandis que le langage naturel ne l'est pas. En effet, un langage n'est pas défini par sa syntaxe, car celle-ci est écrite postérieurement et ne

présente qu'une approximation, d'où on parle de modèle syntaxique. C'est cette approximation qui rend l'analyse syntaxique non spécifique et difficile.

Plusieurs méthodes d'analyse syntaxique se sont développées, mais la plus connue est la notion de grammaire formelle. Elle est présentée comme un ensemble de règles de dérivation exprimant la structure des entités syntaxiques telles que la phrase (PH), le groupe nominal (GN), le groupe verbal (GV) et ainsi de suite. Pour exprimer par exemple qu'une phrase est composée d'un groupe nominal et d'un groupe verbal, on utilise la règle PH → GN + GV. Aussi, pour exprimer qu'un groupe nominal est composé d'un déterminant et d'un nom, nous utilisons la règle GN → Det + Name. À l'aide de cet ensemble de règles, il est donc possible d'analyser un certain nombre de phrases [6]. La figure 3 montre un exemple de représentation arborescente de la phrase : « كَتَبَ التِّلْمِيذُ الدَّرْسَ ».

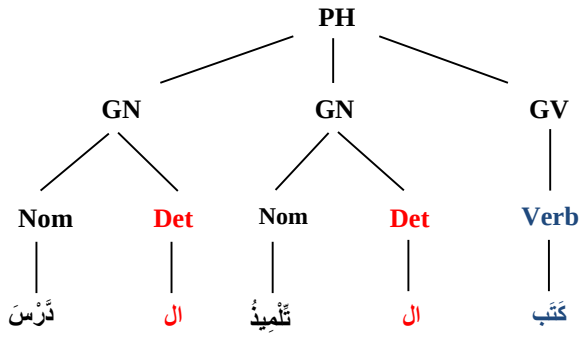


Figure 3 : Arbre syntaxique montrant un exemple de phrase

2.3.3. Analyse sémantique

Selon Jean VERONIS[9] [7], « Dans un système de traitement automatique, l'analyse du sens des phrases consiste généralement à en extraire une représentation simplifiée, stylisée, de type logicomathématique, qui va permettre des calculs et raisonnements ultérieurs ». L'analyse sémantique des énoncés s'appuie sur une analyse syntaxique préalable. Elle cherche à construire une représentation formelle permettant des raisonnements et donc d'inférer de nouvelles informations à partir des informations présentes dans l'énoncé. Parmi les représentations, on trouve la logique des propositions. À l'aide de connecteurs logiques (comme la conjonction « ∧ », la disjonction « ∨ », la négation « ┐ », etc.), on peut former à partir des propositions de nouvelles propositions complexes. La logique des propositions ne s'intéresse

[9]Jean Veronis : Français né en 1955, Professeur de linguistique et d'informatique, spécialiste en TAL.

pas au contenu des propositions mais seulement à leurs valeurs de vérité. Ainsi, de nombreux phénomènes ne peuvent pas être représentés en logique des prédicats.

Une autre forme de représentation appelée « réseaux sémantiques » a été proposée, son principe consiste à représenter la connaissance sous la forme d'un graphe (ou réseau) de concepts. Les nœuds représentent les concepts et les arcs les relations entre ces concepts. Plusieurs types de relations entre concepts existent comme : EST-UN, SORTE-DE, EST-PARTIE-DE, etc. L'utilisation de ces concepts passe par des outils de navigation dans le graphe afin de comprendre le sens de la phrase et les relations entre les différents mots qui la constituent. Les réseaux sémantiques ont été étendus pour améliorer la représentation et l'inférence des connaissances [6].

2.3.4. Analyse pragmatique

Selon J-H. JAYEZ[10] [8], « La pragmatique concerne l'étude de l'environnement d'une phrase, au moment où elle est émise ; elle découle de l'idée qu'une phrase (un énoncé) ne peut prendre tout son sens que si on la (le) replace dans son milieu d'origine ; c'est la prise en compte de toutes les conditions de production d'une phrase, tant il est vrai qu'un acte linguistique effectif ne peut avoir lieu qu'à l'intérieur d'une certaine situation de communication ».

Ce niveau d'analyse recouvre tout ce qui est lié à l'implicite dans la communication. C'est donc le niveau qui pose le plus de problème à concevoir et par conséquent il est beaucoup plus complexe à établir, ce qui explique qu'il n'existe que peu de réalisation opérationnelle, qui concernent quelques applications. On est donc encore loin de savoir construire des analyseurs pragmatiques pour le TALN [6].

2.5. Champs de recherche et applications du TALN

Le champ du traitement automatique du langage naturel couvre de très nombreuses disciplines de recherche qui peuvent mettre en œuvre des compétences aussi diverses que les mathématiques appliquées ou le traitement du signal[11] :

2.5.1. Applications en relation avec la production où la modification de texte

- ✓ Traduction automatique : Il s'agit de l'un des problèmes les plus complexes, dit IA-complet, qui nécessite de nombreuses connaissances, non seulement linguistiques, mais

[10] Jayez, Jacques-Henri : Français né en 1943, Docteur d'Etat en Sciences Mathématiques, linguiste.
[11] https://fr.wikipedia.org/wiki/

aussi concernant le monde. Il s'agit de la première application de recherche, active dès les années 1950.

- ✓ Génération automatique de textes : Écriture de textes syntaxiquement et sémantiquement corrects, pour produire par exemple des bulletins météo ou des rapports automatisés.
- ✓ Résumé automatique de texte, reformulation et paraphrase : Extraction du contenu pertinent d'un texte, détection des informations les plus importantes, des redondances, afin de générer un texte cohérent humainement crédible.
- ✓ Désambiguïsation lexicale : Problème encore non résolu, consistant à déterminer le sens d'un mot dans une phrase, lorsqu'il peut avoir plusieurs sens possibles, selon le contexte général.
- ✓ Correction orthographique : outre une comparaison aux mots du dictionnaire et une recherche approximative afin de proposer des corrections, il existe les correcteurs grammaticaux qui utilisent la sémantique et le contexte afin de corriger les homophonies.
- ✓ Agents conversationnels, et systèmes de questions-réponses : Combinaison d'une étape de compréhension du langage puis une étape de génération de texte.
- ✓ Détection de coréférences et résolution d'anaphores : Détection de la liaison entre plusieurs mots d'une phrase faisant référence à un même sujet.

2.5.2. Applications en relation avec le traitement du signal (parole et graphie)

- ✓ Reconnaissance de l'écriture manuscrite, reconnaissance optique de caractères et lecture automatique de document : Système d'analyse et de traitement des images, couplés à des règles linguistiques permettant d'évaluer la probabilité d'apparition des lettres et mots décodés.
- ✓ Reconnaissance automatique de la parole : Analyse acoustique, association entre segments élémentaires sonore et des éléments lexicaux, puis correspondance des motifs obtenus avec des mots courant, ou des suites de mots apparaissant fréquemment.
- ✓ Synthèse vocale : Une translation vers l'alphabet phonétique est la plus souvent utilisée, mais la catégorie grammaticale est aussi à prendre en compte ; par exemple, il faut reconnaître le second *-ent* comme muet dans l'exemple « Les présidents président ». Les mots dont la prononciation est irrégulière doivent être stockés. De plus, l'intonation et la prosodie sont également à prendre en compte afin d'obtenir un effet naturel.

- ✓ Traitement de la parole : Détection des langues et des dialectes, tant à partir de textes qu'à partir d'énoncés parlés.

2.5.3. Applications en relation avec l'extraction d'information

- ✓ Fouille de textes : Recherche d'informations spécifiques dans un corpus de documents donnés, qui utilise l'indexation de contenu.
- ✓ Recherche d'information : Sous-domaine de la fouille de texte ; l'application la plus connue concerne les moteurs de recherche, qui passent également par l'analyse des métadonnées et des liens entre les pages elles-mêmes.
- ✓ Reconnaissance d'entités nommées : Détermination dans un texte des noms propres, tels que des personnes ou des endroits, ainsi que les quantités, valeurs, ou dates.
- ✓ Classification et catégorisation de documents : Activité qui consiste à classer de façon automatique des ressources documentaires, généralement en provenance d'un corpus.
- ✓ Systèmes de tutorat intelligents : Utilisés notamment pour l'enseignement des langues
- ✓ Analyse de sentiment : Vise à extraire le ressenti d'un texte (généralement positif ou négatif) en fonction des mots et du type de langage utilisé, d'indices typographiques ou de la personne qui l'a écrit.
- ✓ Recommandation automatique de documents : Consiste à extraire l'information importante d'une base de documents afin de les relier en « séries », afin de proposer ses éléments aux personnes intéressées par d'autres éléments de cette série.

3. Langue arabe

La langue arabe est une langue vivante qui fait partie de la famille des langues sémitiques[12], parlée par plus de 500 millions de personnes en tant que langue maternelle, tout en devenant la langue officielle de plus de 22 pays, présentés dans la figure 4, dans une zone s'étendant du golfe Arabo-Persique à l'est jusqu'à l'océan Atlantique à l'ouest.

Figure 4 : Le monde arabe[13]

L'alphabet arabe se compose de vingt-huit lettres (tableau 1) qui sont des consonnes, dont trois sont des semi-consonnes ou voyelles longues. Comme le chinois, le japonais et le coréen, il n'y pas de capitalisation en arabe. De Plus, les lettres arabes changent de forme en fonction de leur position dans le mot [9].

n°	Lettre	Prononciation	Translitération
1	ا	alif	ā
2	ب	bā'	b
3	ج	jīm	j
4	د	dāl	d
5	ه	hā'	h
6	و	wāw	w / ū
7	ز	zayn/zāy	z
8	ح	ḥā'	ḥ
9	ط	ṭā'	ṭ
10	ي	yā'	y / ī
11	ك	kāf	k
12	ل	lām	l
13	م	mīm	m
14	ن	nūn	n
15	س	sīn	s
16	ع	ʻayn	ʻ
17	ف	fā'	f

n°	Lettre	Prononciation	Translitération
18	ص	ṣād	ṣ
19	ق	qāf	q
20	ر	rā'	r
21	ش	shīn	sh
22	ت	tā'	t
23	ث	thā'	th
24	خ	khā'	kh
25	ذ	dhāl	dh
26	ض	ḍād	ḍ
27	ظ	ẓā'	ẓ
28	غ	ghayn	gh

Tableau 1 : Les 28 lettres Arabe[14]

[12] Langues sémitiques : font partie de la famille des langues afro-asiatiques, et sont parlées en Afrique septentrionale et saharienne ainsi qu'au Proche-Orient et au Moyen-Orient.
[13] https://fr.wikipedia.org/wiki/Culture_arabe.
[14] https://fr.wikipedia.org/wiki/Alphabet_arabe.

3.1. Origine de la langue arabe

La première grammaire arabe, rédigée par Sibawahi (8e siècle) dans « Al-Kitab » constitue le premier travail de normalisation de la langue. Il a répondu aux inquiétudes des religieux, qui à l'époque des premières conquêtes musulmanes, voulaient éviter tout risque de corruption de la parole divine pouvant résulter de la manipulation de la langue par les nouveaux convertis à l'Islam d'origine [10].

3.2. Variétés de la langue arabe

Les linguistes arabes ont structuré la langue arabe en deux variétés principales, la première variété dite « l'arabe classique » ou « l'arabe littéraire » et la deuxième dite « l'arabe dialectal ». D'autres linguistes ont ajouté une variété intermédiaire écrite et parlée, et désignée par le nom « arabe standard contemporain ».

3.2.1. Arabe classique

C'est la forme linguistique ancienne de l'arabe dont les règles de la grammaire étaient fixées entre le 8e et le 10e siècle. Il est appris dans les établissements d'enseignement à travers la littérature arabe et les cours de théologie [11].

3.2.2. Arabe standard moderne ou contemporain (ASM)

C'est une variante moins formelle que l'arabe classique. L'arabe standard est utilisé dans la vie officielle, universitaire et administrative. De plus, c'est par le biais de l'arabe standard, que deux locuteurs arabophones d'origines dialectales différentes sont susceptibles de se comprendre [11].

3.2.3. Arabe dialectal

Il est souvent utilisé dans l'expression de la vie quotidienne locale. Elle est considérée comme la langue vernaculaire de l'ensemble des arabophones. Par définition, les dialectes arabes sont les langues maternelles des populations des pays arabes, et ces formes linguistiques sont parfois très différentes d'une région à l'autre [11].

On peut regrouper ces dialectes en quatre grands groupes [12] :

- ✓ Les dialectes arabes, parlés dans la Péninsule Arabique : dialectes du Golfe, dialecte du najd, yéménite ;
- ✓ Les dialectes maghrébins : algérien, marocain, tunisien, hassaniya de Mauritanie ;

- ✓ Les dialectes proche-orientaux : égyptien, soudanais, syro-libano-palestinien, irakien (nord et sud) ;
- ✓ La langue maltaise : est également considérée comme un dialecte arabe.

3.4. Grammaire arabe

La grammaire traditionnelle se divise en : Morphologie et Syntaxe [13].

3.4.1. Morphologie (الصرف)

La morphologie arabe est une science étudiant la structure du mot arabe et ses changements par l'ajout des particules pour former des dérivés et des formes flexionnelles.

La morphologie se divise en deux types :

- Morphologie dérivationnelle : *al-ichtikak* (الإشتقاق), qui étudie la dérivation des mots par un autre mot et leur transformation selon le sens voulu ; autrement dit, la dérivation morphologique est décrite sur une base morpho-sémantique : d'une même racine, se dérivent des mots différents, *siyar* (صيغ).
- Morphologie flexionnelle : comprenant d'une part a flexion *IiEerab* (الإعراب), concerné le changement de marquage casuel selon le changement des facteurs qui précède, et d'autre part, la non conjugué *BinaAe* (البناء), qui concerne la stabilité de marquage casuel même avec les changements des facteurs qui précède.

Cette morphologie est dirigée par plusieurs facteurs comme : le temps, les indices, l'aspect, le genre, le nombre qui sont en général des suffixes et préfixes.

3.4.2. Syntaxe (النحو)

Étudie la formation correcte des phrases par l'analyse de :

- Position des unités lexicales les unes par rapport aux autres pour déterminer l'ordre des unités lexicales.
- Marquage casuel des unités lexicales de la phrase, Ainsi, la fonction syntaxique de chaque unité qu'est déterminée en s'appuyant sur la morphophonologie

3.5. Structure d'une phrase arabe

La langue arabe est composée de deux types de phrases : les phrases verbales et les phrases nominales :

- **Les phrases verbales (الجملة الفعلية) :** servent à indiquer un évènement ou une action. Elles commencent par un verbe suivi d'un sujet et d'un complément ; ce dernier est optionnel.
- **Les phrases nominales (الجملة الإسمية) :** en arabe ne contiennent pas de verbe ; il est sous-entendu. Le verbe peut être implicite, les verbes ne sont pas obligatoires pour construire une phrase.

3.6. Structure d'un mot arabe

En arabe un mot peut désigner toute une phrase grâce à sa structure composée qui est une agglutination d'éléments de la grammaire, ceci définit le mot graphique arabe ; cette appellation est désignée par David Cohen, l'un des premiers théoriciens du domaine du TAL, à un mot décomposable au proclitiques, forme fléchies, enclitique avec la forme fléchie représente le noyau lexical [13].

La figure 5 suivante représente les différents composants d'un mot arabe (maximal) dont l'ordonnancement est de droite à gauche.

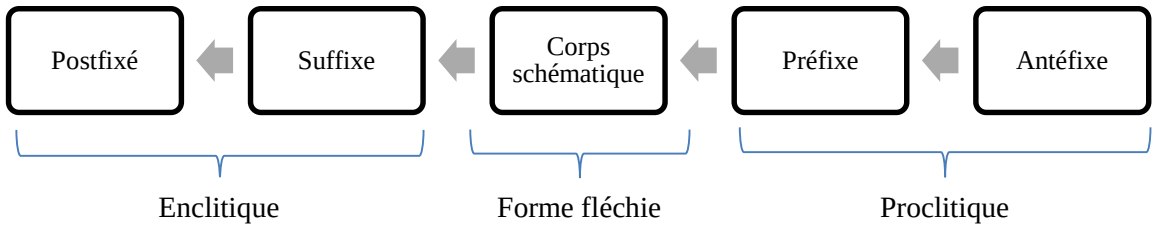


Figure 5 : Structure d'un mot arabe selon David Cohen [13]

- Les Antéfixes sont des prépositions ou des conjonctions ;
- Les Préfixes et Suffixes expriment les traits grammaticaux et présentent les fonctions : cas du nom, mode du verbe et les modalités (nombre, genre, personne...) ;
- Les Postfixés sont des pronoms personnels ;
- Le corps schématique représente la base de mot « radicale ».

Exemple : أَتَتَفَكَّرُونَنَا (En français : Est-ce que vous pensez à nous ?)

La segmentation de ce mot donne les caractéristiques suivantes (tableau 2) :

أَتَتَفَكَّرُونَنَـا « Est-ce que vous pensez à nous ? »				
نَا	ونَ	تَفَكَّر	تَ	أَ
Postfixé	**Suffixe**	**Corps schématique**	**Préfixe**	**Antéfixe**
Pronom Suffixe complément	Suffixe Verbal Expriment le pluriel	Dérivé de la racine **فكّر** selon le schème **فكر**	préfixe verbal du temps de l'inaccompli	conjonction d'interrogation

Tableau 2 : Exemple de la décomposition du mot arabe أَتَتَفَكَّرُونَنَا

3.7. Catégories des mots arabes

On distingue trois grandes catégories des mots arabes : les Verbes, les Noms et les Particules.

3.7.1. Le verbe

Unité lexicale référant à un état ou une action exprimant un sens dépendant du temps comme : عَمِلَ (travailler), ذَهَبَ (partir). Nous pouvons classer les verbes arabes selon plusieurs critères [13] :

- Selon le critère de temps, il existe trois types : l'accompli, inaccompli, impératif.
- Selon leur sens et leur transitivité de sujet au complément aux deux types : intransitive, transitive.
- Selon leurs modes aux deux types : la voix passive et la voix active.
- Selon le nombre des consonnes de la racine, la majorité des verbes, environ de 85%, sont formés sur 3 lettres et le reste entre les racines de 4 et 5 lettres. Ces racines peuvent donner plusieurs schèmes avec des transformations morphologiques.
- Selon le schème et le nombre de consonnes qui constituent la structure verbale, nous avons soit des verbes nus (مُجَرَّد), soit des verbes augmentés (مَزِيد).
- Selon leur conjugaison il existe : le conjugué et le non conjugué ou bien invariant.
- Il existe aussi les verbes d'exclamation ainsi les verbes panégyriques et les verbes de diatribe.

3.7.2. Les noms

Les noms arabes sont de deux catégories, ceux qui sont dérivés de la racine verbale et ceux qui ne le sont pas comme les noms propres et les noms communs. Dans le premier cas, le fait que le nom soit dérivé d'un verbe, il décrit donc une certaine sémantique qui pourrait avoir une influence dans la sélection des phrases importante d'un texte pour le résumé [10].

La déclinaison des noms obéit à plusieurs règles [11] :

- Le féminin singulier : dans la plupart des cas, on ajoute la lettre ة pour obtenir le nom au féminin singulier (exemple : صَغيرٌ « petit » devient صَغِيرَةٌ « petite »).

- Le féminin pluriel : généralement, on rajoute les deux lettres ات (exemple : عامل « *ouvrier* » devient عامِلات « *ouvrières* »).

- Le masculin pluriel : généralement, on rajoute les deux lettres ون ou les deux lettres ين qui dépendent de la position du mot dans la phrase (sujet ou complément d'objet).

Exemple : مُعلِّمٌ « *enseignant* » devient مُعلِّمِين ou مُعلِّمُون « *enseignants* ».

- Le pluriel irrégulier : c'est le cas le plus complexe en arabe, pour transformer un mot au pluriel, on doit insérer des lettres au début, au milieu ou à la fin du mot, (exemple : قُفل « serrure » devient أَقفَال « serrures »).

3.7.3. Les particules

En arabe, les particules sont principalement les mots outils de la langue, comme les conjonctions de coordination et de subordination. Elles ont un rôle très important dans l'interprétation de la phrase. Les particules sont classées selon leur fonction dans la phrase, on en distingue plusieurs types (introduction, explication, conséquence). Elles jouent un rôle important dans l'interprétation de la phrase. Elles sont utilisées dans la phrase pour situer des faits ou des objets par rapport au temps ou au lieu ; elles jouent, également, un rôle clé dans l'enchaînement et la cohérence d'un texte.

Par exemple, on trouve des particules temporelles ou spatiales qui désignent :

- Un temps مُنذ (pendant), قَبل (avant), بَعْد (après).

- Un lieu : حَيْث (où).

On trouve aussi des particules qui peuvent exprimer des pronoms relatifs (la détermination avec une valeur référentielle), par exemple : الَّذين « ceux », الَّذي « ce », الَّتي «cette».

De même façon que les noms et verbes arabes, certaines particules peuvent également avoir des préfixes et suffixes ce qui rajoute une complexité quant à leur identification [11].

3.8. Problèmes de la langue arabe en TALN

L'arabe, comme toutes les langues naturelles, est caractérisée par un ensemble de phénomènes créant des difficultés et des problèmes qu'il faut prendre en considération lors d'un traitement automatique. En plus des phénomènes classiques, comme l'ambiguïté, la coordination ou l'anaphore, nous trouvons aussi dans le cas de l'arabe d'autres phénomènes

propres aux langues sémitiques tel que l'absence de voyelles, la segmentation des textes, l'ordre des mots dans une phrase et l'agglutination (enclitique, proclitique) dont nous avons vu.

3.8.1. Absence des voyelles

Sur le plan de l'écrit, on trouve dans la langue arabe des voyelles diacritiques qui sont écrites au-dessus ou au-dessous une consonne pour lui donner le son et le sens désiré. Mais les arabophones utilisent rarement ces voyelles dans leur écriture, puisque pour eux le contexte suffit à leur faire comprendre le sens exact du mot.

Cela peut cependant aboutir à de nombreuses ambiguïtés lors du traitement automatique de la langue. En fait, dans la langue arabe on utilise différents types de voyelles qui déterminent la prononciation des lettres [14].

3.8.1.1. Voyelles brèves ou courtes (المصوتات القصيرة)

Ces voyelles sont notées avec des signes diacritiques qui permettent de savoir comment une lettre est prononcée dans un mot donné. L'écriture de ces voyelles est facultative et concerne peu de textes (Coran et textes didactiques), ce qui cause beaucoup d'ambiguïté au niveau de la prononciation et la compréhension de textes courants. Ces voyelles ne s'écrivent pas dans le corps du mot, mais ils sont ajoutés sur ou sous la consonne à laquelle ils se réfèrent [14] (tableau 3).

- Le Fatha et le Kasra sont représentés par un tiret mis respectivement au-dessus et au-dessous de la consonne à laquelle ils se réfèrent.
- Le Damma ressemble à un petit Waw (و) et il est écrit au-dessus de sa consonne.

Signe	Translitération	Nom arabe	Nom de la voyelle
◌َ	A	فَتْحَة	Fatha
◌ِ	I	كَسْرَة	Kasra
◌ُ	U	ضَمَّة	Damma

Tableau 3 : Voyelles courtes (arabe)

3.8.1.2. Voyelles longues (المصوتات الطويلة)

Elles se prononcent de manière prolongée. Chaque type de voyelles longues correspond à un autre type de voyelles brèves au niveau de la prononciation. Elles ne sont pas comme les voyelles brèves et qui sont sous la forme des signes, mais elles sont représentées par des lettres et s'insèrent dans le mot exactement comme les consonnes [15] (tableau 4).

Voyelle courte	Exemple de sons courts	Prononciation selon API	Elongation de signe	Résultat	Exemple	Prononciation selon API	Sens
◌َ	ب	/ba/	أ	بَا	بَاب	/bab/	porte
◌ِ	ب	/bi/	ي	بِي	كبِير	/kabir/	grand
◌ُ	س	/su/	و	سُو	سُوق	/suq/	marché

Tableau 4 : Voyelles longues (arabe)

3.8.1.3. Tanwin (التنوين**)** : (Alfatha, Alkasra, Aldamma), c'est-à-dire le doublement de voyelle courte : Tanwin Alfatha et Aldamma se trouvent sur la dernière lettre d'un mot mais Tanwin Alkasra se trouve sous la dernière lettre. Ces types de Tanwin sont utilisés pour indiquer que la prononciation de la dernière lettre est le son de cette lettre influencée par une des voyelles courtes, une Fatha, Damma ou Kasra, selon le type de Tanwin trouvé, et la deuxième Fatha, Damma ou Kasra désigne le son /n/.

Voici les différents types de Tanwin :

- ✓ Tanwin Alfatha : (◌ً) ;
- ✓ Tanwin Alkasra : (◌ٍ) ;
- ✓ Tanwin Aldamma : (◌ٌ).

3.8.1.4. Shadda الشَّدة (◌ّ) : désigne un doublement de la lettre sur laquelle se trouve.

3.8.1.5. Sukun السُّكون **(**◌ْ**)** : indique que la prononciation de la lettre où se trouve le SUKUN devrait être claire et aigu.

3.8.2. Segmentation des textes

La segmentation du texte arabe en phrases est difficile car d'une part la ponctuation est rarement utilisée et d'autre part cette ponctuation ne reflète pas la seule possibilité de segmentation des phrases, ce qui nécessite des analyses locales au niveau du texte afin de pouvoir segmenter : d'où le paradigme « segmenter pour analyser ou analyser pour segmenter ? ». D'autre part, la segmentation manuelle en phrases n'est pas évidente, elle peut différer d'une personne à une autre [10].

3.8.3. Problème de l'ordre des mots dans la phrase

La construction des phrases en arabe est flexible, dans le sens où l'ordre des mots dans une phrase donnée est relativement libre. Généralement, un mot placé au début de la phrase est un terme sur lequel nous voulons attirer l'attention, s'en suit le terme le plus long ou le plus riche en sens ou en sonorité. Cette flexibilité provoque des ambiguïtés syntaxiques artificielles

due à la prise en compte de toutes les règles de combinaison possibles des composants d'une phrase [16]. Pour illustrer cette propriété prenons les phrases suivantes :

- Verbe + sujet + complément :

تأهلت الجزائر إلى كأس العالم (L'Algérie s'est qualifiée pour la coupe du monde)

- Sujet + verbe + complément :

الجزائر تأهلت إلى كأس العالم (C'est l'Algérie qui s'est qualifiée en coupe du monde)

- Complément + verbe + sujet :

إلى كأس تأهلت الجزائر العالم (C'est pour la coupe du monde que l'Algérie s'est qualifiée)

4. Conclusion

Dans ce chapitre nous avons présenté l'état de l'art du TALN à partir de son histoire, niveaux de traitement, domaines d'applications, ainsi que l'étude des caractéristiques principales de la langue arabe qui sont différentes à d'autres langues. La langue arabe se caractérise par une richesse flexionnelle, par ses signes de vocalisation et son ambiguïté, ce qui présente des nombreux défis pour des domaines divers tels que le traitement automatique du langage naturel ou aussi la recherche d'informations.

Chapitre III

Recherche d'Information (RI)

Chapitre III :

Recherche d’Information

1. Introduction

Parmi les caractéristiques inhérentes de l’ère de l'information est la croissance rapide et l’explosion de la quantité d'information hétérogène ; ce qui conduit à la nécessité de concevoir des approches pour enregistrer, filtrer, restituer et gérer cette quantité illimité d'informations.

Les utilisateurs, qui ont besoin d’informations, font face à un problème de surcharge d'informations, ce qui nécessite des technologies appropriées pour accéder à ce volume croissant d’informations produites. Cela a donné naissance au domaine de Recherche d’Information (RI). La RI classique traite principalement des documents non structurés, qui consistent principalement en une forme libre de langage naturel, qui n'est pas toujours bien structuré et peut être sémantiquement ambiguë. Ainsi, les tendances actuelles dans ce domaine traitent de la recherche d’information sémantique telle que la recherche d’information à base d’ontologies et la recherche personnalisée d'informations basée sur le profil utilisateur [17].

Dans ce chapitre, nous allons d’abord commencer par présenter un bref historique de la RI, avant d’arriver à une définition adéquate de la RI et de déterminer les concepts de base de la recherche d'information et ses composants. Ensuite nous allons donner une description sur les processus et les principaux modèles de RI. Enfin, ce chapitre se termine par décrire les mesures d’évaluation communément utilisées dans le contexte de la RI.

2. Bref historique de la recherche d’information

La RI n'est pas un domaine récent, il date des années 1940, dès la naissance des ordinateurs. Au début, la RI se concentrait sur les applications dans des bibliothèques, d'où aussi le nom "automatisation de bibliothèques". Depuis le début de ces études, la notion de pertinence a toujours été un objet. Dans les années 1950, début de petites expérimentations en utilisant des petites collections de documents (références bibliographiques). Dans les années 1960 et 1970, des expérimentations plus larges ont été menées, et le développement d’une méthodologie d'évaluation du système, qui est aussi utilisée maintenant dans d'autres domaines. Des corpus de test ont été conçus pour évaluer des systèmes différents, ils ont beaucoup contribué à l'avancement de la RI, car on pouvait les utiliser pour comparer différentes techniques, et de mesurer leurs impacts en pratique. Le système qui a le plus influencé le

domaine est sans aucun doute SMART, développé à la fin des années 1960 et au début des années 1970. Les travaux sur ce système ont été dirigés par G. Salton, professeur à Cornell. Certaines nouvelles techniques ont été implantées et expérimentées pour la première fois dans ce système (par exemple, le modèle vectoriel et la technique de relevance feedback). Du côté de modèle, il y a aussi beaucoup de développements sur le modèle probabiliste. Les années 1980 ont été influencées par le développement de l'intelligence artificielle. Ainsi, on tentait d'intégrer des techniques de l'IA en RI, par exemple, système expert pour la RI, etc. Les années 1990 (surtout à partir de 1995) sont des années de l'Internet. Cela a pour effet de propulser la RI en avant-scène de beaucoup d'applications. C'est probablement grâce à cela que vous entendez parler de la RI. La venue de l'Internet a aussi modifié la RI. La problématique est élargie. Par exemple, on traite maintenant plus souvent des documents multimédia qu'avant. Cependant, les techniques de base utilisées dans les moteurs de recherche sur le web restent identiques [18] [19].

3. Définitions de la recherche d'information

Définition 1 : La recherche d'information est un ensemble de techniques offrant la possibilité de retrouver à partir d'une grande masse de documents ceux qui sont susceptibles correspondre aux besoins d'un utilisateur utilisant souvent une requête dans un langage naturel [20].

Définition 2 : La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information pertinente pour un utilisateur [31].

Un système de recherche d'information (SRI) est un ensemble de programmes informatiques ayant pour but de satisfaire le besoin en information d'une requête utilisateur. Son rôle principal est de sélectionner les documents les plus pertinents correspondant à un besoin en information formulé par une requête [21].

Cette définition fait ressortir les éléments clés suivants[15] (figure 6) :

- **Document** : On appelle document toute unité qui peut constituer une réponse à une requête utilisateur. Un document peut être un texte, un morceau de texte, une page Web, une image, une bande vidéo, etc.

[15] https://sites.google.com/site/btsrechercheinformation/home/ii-generalites-sur-les-sri

- **Requête** : Une requête correspond à la traduction du besoin d'information de l'utilisateur dans un langage interrogation du SRI. Elle est généralement constituée d'une liste de mots-clés.

- **Pertinence** : la notion de pertinence est très complexe tant est une notion subjective. Elle peut être définie comme étant une correspondante entre un document et une requête, ou une mesure d'informativité du document à la requête. Elle constitue l'objectif principal de la RI.

Deux types de pertinence sont définis, la pertinence système et la pertinence utilisateur [2] :

1- La pertinence système : est souvent présentée par un score attribué par le système de recherche d'information (SRI) afin d'évaluer l'adéquation du contenu des documents vis-à-vis de celui de la requête. Ce type de pertinence est objectif et déterministe.

2- La pertinence utilisateur : quant à elle, se traduit par les jugements de pertinence utilisateur sur les documents fournis par le système de recherche d'information en réponse à une requête. La pertinence utilisateur est subjective ; car pour un même document retourné en réponse à une même requête, il peut être jugé différent par deux utilisateurs distincts (qui ont des centres d'intérêt différent). De plus, cette pertinence est évolutive, un document jugé non pertinent à un certain instant pour une requête peut être jugé ultérieurement pertinent, car la connaissance de l'utilisateur sur le sujet a évolué.

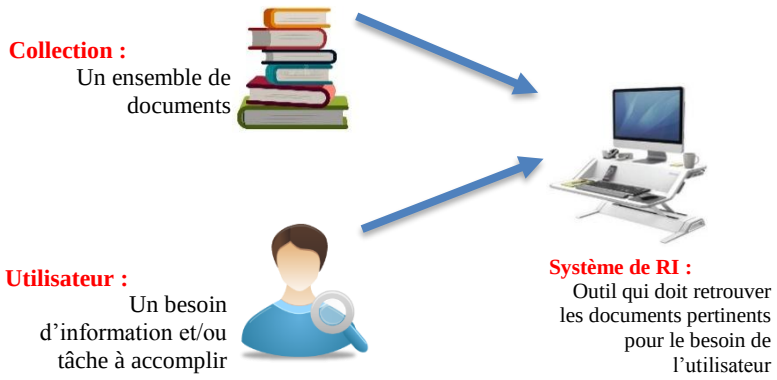


Figure 6 : Les acteurs de Recherche d'Information[16]

4. Processus de recherche d'information

Le processus de RI c'est la procédure fondamentale qui a pour but la mise en relation des informations disponibles d'une part, et les besoins de l'utilisateur d'autre part. L'accès à l'information paraît à l'utilisateur une tâche simple pouvant être récapitulée en quelques clics,

[16] https://vdocuments.net/xavier-tannier-xaviertannierlimsifr-recherche-evaluation-indexation-et-recherche-dinformation.html

tandis que derrière cette simplicité se cache un processus sophistiqué. En fait le processus de RI comporte deux grandes phases : l'indexation et la recherche [22].

Ce processus de recherche peut se résumer comme suit (figure 7) :

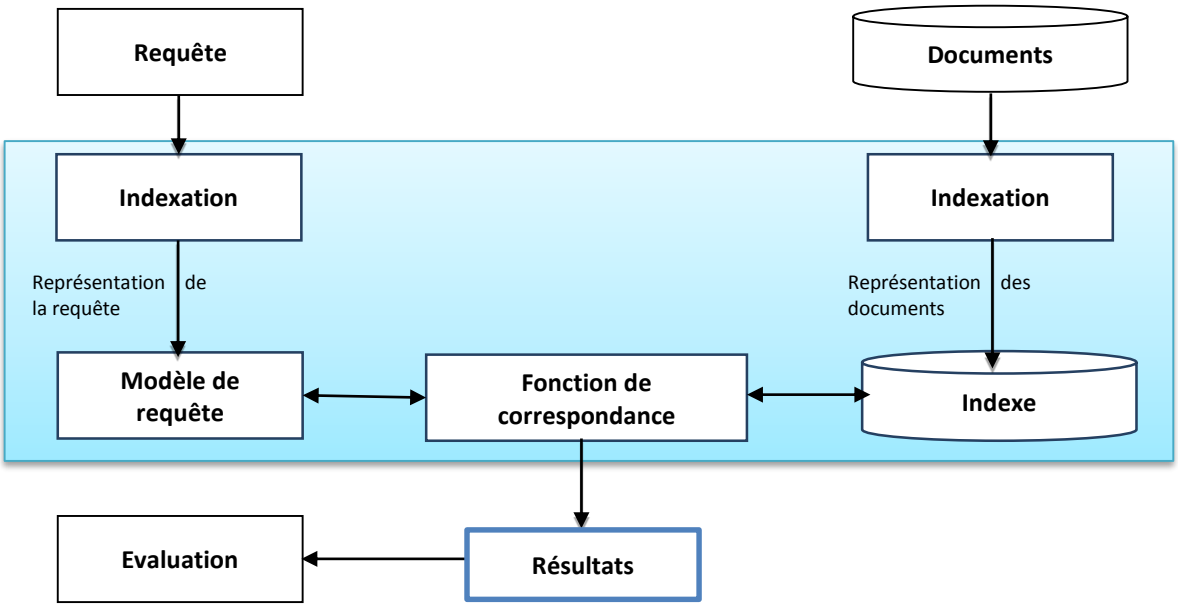


Figure 7 : Processus de système de recherche d'information (SRI)[17]

L'architecture générale d'un SRI, illustrée par la figure 5, fait ressortir des éléments constitutifs tel que : le document, le besoin en information, la requête et la pertinence, ainsi que trois principales fonctionnalité : l'indexation, la recherche et la reformulation de la requête.

4.1. Phase d'indexation

L'indexation représente une étape cruciale dans le processus global de la RI. Son objectif est la représentation des ressources (requêtes et documents) dans un format exploitable par un SRI dans le but de construire ce qu'on appelle index.

L'indexation des documents du corpus a pour objectif d'éviter au système de les analyser à chaque interrogation, en effet l'index créé permet d'établir le lien vers les documents indexés à travers les mots-clés représentatifs de leurs contenus [22].

La phase d'indexation peut être effectuée de trois manières [2] :

- **Manuellement** : chaque document de la collection est analysé par un spécialiste de domaine ou un documentaliste. L'indexation manuelle assure une meilleure précision dans les documents

[17] Support de cours Dr. L. OUKID, Module Recherche d'Information, Université de Blida, 2021.

restitués par le SRI en réponse aux requêtes des utilisateurs. Néanmoins, cette indexation présente un certain nombre d'inconvénients liés notamment à l'effort et le prix qu'elle exige (en temps et en nombre de personne). De plus, cette indexation est subjective, qui liée au facteur humain, où différents spécialistes peuvent indexer un document avec les termes différents.il se peut aussi arriver qu'un spécialiste indexe différemment un document, à différents moments.

- **Automatique** : dans ce cas, l'indexation est entièrement automatisée. Elle est réalisée par un programme informatique et elle passe par un ensemble d'étapes pour créer d'une façon automatique l'index. Ces étapes sont : l'analyse lexicale, l'élimination des mots vides, la normalisation (lemmatisation ou radicalisation), la sélection des descripteurs, le calcul de statistiques sur les descripteurs et les documents (fréquence d'apparition d'un descripteur dans un document et dans la collection, la taille de chaque document, etc.), et enfin la création de l'index et éventuellement sa compression.

- **Semi-automatique** : la tâche d'indexation est réalisée ici conjointement par un programme informatique et un spécialiste du domaine. Le choix final des descripteurs revient à l'indexeur humain. Dans ce type d'indexation un langage d'indexation contrôlé est généralement utilisé.

Le processus d'indexation automatique, comme nous l'avons vu, comprend plusieurs étapes, qui sont décrites comme suit (figure 8) :

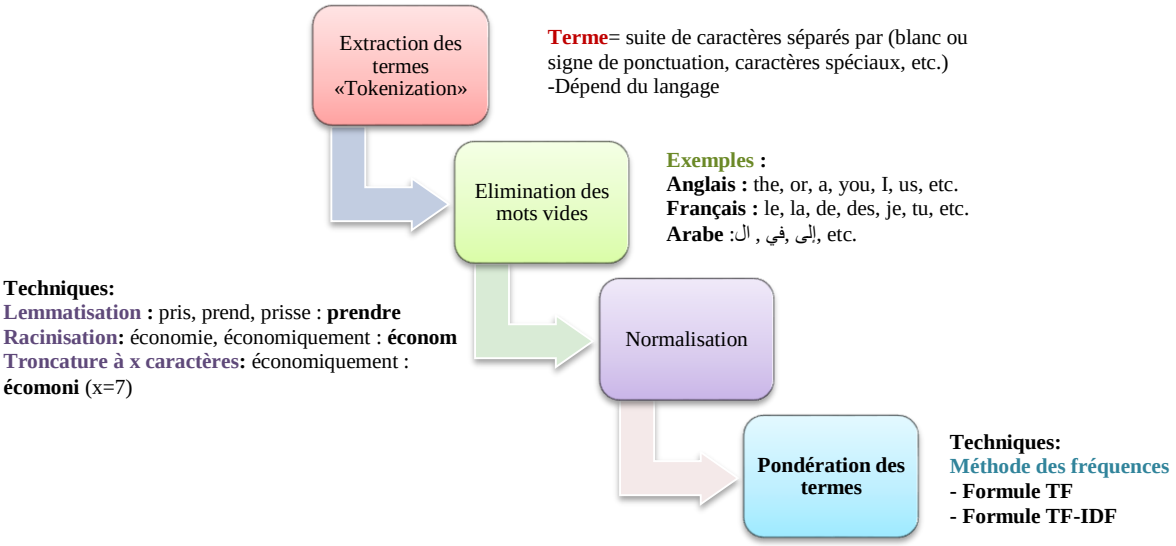


Figure 8 : Processus d'indexation automatique classique[18]

[18] Support de cours Dr. L.OUKID, Module Recherche d'Information, Université de Blida, 2021.

4.1.1. Analyse lexicale (Tokenization)

Il s'agit d'analyser chacun des documents afin d'en extraire le contenu. Un texte peut être découpé en plusieurs unités différentes chacune nécessitant un traitement bien défini. Le format de découpage le plus utilisé est basé sur les mots. Les contenus textuels des documents sont dans ce cas analysés et l'ensemble des mots qui s'y trouvent est tiré dans une liste. Cette dernière contient pour chaque document, l'ensemble des mots qui s'y trouvent [21].

4.1.2. Élimination des mots vides

L'élimination de mots vides a pour objectif de supprimer les termes non significatifs (pronoms personnels, prépositions, etc.) ou mots arithmétiques (les mots qui peuvent se trouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traite pas, comme par exemple contenir, appartenir, etc.). Cette étape vise à réduire le nombre de mots utilisés pour indexer les documents et rendre la recherche plus rapide [23] [24].

On distingue deux techniques pour éliminer les mots vides :

- L'utilisation d'une liste de mots vides (aussi appelée anti-dictionnaire ou stop words) ;
- L'élimination des mots dépassant un certain nombre d'occurrences dans la collection.

4.1.3. Lemmatisation/Racinisation

Un mot donné peut avoir déférentes formes dans un texte. On peut par exemple citer économie, économiquement, économétrie, économétrique, etc. Il n'est pas forcément nécessaire d'indexer tous ces mots et un seul suffirait à représenter le concept véhiculé. Pour résoudre le problème, une substitution des termes par leur racine ou lemme est utilisée. La lemmatisation consiste à remplacer chaque mot (par ex. «الأفكار») par sa forme canonique («فكر »). La racinisation consiste à remplacer chaque mot (par ex. «الأفكار») par sa racine (« فكار »).

Plusieurs méthodes de lemmatisation ont été proposées dans la littérature, parmi lesquelles : la méthode de n-grammes (sur laquelle nous nous appuyons dans notre travail), la troncature, les dictionnaires ou l'élimination des affixes [17] [23].

4.1.4. Pondération

La pondération est l'étape finale dans la plupart des applications d'indexation. Elle permet d'assigner aux termes leur degré d'importance dans les documents. Un terme peut être expressif s'il apparaît fréquemment pour être statistiquement important sans toutefois excéder une certaine limite qui le classait dans la catégorie des mots outils (vides). La plupart des techniques de pondération sont basées sur deux facteurs : la pondération locale qui quantifie la

représentativité locale d'un terme dans le document (*tf* : Term Frequency) et la pondération globale qui quantifie la représentation du terme vis-à-vis de la collection complète des documents (*idf* : Inverse of Document Frequency).

- ***Tf*** : cette mesure est proportionnelle à la fréquence du terme dans le document. Plusieurs formules de pondération ont été proposées, parmi lesquelles : la fonction brute (nombre d'occurrences), ou bien une combinaison avec d'autre fonction [23] :

$$Tf_{t,d} = \frac{n_{t,d}}{N_d}$$

Où $n_{t,d}$ est la fréquence d'apparition du terme *t* dans le document *d* et N_d est le nombre total des termes dans d.

- ***Idf*** : ce facteur mesure l'importance d'un terme dans toute la collection. Un terme qui apparaît souvent dans la base documentaire ne doit pas avoir le même impact qu'un terme moins fréquent. Il est généralement exprimé comme suit [23] :

$$Idf_t = log \frac{D}{\{d_j : t_i \in d_j\}}$$

Où *D* est le nombre total de documents dans la collection et $\{d_j : t_i \in d_j\}$ représente le nombre de documents où le terme *t* apparait

Les systèmes de pondération les plus courants utilisent le poids local et global à la fois, tel que le poids d'un terme égal Tf * Idf. Cette mesure donne une bonne approximation de l'importance du terme dans les corpus des documents de tailles homogènes.

$$\boldsymbol{Tf - Idf = Tf_{t,d} * Idf_t}$$

Donc, le Tf * Idf (Term Frequency * Inverse Document Frequency) est le résultat d'un calcul, dans l'algorithmie des moteurs de recherche, permettant d'obtenir un poids, une évaluation de la pertinence d'un document par rapport à un terme, en tenant compte les deux facteurs *Tf* et *Idf* dans le corpus étudié.

4.2. Phase de recherche (ou d'interrogation)

Cette étape inclut un ensemble d'actions qui commence par la formulation du besoin d'information de l'utilisateur en une requête avant qu'elle soit analysée, indexée et représentée suivant le modèle adopté par le système. Il vient après la phase de recherche proprement dite qui consiste à faire l'appariement entre la requête soumise et la collection de documents maintenue sous forme d'index afin de repérer ceux qui peuvent répondre à la requête. Cette correspondance est faite par le biais des mesures de similarité telles que la mesure du cosinus, le produit scalaire, etc. [22].

4.3. Indexation sémantique et WordNet

4.3.1. Indexation sémantique

Les SRI classiques présentent des insuffisances du fait de leur incapacité à traiter avec l'ambiguïté de la langue et l'imprécision sémantique des mots simples. Pour lever ces problèmes d'ambiguïté et de disparité des mots, de nombreux travaux de recherche en RI se sont orientés vers la prise en compte des sens des mots dans le processus d'indexation. L'indexation sémantique, se base sur les sens des mots (entités sémantiques) plutôt que sur les mots simples (entités lexicales) pour indexer les documents. Pour retrouver les sens des mots dans un contenu donné, l'indexation sémantique se base sur des techniques de désambiguïsation contextuelle des mots dans les documents et requêtes.

La désambiguïsation a pour objet de retrouver le sens d'un mot dans un contenu donné. Pour ce faire, la désambiguïsation s'appuie [25] (figure 9) :

1- sur des corpus d'apprentissage : Une manière d'indexer serait par exemple, d'associer aux mots extraits, des mots du contexte qui aident à déterminer leur sens. Une autre manière serait d'apprendre le sens d'un mot à partir de ses usages possibles du mot à désambiguïser, ou à partir de règles d'agencement ou règles de fonctionnement des mots à désambiguïser.

2- sur des ressources linguistiques externes : Telles que les thésaurus, dictionnaires automatisés, ontologies, et autres comme Wikipédia, qui constituent des sources d'évidence pour les définitions et sens du mot cible. On parle alors d'indexation conceptuelle.

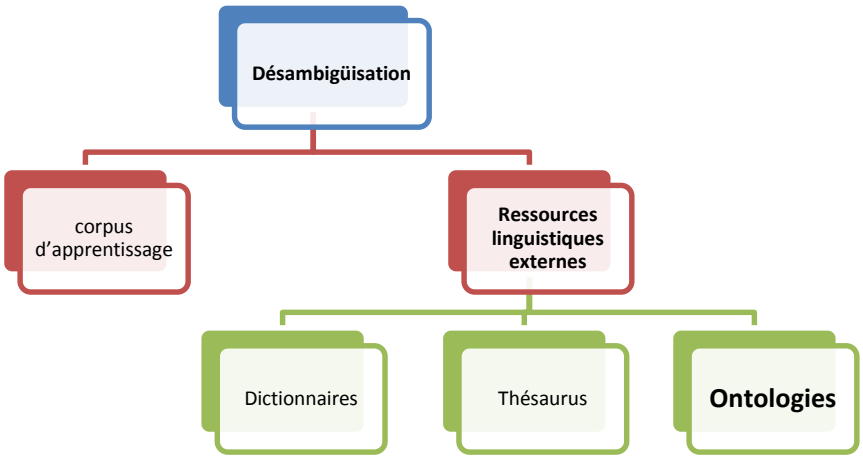


Figure 9 : Indexation sémantique et la désambiguïsation des sens des termes

La réalisation d'une partie de notre projet se base sur un corpus Anglais-Arabe, pour cela on a opté d'utilisé l'ontologie « WordNet » comme source d'évidence pour l'identification des sens des mots et requête à travers « les synsets » et aussi pour proposer un classement sémantique des résultats retournés par le moteur de recherche.

4.3.2. WordNet

WordNet est une base de données lexicale développée depuis 1985 par des linguistes du laboratoire des sciences cognitives de l'université de Princeton. C'est un réseau sémantique de la langue anglaise, qui se fonde sur une théorie psychologique du langage. La première version diffusée remonte à juin 1991 [33].

Son but est de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise. Le système se présente sous la forme d'une base de données électronique qu'on peut télécharger sur un système local. Des interfaces de programmation sont disponibles pour de nombreux langages.

WordNet (figure 10) est distribué sous dernière version distribuée une licence libre, permettant de l'utiliser commercialement ou à des fins de recherche. La dernière version distribuée en avril 2013 est la 3.1. Cette version est par ailleurs consultable en ligne[19].

[19] https://fr.wikipedia.org/wiki/WordNet.

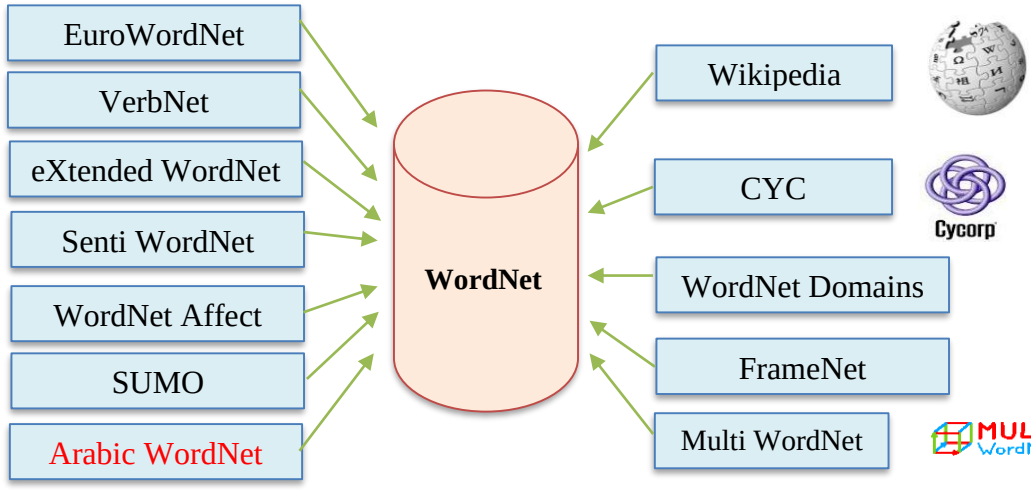


Figure 10 : Ressources descendances de WordNet[20]

On peut considérer WordNet comme un graphe ou un réseau lexicale sémantique, souvent qu'on qualifie d'ontologie légère (Light Ontology), où :

• Les synsets sont les nœuds.

• Les relations sémantiques entre synsets sont les arcs.

4.3.2.1. Synsets

La composante atomique sur laquelle repose le système entier est le synset, c'est un groupe de mots interchangeables, dénotant un sens ou un usage particulier. Par exemple dans la version 2.0 de WordNet, le nom commun anglais « car » est décrit à l'aide de cinq synsets, comme il est montré dans la figure 11 [26].

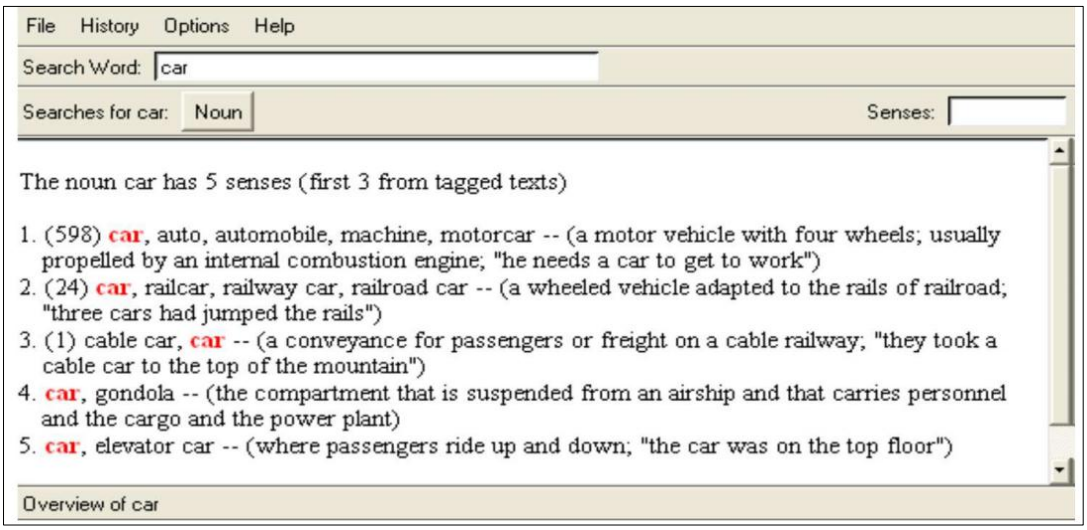


Figure 11 : Synset : différents sens du mot « car »

[20] http://www.mcours.net/cours/pdf/leilclic3/leilclic939.pdf

Chaque Synset dénote une acception différente du mot « car », décrite par une courte définition. Une occurrence particulière de ce mot dénotant par exemple le premier sens (le plus courant), dans le contexte d'une phrase ou d'un énoncé, serait ainsi caractérisée par le fait qu'on pourrait remplacer le mot polysémique par l'un ou l'autre des mots du synset sans altérer la signification de l'ensemble.

4.3.2.2. Les relations sémantiques

Dans WordNet, les concepts sont reliés par des relations sémantiques. La relation de synonymie est la relation de base dans WordNet. Elle relie les termes d'un même nœud. Les nœuds (les concepts ou les synsets) sont reliés entre eux par des relations sémantiques telles que, la relation de composition (partie-tout) et la relation hyponymie-hyperonyme (est-un), comme représentées dans le schéma de la figure 12 [26].

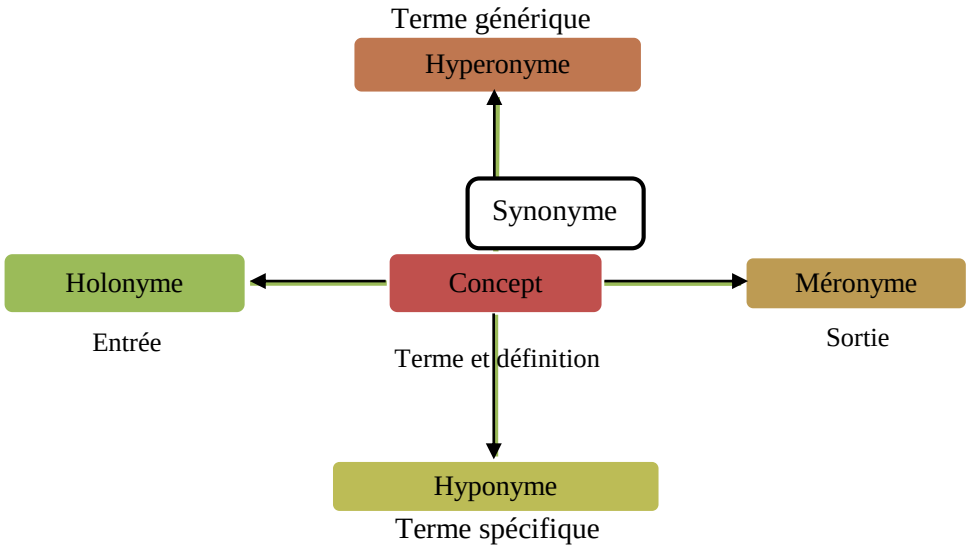


Figure 12 : Rrelations entre les Synsets

5. Modèles de Recherche d'Information

Un modèle de RI a pour rôle de fournir une formalisation du processus de RI et un cadre théorique pour la modélisation de la mesure de pertinence. Il existe un grand nombre de modèles de RI textuelle développé dans la littérature. Ces modèles ont en commun le vocabulaire d'indexation basé sur le formalisme mots clés et diffèrent principalement par le modèle d'appariement requête-document [27].

Le tableau 5 suivant résume les déférents modèles de recherche d'information :

Modèle RI	**Modèles ensemblistes**	Modèle booléen
		Modèle booléen étendu
		Modèle booléen basé sur les ensembles flous
	Modèles algébriques	Modèle vectoriel
		Modèle LSI (Latent SemanticIndexing)
	Modèles probabilistes	Modèle probabiliste
		Modèle de langage

Tableau 5 : Taxonomie des modèles de RI

5.1. Modèle booléen

Le modèle booléen est basé sur la théorie des ensembles. Dans ce modèle, les documents et les requêtes sont représentés par des ensembles de mots clés. Chaque document est représenté par une conjonction logique des termes non pondérés qui constitue l'index du document. Un exemple de représentation d'un document est comme suit : d = t1∧t2∧t3...∧ tn [28].

Une requête est une expression booléenne dont les termes sont reliés par des opérateurs logiques (OR, AND, NOT) permettant d'effectuer des opérations d'union, d'intersection et de différence entre les ensembles de résultats associés à chaque terme. Un exemple de représentation d'une requête est comme suit : " q = (t1∧t2) ∨ (t3∧t4) ". La fonction de correspondance est basée sur l'hypothèse de présence/absence des termes de la requête dans le document et vérifie si l'index de chaque document dj implique l'expression logique de la requête q [28]. Le résultat de cette fonction est donc binaire est décrit comme suit :

$$RSV(q, d) = \{1,0\}.$$

5.2. Modèle vectoriel

Ce modèle représente les documents et les requêtes par des vecteurs dans un espace à n dimensions. Les dimensions étant constituées par tous les termes d'indexation. Les coordonnées d'un vecteur document représentent les poids des termes correspondants [21].

La création d'un index implique une lecture lexicologique pour identifier les termes significatifs, où l'analyse morphologique ramène les différentes formes de mot aux « lemmes » communs, et l'occurrence de ces lemmes est calculée. Des substituts de requête et de document sont comparés selon leurs vecteurs [2].

Par exemple, Soit l'espace vectoriel suivant : $<t_1, t_2, t_3, \ldots, t_n>$ Un document et une requête peuvent être présentés comme suit :

$d = < a_1, a_2, a_3 \ldots a_n >$

$q = < b_1, b_2, b_3 \ldots b_n >$

Ainsi, a_i et b_i correspondent aux poids du terme t_i dans le document et dans la requête. Étant donnés ces deux vecteurs, leur degré de correspondance est déterminé par leur similarité.

Il y a plusieurs façons de calculer la similarité entre deux vecteurs, les plus répandues sont (tableau 6) :

Mesures	Formules
Produit scalaire	$RSV(q,d_i) = \sum_{j=1}^{\|T\|} w_{qj} . w_{ij}$
Mesure de cosinus	$RSV(q,d_i) = \frac{q.d_i}{\|\|q\|\|.\|\|d_i\|\|} = \frac{\sum_{j=1}^{\|T\|} w_{qj} . w_{ij}}{\sqrt{\sum_{j=1}^{\|T\|} {w_{qj}}^2} \text{ x } \sqrt{\sum_{j=1}^{\|T\|} {w_{ij}}^2}}$
Mesure de Dice	$RSV(q,d_i) = \frac{2\ X \sum_{j=1}^{\|T\|} w_{qj} . w_{ij}}{\sqrt{\sum_{j=1}^{\|T\|} {w_{qj}}^2 + \sum_{j=1}^{\|T\|} {w_{ij}}^2}}$
Mesure de Jaccard	$RSV(q,d_i) = \frac{\sum_{j=1}^{\|T\|} w_{qj} . w_{ij}}{\sqrt{\sum_{j=1}^{\|T\|} {w_{qj}}^2 + \sum_{j=1}^{\|T\|} {w_{ij}}^2 - \sum_{j=1}^{\|T\|} w_{qj} . w_{ij}}}$

Tableau 6 : Mesures de similarités utilisées dans le modèle vectoriel [2]

5.3. Modèle probabiliste

Ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête. Le principe de base consiste à retrouver des documents qui ont en même temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents. Etant donné une requête utilisateur *Q* et un document *D*, il s'agit de calculer la probabilité de pertinence du document pour cette requête [27].

Deux possibilités se présentent : *R, D* est pertinent pour q et D, n'est pas pertinent pour q. Les documents et les requêtes sont représentés par des vecteurs booléens dans un espace à n dimensions. Un exemple de représentation d'un document dj et une requête q est le suivant :

dj = (w1, j, w2, j, w3, j, .., wn, j),

q = (w1, q, w2, q, w3, q, .., wn, q). Avec wk, j ∈ [0 1] et wk, q ∈ [0 1].

La valeur de wk, j (resp. wk, q) indique si le terme t_k apparaît ou non dans le document *dj* (resp. q).

Le modèle probabiliste évalue la pertinence du document dj pour la requête q. Un document est sélectionné si la probabilité que le document d soit pertinent, notée *p(R/D),* est supérieure à la probabilité que d soit non pertinent pour q, notée *p(R/D)* où *R* est l'événement de pertinence et est l'événement de non pertinence.

Le score d'appariement entre le document d et la requête *Q*, noté RSV (Q, D) est donné par : $(,) = \frac{(/)}{(/)}$

Ces probabilités sont estimées par de probabilités conditionnelles selon qu'un terme de la requête est présent, dans un document pertinent ou dans un document non pertinent. Cette mesure de similarité entre la requête et les documents peut se calculer par différentes formules [27].

5.4. Modèle LSI (Latent Semantic Indexing)

Le modèle LSI est une variante du modèle vectoriel. Il présente en effet une extension de ce dernier. Il se distingue par la transformation de la représentation traditionnelle par mots-clés en une représentation plus conceptuelle, plus sémantique, qui vise à favoriser le rapprochement de documents et requêtes sémantiquement similaires.

Ce modèle a été proposé comme solution au problème de bruit constaté au niveau de la représentation vectorielle traditionnelle basée uniquement sur les mots (i.e. contient des termes non représentatifs du contenu textuel). En fait, le modèle propose, en s'appuyant sur une décomposition en valeurs singulières de la matrice pondérée classique d'occurrences des termes d'indexation dans les documents, de créer un espace vectoriel plus petit. Les dimensions de l'espace réduit ne sont plus représentées par les termes mais par une combinaison linéaire de ces derniers. Ces combinaisons sont susceptibles de mieux faire ressortir les affinités sémantiques latentes entre les mots et, par conséquent, de mieux exprimer les concepts contenus dans les documents. L'utilisation du modèle LSI en RI consiste à traduire la requête de l'utilisateur dans ce nouvel espace. L'appariement d'un document et d'une requête revient alors à appliquer une mesure de similarité standard (exemple : la mesure du cosinus) entre les vecteurs dans l'espace réduit. Les documents peuvent, comme dans le modèle vectoriel, être classés selon leur pertinence par rapport à la requête [21].

6. Evaluation des Systèmes de Recherche d'Information

L'aspect évaluation occupe une place très particulière en RI. L'évaluation des systèmes de recherche d'information constitue une étape primordiale dans l'élaboration d'un modèle de

recherche d'information. Elle permet de caractériser le modèle et de fournir des éléments de comparaison entre les modèles existants. D'une façon générale, un système de recherche d'information idéal a deux objectifs [29] :

- Retrouver tous les documents pertinents,

- Rejeter tous les documents non pertinents.

Ces deux objectifs sont évalués par des mesures de précision et de rappel que nous définissons ci-après (figure 13).

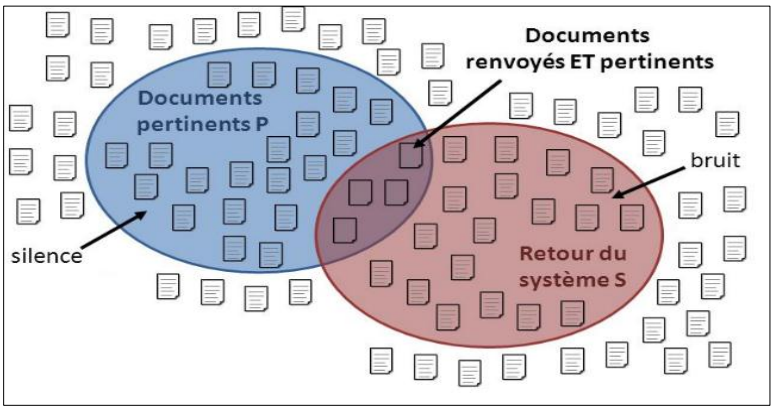


Figure 13 : Précision et Rappel[21]

6.1. Rappel et Précision

Les mesures de précision/rappel sont obtenues en partitionnant l'ensemble des documents restitués par le SRI en deux catégories : les documents pertinents et les documents non pertinents. Ces deux catégories se définissent comme suit :

- Le rappel mesure la proportion de documents pertinents restitués parmi tous les documents pertinents disponibles. Si le rappel vaut 1 c'est que les documents pertinents disponibles ont tous été restitués par le système, inversement si le rappel vaut 0 c'est qu'aucun document pertinent n'a été restitué. Cette mesure permet aussi de déterminer le **silence**, c'est-à-dire la proportion de documents pertinents non trouvés [30], elle est définie par S=1-R ou R est le rappel du SRI.

- La précision mesure la proportion de documents pertinents restitués parmi tous les documents restitués. Elle mesure la capacité du système à trouver exclusivement des documents pertinents. La précision vaut 1 quand tous les documents restitués sont pertinents. Elle vaut 0 si aucun des documents restitués n'est pertinent. Cette mesure détermine également le **bruit**, c'est-à-dire la

[21] https://vdocuments.net/xavier-tannier-xaviertannierlimsifr-recherche-evaluation-indexation-et-recherche-dinformation.html

proportion de documents non pertinents restitués par le système [30], elle est définie par B=1-P ou p est la précision du SRI.

Ces deux mesures peuvent être définies par :

$$\text{Précision} = \frac{nbr\ de\ documents\ pertinents\ retrouvés}{nbr\ de\ documents\ retrouvés}$$

$$\text{Rappel} = \frac{nbr\ de\ documents\ pertinents\ retrouvés}{nbr\ de\ documents\ pertnents}$$

La valeur de ces taux est influencée par le processus d'indexation. En effet, plus l'indexation est exhaustive, plus le taux de rappel est potentiellement important : toutes les informations susceptibles d'être pertinentes peuvent être restituées, mais certaines ne seront pas pertinentes pour l'utilisateur. Symétriquement, plus l'indexation est spécialisée, plus le taux de précision est élevé, mais cela induit le risque d'avoir un taux de rappel faible : les informations restituées seront pertinentes, mais d'autres informations pertinentes ne seront pas restituées.

6.2. Courbe de Rappel/Précision

Les performances d'un système de RI peuvent être représentées par une courbe Rappel/ Précision. Lorsque les valeurs exactes de rappel ne peuvent pas être atteintes, Il est fréquent d'employer une interpolation sur ces courbes, qui consiste à lisser la courbe initiale pour qu'elle soit décroissante.

Il y a une forte relation entre les taux de précision et les taux de rappel : quand l'une augmente l'autre diminue (figure 14). En pratique, la précision évolue en fonction du rappel et vice versa [17].

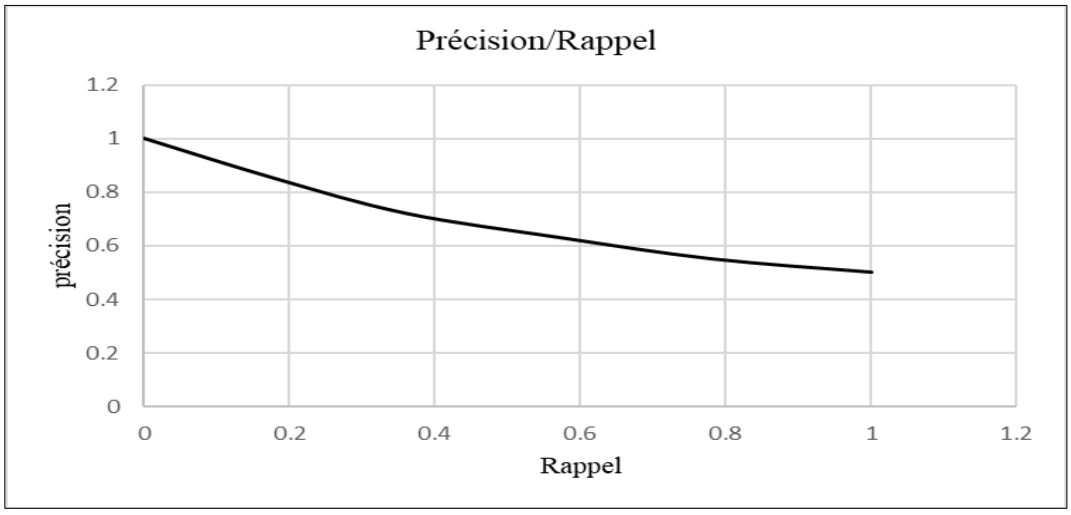


Figure 14 : Courbe typique Précision/Rappel

7.3. F-mesure

Examiner la succession des précisions à n ou des rappels à n permet d'évaluer un ordonnancement dans son ensemble et ainsi de déterminer telle ou telle propriété de l'algorithme étudié. Il peut être intéressant d'avoir une valeur synthétisant ces deux mesures. On voudrait pouvoir maximiser la précision et le rappel, mais comme on l'a vu ces deux mesures évoluent souvent de façon opposée. La précision est globalement décroissante au fur et à mesure que le SRI restitue des documents, alors que le rappel est globalement croissant. On peut choisir la mesure F comme valeur synthétique exploitant la précision et le rappel. Elle est calculée comme suit [13] :

$$\mathrm{F} = 2 \times \frac{Rappel \times Précision}{Rappel + Précision}$$

7. Conclusion

Dans ce chapitre nous avons fait un rappel des notions et concepts en Recherche d'Information et nous avons décrit plus particulièrement les principales étapes à savoir l'indexation (classique et sémantique) et la recherche. Nous avons introduit les principaux modèles sur lesquels se basent les SRI, et de présenter les méthodes d'évaluation à optées pour attester de la validité des mécanismes implémentés au cœur de ces systèmes.

Chapitre IV :
Réalisation du Système de Traduction de Contexte De et Vers la Langue Arabe

Chapitre IV :

Réalisation du système de traduction de contexte de et vers la langue arabe

1. Introduction

Après avoir étudié les concepts de base et se familiariser avec la traduction automatique, permettant la conception d'un système de traduction de contexte, nous allons tout au long de ce chapitre présenter notre démarche pour la réalisation d'un tel système ayant la capacité de traduire de et vers la langue arabe et ce, par contexte. Cette tâche n'a jamais été facile car elle présente de nombreux défis et problèmes que nous présentons au fur et à mesure de son développement. Ensuite, Nous procédons à une description générale de nos approches en abordant l'ensemble des étapes par lesquelles notre système doit passer. Nous finalisons par présenter les détails conceptuels qui contribueront à la composition du système.

2. Architecture du notre système

Les modèles de langage statistiques décrivent statistiquement les contraintes sur l'ordre des mots trouvés dans le langage. Ils sont généralement utilisés dans un nombre d'applications telles que la traduction automatique. Cependant, il est difficile de construire un modèle de langage en absence de grands corpus modélisant le langage considéré. De tels corpus tendent à contenir, pour chaque mot, le maximum de contextes possibles permettant ainsi de calculer des modèles de langage robustes et précis.

Avec le développement de l'Internet et de ses services, le Web est devenu une immense source de documents de toutes natures dans différentes langues et différents domaines. Cette source alliée à des supports de stockage permet la constitution rapide de corpus.

Bien qu'elle soit parlée par plus de 500 millions de personnes dans le monde, selon des statistiques publiées par Wikipédia en 2021, l'arabe ne dispose pas d'assez de ressources, spécialement pour le problème de l'apprentissage statistique. Ce n'est qu'en ces dernières années que la communauté des chercheurs a commencé à s'intéresser à cette langue et à fournir plus d'effort.

La collecte d'un corpus parallèle pour la langue arabe prend beaucoup de temps, et compte tenu du fait que la qualité des données est étroitement liée à la qualité de la traduction, la tâche est ardue.

En raison des contraintes de temps, nous avons opté pour des corpus provenant du projet OPUS « Open parallel CorpUS », une collection grandissante de textes traduits sur le Web en différentes langues, dont l'arabe. Dans ce projet, les spécialistes sont arrivés à convertir et aligner des données en ligne gratuites, à ajouter des annotations linguistiques et à fournir un corpus parallèle accessible au public, dans des domaines variés (juridique, réglementaire, diplomatique, technique, sous-titre de films, etc.). L'OPUS est basé sur des produits open source et le corpus est également livré sous forme de package de contenu ouvert. Plusieurs outils sont utilisés pour compiler la collection actuelle. Tout le prétraitement est effectué automatiquement. Aucune correction manuelle n'a été effectuée[22].

La figure 15, illustre un organigramme de notre projet. Il se compose de 3 phases :

1- Collection et préparation des données ;
2- Traitement de corpus ;
3- Prétraitement de la requête.

Dans cette section, nous allons expliquer en détails les principaux éléments de ce système.

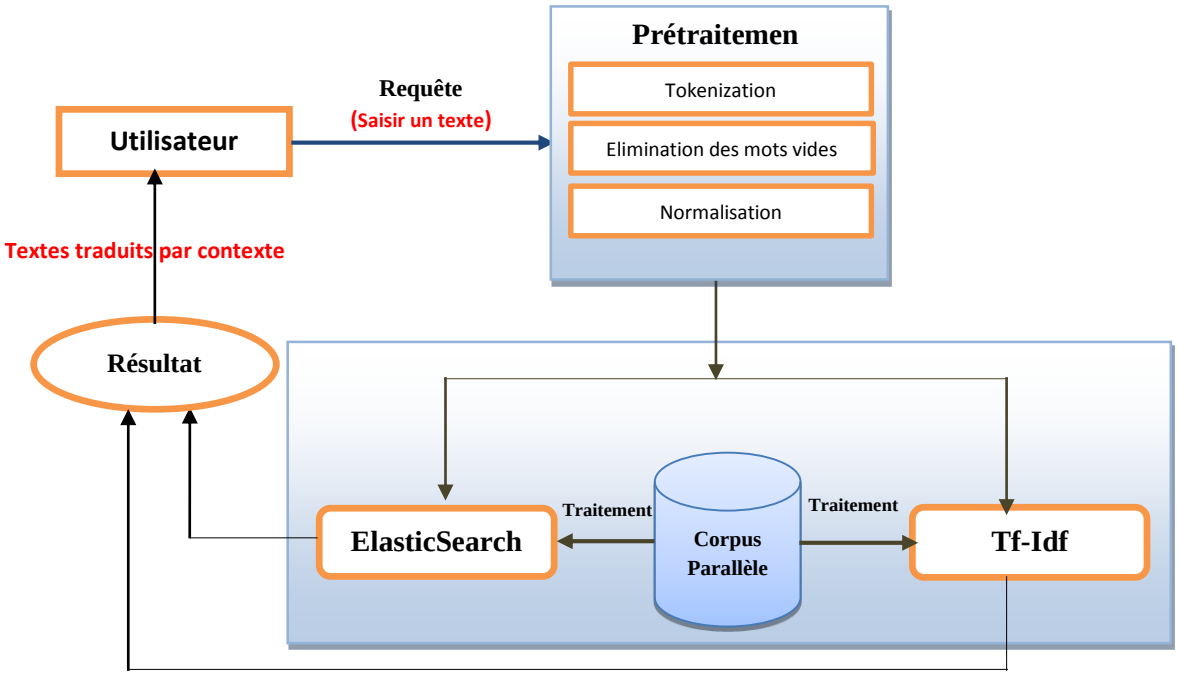


Figure 15 : Organisation des modules de notre système de traduction de contexte de et vers la langue arabe

[22] https://opus.nlpl.eu/

2.1. Collection et préparation de corpus

2.1.1. Collection du corpus

L'objectif principale du système est de réaliser une traduction de contexte de et vers la langue arabe en exploitant une grosse masse des données disponible, sous forme de Corpus parallèle (ou *bitexte*). Ce système agit comme un moteur de recherche du contexte de mots ou d'expression dans ce Corpus, connaissant ainsi le but de ces significations et exploitant leur véritable sens.

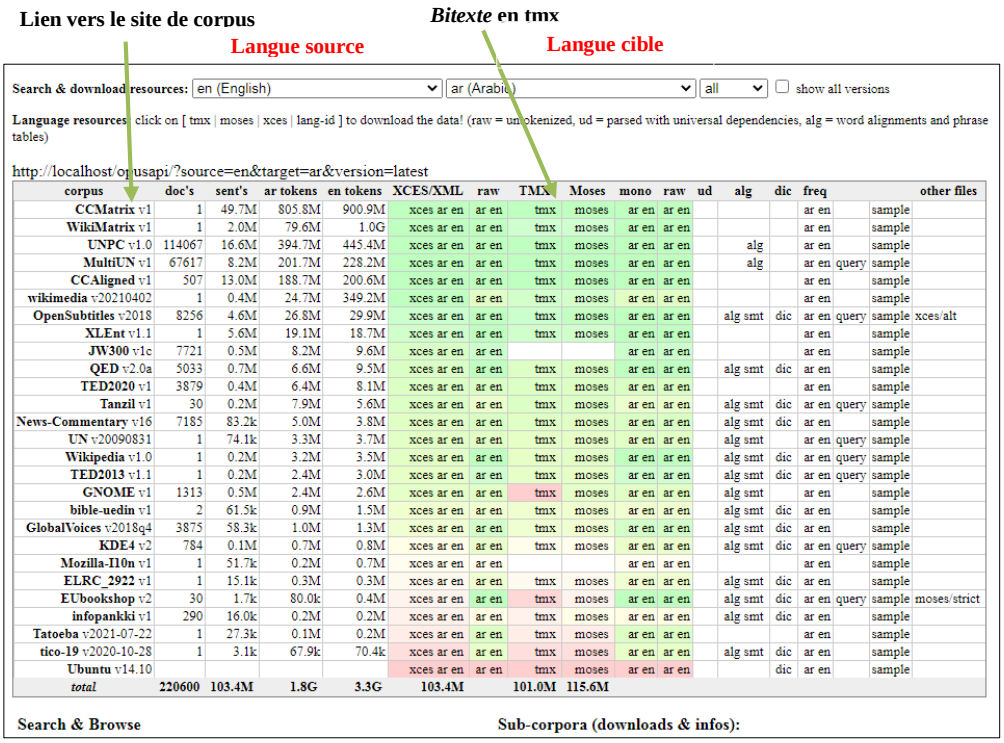
Six(6) corpus, à caractère politique, ont été téléchargés depuis le site *https://opus.nlpl.eu/* : ensemble de corpus de textes avec des phrases alignées qui permettent de rechercher et d'analyser des traductions entre plusieurs langues (tableau 7). Chaque corpus représente une paire de langues ; la langue arabe et l'une des langues suivantes : anglais, français, espagnol, allemand et chinois. Chaque corpus est sous format « tmx » (Translation Memory Exchange), de taille 5 Giga Octets (environ 8000 millions de lignes).

Étant donné que ce système a besoin de performances élevées du système d'exploitation et de taille suffisante de mémoires pour traiter cette énorme quantité de textes, nous nous sommes concentrés sur l'utilisation du corpus arabe-anglais (figure 16) et ce, il peut s'appliquer à tous les autres Corpus sans exception.

OPUS parallel corpora - the number of aligned tokens for each pair of languages.	Afrikaans	Arabic	Bulgarian	Bosnian	Czech	Danish	German	Greek	English	Spanish	Estonian	Persian	Finnish	French	Hebrew	Hindi	Croatian	Hungarian	Italian	Japanese	Korean	Lithuanian	Latvian	Macedonian	Dutch	Norwegian	Polish	Portuguese	Portuguese (Brazil)	Romanian	Russian	Slovak	Slovenian	Albanian	Serbian	Swedish	Turkish	Ukrainian	Chinese Simplified	Chinese Traditional
Afrikaans		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.7	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.3	0.3	0.3	0.3	0.2	0.2	0.3	0.3	0.3		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0	0.1		0.3	0.3	0.3	0.3
Arabic	0.3		0.5	2k	0.5	0.6	5.6	0.6	251.9	259.1	0.6	0.4	0.5	257.2	0.3	0.4	0.4	0.5	0.6	0.5	0.3	0.4	0.4	0.3	0.5	0.1k	0.6	0.5	0.5	0.4	262.2	0.5	0.5	0.6k	0.2	6k	0.5	0.5	249.9	0.6
Bulgarian	0.3	0.6		10.8	48.8	28.1	23.8	52.9	80.1	58.6	28.1	0.4	38.9	44.2	0.4	0.4	28.6	48	31.6	1	0.4	15.8	15.2	6.8	51.7	7.2	45.3	50.8	43.4	65.2	10.4	21.6	39.5	7.2	35.1	33	40.1	0.9	2.8	0.6
Bosnian	0	3k	10.5		6.7	2.4	1.6	10	13.9	8.7	2.7	0.01	4.7	5.7	4k		9.1	6.7	3.1	0.1	3k	0.2	0.1	3.3	7.6	1.5	5.9	7.1	8.4	12.1	1.9	1.5	5.2	3.8	10.7	3.9	10		0.5		
Czech	0.3	0.5	47.8	6.9		33	29.1	50.8	84	64.5	30.6	0.4	42.3	34.2	0.4	0.4	25.2	53.7	36.3	1.1	0.4	17.7	16.3	1.3	59	9.1	55.1	54.5	44.7	57.2	10.7	28	43.2	2	29.6	36.5	35.9	1	2.8	0.6
Danish	0.3	0.7	28.2	2.7	34.6		59.4	51.2	80.3	73.4	26.1	0.5	68.7	20.5	0.4	0.5	11.8	32.7	62	1.1	0.5	17.4	16.6	1.1	75.2	10.5	32.2	70.9	18.8	31.2	5.4	20.1	31.9	0.9	12.4	62.1	15.7	1.9	2.2	0.9
German	0.3	6.4	23.9	1.8	30.7	59.5		47.2	85.3	78	22.3	0.5	59.9	77.5	0.4	0.5	8.9	28.4	65.8	1.7	0.5	16.6	15.6	0.7	69.3	6.2	29.3	69.3	14.1	25.4	11.8	19.8	24.3	0.3	8.7	55.9	11.9	2.1	7.7	0.9
Greek	0.4	0.8	54.5	10.7	53.2	53	49		105.3	39.2	30.3	0.6	36.9	39.5	0.5	0.5	5.8	17.7	58.2	1.4	0.5	18	17.2	6.9	79.6	8.2	47.9	77.6	45.6	62.7	10.6	22.9	40.5	7.9	34.2	54.5	39.4	1.3	2.9	0.9
English	0.7	290.9	95.4	17.2	103.5	87.8	91.6	119.4		550.4	44.6	5.4	110	471.7	0.5	0.4	58.2	100.8	108	2.5	0.5	18.7	17.1	7.4	153.3	18.5	97	149.2	119.2	120.4	358.4	32.8	70.3	9.3	70.6	93.5	80.9	2.9	289.2	0.9
Spanish	0.3	317.3	62.3	9.7	70.2	77.2	82.3	39.1	548.9		36	0.6	89	422.7	0.5	0.5	31.9	68.4	88	2.3	0.5	18.9	17.5	1.7	116.1	12	62.6	121.7	87.3	76.6	342.5	28	51.7	2.5	41.4	79.3	48.2	2.6	310.2	1
Estonian	0.3	0.6	25.5	2.8	28.5	23.1	19.6	26.4	35.9	30.7		0.5	25.4	26.7	0.4	0.4	9.7	27	22	1	0.4	13.7	14.8	0.9	30.1	3.7	27.1	29.6	15.4	27.3	5.1	17.1	26.6	0.9	10.4	21.3	12.8	2	1.7	0.8
Persian	0.3	0.6	0.6	0.03	0.5	0.6	0.6	0.7	5.2	0.6	0.6		0.6	0.6	0.4	0.4	0.4	0.6	0.6	0.6	0.4	0.5	0.4	0.4	0.6	28k	0.6	0.6	0.6	0.5	0.6	0.5	0.6		0.2	0	0.6	0.6	0.6	0.6
Finnish	0.3	0.5	32.1	4	35.8	54.1	47.1	27.9	76.7	66.5	23.5	0.4		60.6	0.3	0.4	14.6	34.3	51.7	0.8	0.3	15.2	14	1.1	65.8	10	39.1	64.3	24	36.3	5.6	18.3	31.2	1.4	16.2	54.4	19.5	0.7	1.8	0.5
French	0.4	361.5	52	6.9	40.9	23.3	92.1	44.8	546.6	481.1	35.5	0.7	90.9		0.6	0.6	24.8	60.5	96.5	2.5	0.7	21.2	19.7	1.4	206.3	11.4	53.9	114	55.2	60.5	424.7	28.6	47.2	1.8	29.5	82.2	37.2	2.8	352.1	1.3
Hebrew	0.2	0.3	0.3	2k	0.3	0.4	0.4	0.3	0.4	0.3	0.3	0.3	0.3	0.4		0.3	0.3	0.3	0.4	0.3	0.2	0.3	0.3	0.3	0.3	1k	0.4	0.3	0.3	0.3	0.4	0.3	0.3	0.4k	0.2	6k	0.3	0.3	0.3	0.3
Hindi	0.7	1.4	1.2		1.3	1.5	1.5	1.5	1.5	1.4	1.5	1.1	1.3	1.5	1		1	1.3	1.5	1.4	0.9	1.1	1.1	1	1.4		1.4	1.4	1.4	1.1	1.4	1.2	1.3		0.5		1.4	1.4	1.4	1.4
Croatian	0.3	0.4	27.8	9.1	24.8	10.7	7.9	5.1	45.9	28.8	9.5	0.4	17.3	20.5	0.3	0.3		23.5	12.7	0.7	0.3	0.9	0.6	6	27.8	6.3	23.3	25.6	27.7	33.6	6.6	4.5	21.8	6	33.9	14.4	27.7	0.6	1.8	0.4
Hungarian	0.3	0.6	45.3	6.5	51.6	30.4	26.4	15.9	78.6	60.6	28.2	0.5	39	48.2	0.4	0.4	22.6		35.6	1.1	0.5	17.1	16.8	1.2	56.2	7.8	45	52.7	45.4	54.3	10.5	22.7	39.1	1.9	26.4	34.1	31.9	1	2.9	0.7
Italian	0.4	0.8	35.2	3.6	41.3	67.3	70.7	61.1	105.3	90	27.9	0.6	70.8	87.8	0.5	0.5	14.6	42		1.3	0.5	19	18.3	0.9	84.9	6.8	39.4	86.4	29.9	39.9	10.8	24.1	32	1	15.8	61.6	21.8	2.4	1.8	1
Japanese	0.4	0.9	1.3	0.1	1.4	1.5	2.3	1.7	3	2.8	1.5	0.7	1.3	2.7	0.6	0.7	1	1.4	1.5		0.6	0.9	0.8	0.6	1.8	0.2	1.6	1.8	1.8	1.2	1.9	1.1	1.3	0.1	0.7	1.6	1.5	1.3	1.8	1.1
Korean	0.2	0.3	0.4	2k	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.3	0.4	0.3	0.3	0.3	0.4	0.4	0.4		0.3	0.3	0.3	0.4		0.4	0.4	0.4	0.3	0.4	0.3	0.4		0.2		0.4	0.4	0.4	0.4
Lithuanian	0.3	0.5	14.3	0.2	16.9	16.5	15.8	15.7	17.1	16.8	14.8	0.4	16.6	16.5	0.3	0.4	0.9	16.4	16	0.6	0.4		15	0.4	16.3	0.3	16.9	16.7	1.5	14.2	1	15.8	16.4	0.1	0.9	14.3	1.4	0.6	0.6	0.6
Latvian	0.2	0.4	13.5	0.1	15.5	15.6	14.9	14.9	15.7	15.5	16	0.3	15.3	15.3	0.3	0.3	0.6	16	15.4	0.5	0.3	15		0.3	15.5	0.1	15.2	15.4	0.8	13.6	0.6	14.9	15.3	0.1	0.5	13.8	0.8	0.5	0.6	0.5
Macedonian	0.3	0.4	6.8	3.5	1.4	1	0.7	6.7	7.1	1.6	0.9	0.4	1.3	1.2	0.3	0.3	6.4	1.3	0.9	0.4	0.3	0.4	0.3		1.4	0.4	1.6	1.5	1.4	6.8	0.7	0.6	1.5	5.4	6.7	0.8	6.6	0.4	0.5	0.4
Dutch	0.3	0.7	54.1	8.5	63.1	77.6	71.2	80.6	137.6	114.5	34.3	0.5	86.2	181.8	0.4	0.4	30.5	62.8	81.1	1.4	0.5	18	17.2	1.5		12.3	59	108.2	56.2	67.5	12.2	24.7	51.7	2.4	35.4	76.6	40.5	2	3.8	0.9
Norwegian		2k	7.5	1.5	9.5	10.2	5.8	8.4	14.7	11.4	3.8	13k	12.7	10	3k		6.6	8.6	6.3	0.2		0.3	0.2	0.4	12		9.5	10.4	8.7	9.2	2.3	1.3	7.4	0.5	6.7	13.1	8.5	0.1	0.6	
Polish	0.3	0.7	42.8	5.6	53.6	30.6	27.9	44.5	76.1	55.7	28.9	0.5	44.9	43.5	0.4	0.4	22.2	45.4	34.3	1.2	0.5	18.2	16.5	1.4	53.5	8.7		48.7	34.4	50.8	8.2	22.4	39.5	1.7	24.6	32.7	28.3	1.8	2.3	0.9
Portuguese	0.3	0.8	55.2	7.9	60.5	76.6	75.2	81.5	139.3	123.7	36.1	0.6	87.7	103.8	0.5	0.5	28.5	60.4	86.2	1.6	0.5	20	18.5	1.6	111.8	11	55.8		68.4	67.9	11.9	27.4	49.3	2.5	33.9	75.2	40.6	3	3.7	1.1
Portuguese (Brazil)	0.3	0.8	46.8	9.3	48.2	19	13.7	47.8	101.4	86.3	17.2	0.6	31.2	49.5	0.5	0.5	30.3	51.3	28	1.6	0.5	1.7	1	1.5	55.9	9	39.3	67.5		61.5	15.2	8.9	31.2	2.5	39.3	25.1	45.3	3	4	1.1
Romanian	0.3	0.5	70.9	13.6	63.5	33.2	27	66.4	108.9	78.7	32.1	0.4	47.8	56	0.4	0.4	38.1	62.7	38.9	1	0.4	16.5	15.9	7.1	69.8	9.9	58.4	68.5	63.4		12.8	24.2	49.2	8	47.5	40.2	51.5	0.9	3.2	0.5
Russian	0.3	274.9	10.8	2	11.1	5.1	10.8	10.6	321.8	293.5	5.3	0.5	7	316.2	0.4	0.4	6.9	11.4	9.6	1.3	0.4	1	0.7	0.7	11.7	2.3	9	11.1	14.5	12.1		2.7	7.2	0.5	8.2	6.3	11.1	1.6	242.2	0.8
Slovak	0.3	0.6	20.7	1.5	28.1	19.8	19.3	21.1	29.4	25.8	19.1	0.4	21.1	23.2	0.4	0.4	4.7	23.2	21.1	0.7	0.4	16.5	15.6	0.6	23.4	1.3	22.2	24.2	8.4	22	2.7		23	0.5	0.2	18.5	7.1	0.6	1	0.6
Slovenian	0.4	0.6	38.2	5.1	43	30.8	23.9	38.4	57	47.1	29	0.5	36.7	38.9	0.4	0.4	21.5	40.2	28.3	1	0.4	17.8	16.8	1.4	47.9	6.9	40	43.7	27.9	43.7	6.8	23.4		1.6	0.3	31.1	23.2	0.9	2.3	0.7
Albanian	0	1k	7.8	4.2	2.3	1	0.3	8.2	9.3	2.7	1.1		1.9	1.7	0.9k		6.8	2.3	1	0.1		0.1	0.1	5.7	2.5	0.6	2	2.6	2.7	8.1	0.6	0.6	1.9		5.9	1.4	7.6	0	0.2	0.3
Serbian	0.1	0.3	34.5	10.7	29.6	11.4	7.9	32.9	56.2	38	10.5	0.2	19.5	24.8	0.2	0.2	34.2	27.7	13.9	0.6	0.2	1	0.6	6.3	32.6	6.5	26.1	30.8	36.2	42.5	7.9	0.3	0.3	5.3		16.1	33.9	0.7	2	
Swedish		7k	31.5	4	36	59	52.9	50.2	79.3	71.8	22.4		65.6	66.2	11k		14.8	34.8	54	1.1		14.4	13.8	0.8	70.4	12.7	32.8	66.3	23.3	35.5	6.1	17.9	30.5	1.3	16.2		19.2	0.1	1.9	4k
Turkish	0.3	0.6	36.4	9.3	32.8	13.3	9.7	34.8	58.6	40.1	11.8	0.5	21.4	28.3	0.4	0.4	25.7	30.6	17.3	1.1	0.4	1.3	0.8	5.9	34.1	7.5	27.4	33.8	38.2	42.2	9.8	6.2	21.9	6.3	31	17.3		1	2.7	0.7
Ukrainian	0.3	0.7	0.9	0	1.1	2	2.2	1.2	2.9	2.4	2.5	0.5	0.8	2.3	0.4	0.5	0.6	1.1	2.3	0.9	0.5	0.6	0.6	0.4	2	0.1	1.9	2.7	2.7	0.9	1.8	0.7	1		0.7	0.1	1.1		0.9	1
Chinese Simplified	0.4	275.3	3.8	0.7	3.9	2.6	7.5	3.8	273.8	277.9	2.2	0.5	2.9	275.2	0.4	0.5	2.5	4	2	1.3	0.5	0.7	0.7	0.5	4.8	0.8	3.2	4.2	4.8	3.9	275.9	1.2	3.2	0.3	2.7	2.5	3.9	1		0.9
Chinese Traditional	0.2	0.5	0.4		0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.3	0.4	0.6	0.3	0.3	0.3	0.5	0.6	0.5	0.3	0.4	0.4	0.3	0.6		0.6	0.6	0.6	0.4	0.5	0.4	0.5	0.2		1k	0.5	0.6	0.5	

Tableau 7 : Tableau comparatif des corpus parallèles d'OPUS[23]

[23] https://www.sketchengine.eu/opus-parallel-corpora/



corpus	doc's	sent's	ar tokens	en tokens	XCES/XML	raw	TMX	Moses	mono	raw	ud	alg	dic	freq			other files
CCMatrix v1	1	49.7M	805.8M	900.9M	xces ar en	ar en	tmx	moses	ar en	ar en				ar en		sample	
WikiMatrix v1	1	2.0M	79.6M	1.0G	xces ar en	ar en	tmx	moses	ar en	ar en				ar en		sample	
UNPC v1.0	114067	16.6M	394.7M	445.4M	xces ar en	ar en	tmx	moses	ar en	ar en		alg		ar en		sample	
MultiUN v1	67617	8.2M	201.7M	228.2M	xces ar en	ar en	tmx	moses	ar en	ar en		alg		ar en	query	sample	
CCAligned v1	507	13.0M	188.7M	200.6M	xces ar en	ar en	tmx	moses	ar en	ar en				ar en		sample	
wikimedia v20210402	1	0.4M	24.7M	349.2M	xces ar en	ar en	tmx	moses	ar en	ar en				ar en		sample	
OpenSubtitles v2018	8256	4.6M	26.8M	29.9M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en	query	sample	xces/alt
XLEnt v1.1	1	5.6M	19.1M	18.7M	xces ar en	ar en	tmx	moses	ar en	ar en				ar en		sample	
JW300 v1c	7721	0.5M	8.2M	9.6M	xces ar en	ar en			ar en	ar en				ar en		sample	
QED v2.0a	5033	0.7M	6.6M	9.5M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en		sample	
TED2020 v1	3879	0.4M	6.4M	8.1M	xces ar en	ar en	tmx	moses	ar en	ar en				ar en		sample	
Tanzil v1	30	0.2M	7.9M	5.6M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en	query	sample	
News-Commentary v16	7185	83.2k	5.0M	3.8M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en		sample	
UN v20090831	1	74.1k	3.3M	3.7M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt		ar en	query	sample	
Wikipedia v1.0	1	0.2M	3.2M	3.5M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en	query	sample	
TED2013 v1.1	1	0.2M	2.4M	3.0M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en	query	sample	
GNOME v1	1313	0.5M	2.4M	2.6M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt		ar en		sample	
bible-uedin v1	2	61.5k	0.9M	1.5M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en		sample	
GlobalVoices v2018q4	3875	58.3k	1.0M	1.3M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en		sample	
KDE4 v2	784	0.1M	0.7M	0.8M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en	query	sample	
Mozilla-I10n v1	1	51.7k	0.2M	0.7M	xces ar en	ar en			ar en	ar en				ar en		sample	
ELRC_2922 v1	1	15.1k	0.3M	0.3M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en		sample	
EUbookshop v2	30	1.7k	80.0k	0.4M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en	query	sample	moses/strict
infopankki v1	290	16.0k	0.2M	0.2M	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en		sample	
Tatoeba v2021-07-22	1	27.3k	0.1M	0.2M	xces ar en	ar en	tmx	moses	ar en	ar en				ar en		sample	
tico-19 v2020-10-28	1	3.1k	67.9k	70.4k	xces ar en	ar en	tmx	moses	ar en	ar en		alg smt	dic	ar en		sample	
Ubuntu v14.10					xces ar en	ar en	tmx	moses	ar en	ar en			dic	ar en		sample	
total	**220600**	**103.4M**	**1.8G**	**3.3G**	**103.4M**		**101.0M**	**115.6M**									

Figure 16 : Ressources pour Corpus Anglais-Arabe avec plus d'un million de phrases parallèles « OPUS »[24]

- **Le format tmx :** est un format universel d'échange de données dédié aux applications de traduction assistée par ordinateur TAO (en anglais : Computer-Aided Translation CAT). Les fichiers tmx contiennent une mémoire de traduction, une base de données contenant des fragments de texte originaux et traduits. La mémoire de traduction facilite le processus de traduction car elle est utilisée dans les recherches automatiques en tant que base de données de fragments précédemment traduits. Les fichiers tmx utilisent le format de données xml et peuvent être exportés et importés à l'aide de divers outils de TAO[25].

- **Editeur de format tmx** :

Il existe plusieurs éditeurs capables de lire les fichiers au format tmx pour les Systèmes d'exploitation Windows, dont : OmegaT, SDL Trados Studio,Heartsome TMX editor, etc.

Ces éditeurs ne peuvent pas ouvrir notre Corpus. Parmi les problèmes rencontrés dans le choix de l'éditeur, on retrouve :

- Ne supporte pas les fichiers volumineux.
- Endommagement du fichier tmx en train de s'ouvir.

[24] https://opus.nlpl.eu/
[25]https://www.file-extension.org/

- Connexions erronées du fichier tmx dans les inscriptions du registre.
- Elimination de la description de l'extension tmx du registre Windows par hasard.
- Installation incomplète de l'application desservant le format tmx.
- Le fichier tmx en train de s'ouvrir est infecté par un logiciel non désiré et dangereux.
- L'ordinateur a trop peu de ressources d'équipement pour se débrouiller avec l'ouverture du fichier tmx.
- Les pilotes de l'équipement utilisé par l'ordinateur pour ouvrir le fichier tmx ne sont pas mis à jour.

Après plusieurs tentatives de recherche et expérimentation, nous avons choisi l'éditeur EmEditor, avec lequel nous avons pu lire le Corpus.tmx sans aucun problème.

L'EmEditor est un éditeur de texte pour Windows, prenant en charge des fichiers extrêmement volumineux (jusqu'à 248 Go ou 2,1 milliards de lignes), des fichiers binaires, la comparaison, l'édition en plusieurs lignes, une numérotation facile, des macros pouvant être scriptées, des fichiers tmx, Json, CSV, etc. EmEditor est impressionnant par la rapidité avec laquelle il a la capacité de charger les fichiers volumineux - il peut par exemple ouvrir un fichier de 1,34 Go en moins de neuf (9) secondes et être prêt pour l'édition tout de suite après[26].

2.1.2. Préparation de corpus :

Après avoir été téléchargé et lu, le Corpus structuré peut être converti en formats Json et Csv pour une meilleure utilisation.

- Format Json est un format de représentation logique de données, hérité de la syntaxe de création d'objets en JavaScript. C'est un format réputé léger (il ne contient pas trop de caractères de structuration), assez facilement lisible par les humains, facilement *parsable* par les machines, et indépendant des langages qui l'utilisent (sa seule fonction est de décrire des données, qui sont ensuite utilisées différemment pour chaque cas suivant le contexte).

Exemple : un fichier json simple

```
{
 "nom" : "Mohamed",
          "prenom" : "Kamel",
          "age" : 50,
          "etat" : "Tamanrasset"
        }
```

[26] https://progsoft.net/

Bien que Json puise sa syntaxe du JavaScript, il est indépendant de tout langage de programmation. Il peut ainsi être interprété par tout langage à l'aide d'un *parser*[27].

Afin de convertir notre Corpus tmx au format Json, on a créé deux (2) fonctions, l'une est basée sur la méthode d'itération (figure 17) et la deuxième est basée sur Mmap (figure 18). La méthode d'itération s'avère la plus efficace pour ça flexibilité, garantie, rapidité d'exécution et d'autres parts elle ne nécessite pas une performance élevée du système.

Une fois le fichier converti en Json, sa taille est devenue 3.8 Go au lieu de 5Go.

```
def convert_large_xml_to_json(file_json,file_tmx):
    id=1
val=0
start=time.time()
with open(file_json,'w+',encoding="utf8") as file:
for event, elem in ET.iterparse(file_tmx, events=("start", "end")):
if elem.tag=="seg" and event=="start":
if val==0:
                    m={}
#print("arab :",elem.attrib)
m["ar"]=elem.text
                    val=1
else:
                    i={}
                    i["_id"]=id
                    index={}
                    index["index"]=i
#print("de :",elem.attrib)
m["fr"]=elem.text
                    val=0
id+=1
json.dump(index, file, ensure_ascii=False)
                    file.write('\n')
                    json.dump(m, file, ensure_ascii=False)
                    file.write('\n')
            elem.clear()

print("time is :",time.time()-start," ms")
```

Figure 17 : Convertir le Corpus Tmx en Json par Fonction d'Itération

[27]https://stph.scenari-community.org/

```
def mmap_io(path):
    collection=[]
    id_=0
with open(path,encoding='utf8') as f:
with mmap.mmap(f.fileno(),length=0,access=mmap.ACCESS_READ)as mmap_read:
for line in mmap_read.read():
if id_%2==0:
                    id_+=1
else:
                    file_json = json.loads(str(line))
                    collection.append(file_json)
                    id_+=1
yield collection
```

Figure 18 : Convertir le Corpus Tmx en Json par Fonction Mmap

- Format Csv est le sigle de « Coma Separated Values », qui peut se traduire par « valeurs séparées par des virgules ». Il s'agit d'un format de fichier qui se caractérise par la représentation de données tabulaires sous forme de texte, les valeurs de chaque cellule étant séparées par des virgules ou des points-virgules. La fonction principale des fichiers Csv est de permettre la portabilité des données tabulaires d'un programme à l'autre.

Cette fois, le fichier Corpus.json est converti au format Csv par une fonction créée, comme illustré par la figure 19.

```
def large_json_to_csv(path):
    i=0
df=pd.read_json(path,lines=True,encoding='utf8',chunksize=10000)
for line in df:

        x=[line['ar'].tolist()[x] for x in range(1,len(line),2)]
        y=[line['en'].tolist()[y] for y in range(1,len(line),2)]
        dict={'ar':x,'en':y}
        dataset=pd.DataFrame(dict)
#print(dataset.head())
dataset.to_csv('ar-en.csv',index=True,mode='a')
print("Done :", i)
        i+=1
```

Figure 19 : Fonction : Convertir le format Json en Csv

Les figures 20 et 21 montrent à quoi ressemblent les corpus après avoir été convertis de tmx (figure 22) aux formats Json et Csv :

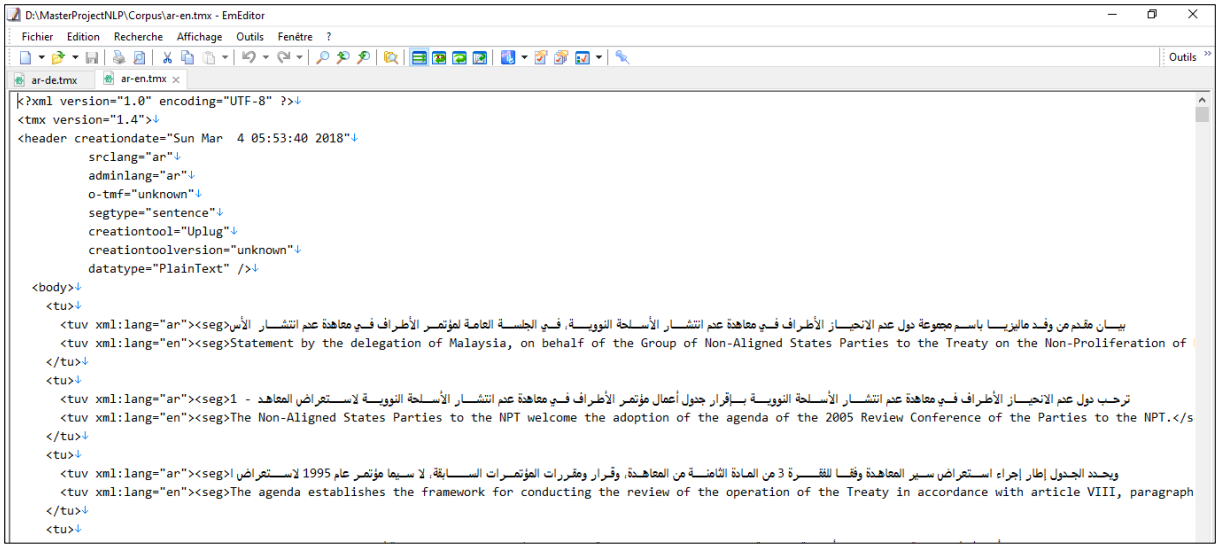


Figure 20 : Corpus sous format Tmx (après téléchargement)

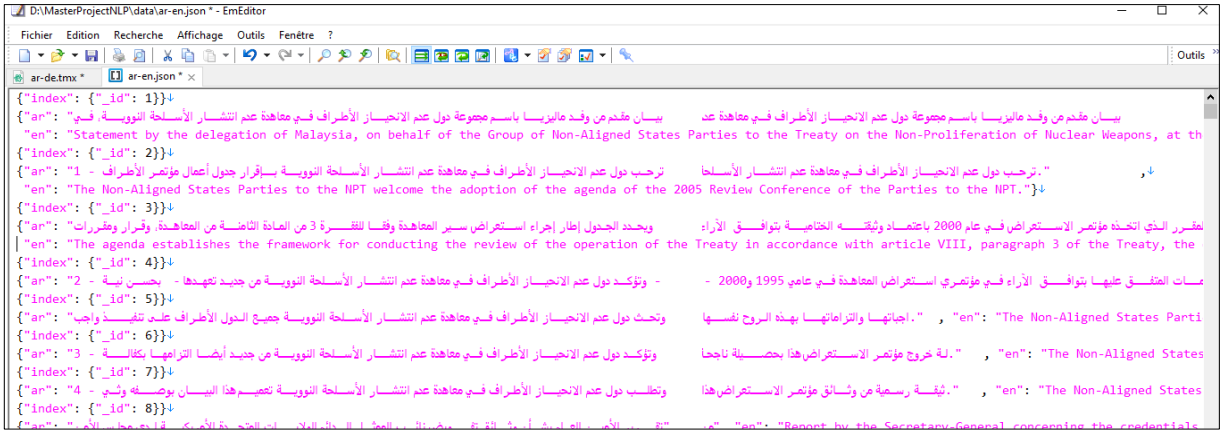


Figure 21 : Corpus sous format Json

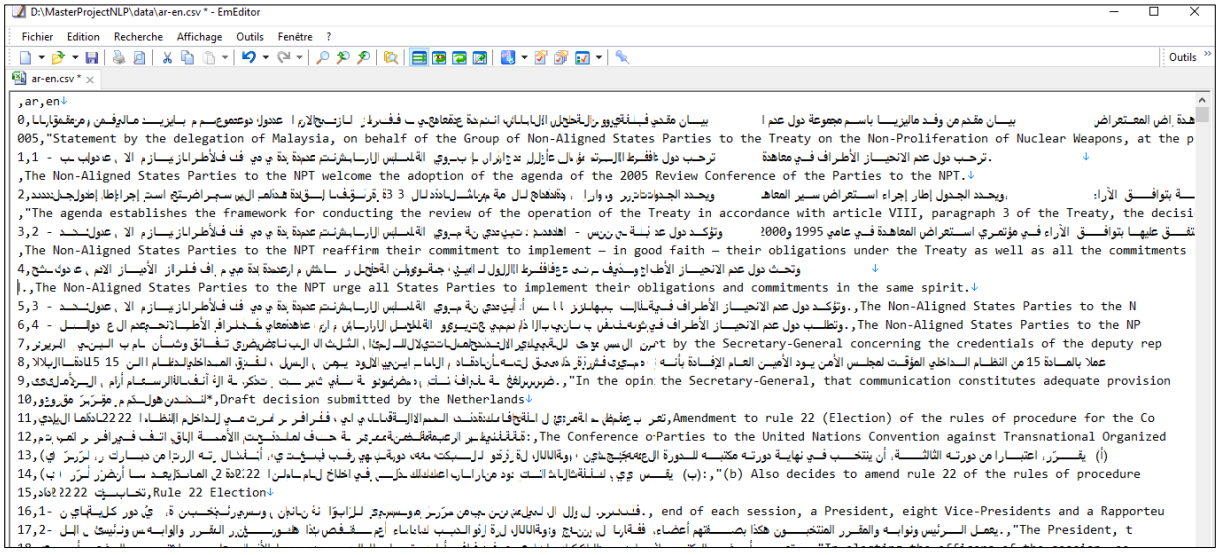


Figure 22 : Corpus sous format Csv

2.2. Traitement du Corpus :

Au début, nous avons créé notre propre moteur de recherche afin d'extraire la traduction contextuelle de chaque mot ou expression choisie dans l'ensemble téléchargé, en utilisant des modèles d'indexation (booléien, vectoriel et LSI), mais nous avons retrouvé que ce moteur ne supporte pas les gros fichiers, comme l'ensemble des corpus qui ont été téléchargés (5 Go pour chaque paire de langues). Cependant, pour résoudre ce problème qui nous a pris beaucoup de temps en programmation, nous nous sommes tournés vers l'utilisation de deux approches suivantes (figure 23) :

- Première approche : utilisation du moteur de recherche ElasticSearch, qui est basé sur Json et qui a permis d'importer notre corpus et de l'utiliser malgré sa taille.

- Deuxième approche : utilisation de Tf-Idf, qui est basé sur SPARK, en utilisant le format csv, et qui a permis de calculer Tf-Idf d'une manière plus simple et efficace.

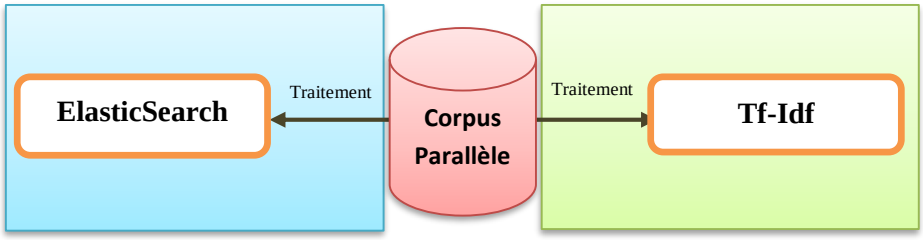


Figure 23 : Deux approches utilisées lors du traitement du Corpus

2.2.1. Première approche : Utilisation de l'ElasticSearch

Depuis sa sortie en 2010, ElasticSearch est devenu extrêmement populaire. Il a été développé la même année par Shay Banon qui le commercialise via la société Elastic qu'il a créé. Malgré son aspect commercial et le fait qu'il ne fasse pas partie de la fondation Apache, ElasticSearch est open source et l'un des logiciels du projet de ELK (outil d'analyse de logs composé de 3 logiciels : **E**lasticSearch, **L**ogstash et **K**ibana). Formellement, ElasticSearch est un moteur distribué de stockage, de recherche et d'analyse de contenu.

- En tant que moteur de stockage, il stocke les données en format JSON, annulant ainsi le besoin de joindre à son application de recherche un support de stockage.
- En tant que moteur de recherche, Il utilise *Apache Lucene* pour les fonctionnalités d'indexation et de recherche de contenu sur ces documents JSON.

- En tant que moteur d'analyse de contenu, il s'appuie sur **Logstash**, un logiciel de gestion de logs et **Kibana**, une plateforme d'exploration et de visualisation des données, pour effectuer des analyses sur les données qu'il stocke.

Sa particularité lui vient du fait qu'il s'interroge à partir d'une API REST, accessible à partir du protocole HTTP et utilise le format JSON aussi bien pour le stockage des données que pour le renvoi des réponses de requêtes. Les standards HTML, REST et JSON le rendent facile à s'intégrer avec d'autres applications, simple à utiliser pour les utilisateurs. De plus, le fait qu'il utilise le JSON pour le stockage des données rend l'exploitation des données possible à partir de n'importe quel langage de programmation possédant des API permettant de lire du JSON (comme Python dans notre cas). Les aspects distribués d'ElasticSearch lui permettent d'effectuer les recherches de contenu très rapidement. Toute interaction avec ElasticSearch se fait à travers son API REST qui permet d'envoyer des requêtes HTTP.

Traditionnellement, les moteurs de recherche de contenu sont déployés sur des moteurs de base de données pour fournir aux applications développées sur ces bases de données des fonctionnalités d'indexation et de recherche. Historiquement, ceci a été le cas parce que les moteurs de recherche de contenu n'offraient pas de support de stockage persistent sur lequel directement effectuer les recherches. ElasticSearch est l'un des moteurs de recherche de contenu moderne qui fournit à la fois le support de stockage NoSQL, l'indexation et la recherche de contenu, et l'Analytics. ElasticSearch est un moteur NoSQL orienté-document, au même titre que MongoDB ou RavenDB et il fournit toutes les fonctionnalités de stockage distribué que ce type de moteur offre. Cependant, il est possible aussi d'utiliser ElasticSearch avec d'autres moteurs de base de données. Si on veut développer une nouvelle application ou ajouter des fonctionnalités de recherche de contenu dans une application existante, l'API REST est l'une des fonctionnalités qui rend ElasticSearch attractif. En effet, ElasticSearch expose ses fonctionnalités à travers cette *API REST*, cela permet d'écrire les requêtes en JSON, de recevoir les résultats des requêtes en JSON et d'effectuer l'ensemble des interactions avec ElasticSearch via ce même API[28].

L'installation de ElasticSearch nécessite la présence, au minimum, de Java 6 sur la machine destinée à héberger le moteur de recherche. A ce stade, le répertoire d'installation comprend les répertoires suivants :

[28] https://www.data-transitionnumerique.com/

- **bin** : contient le script de lancement *elasticsearch* et le script d'installation de plugins `plugin` (respectivement *elasticsearch.bat* et *plugin.bat* pour Windows).
- **config** : contient les fichiers de configuration d'ElasticSearch (*elasticsearch.yml*) et celui des logs (*logging.yml*).
- **lib** : contient certaines librairies utilisées par ElasticSearch.

Pour démarrer ElasticSearch sur le Système Exploitation Windows, il faut exécuter le script *elasticsearch.bat.*

Au lancement, ElasticSearch va créer de nouveaux répertoires :

- **data** : destiné à contenir les données indexées.
- **logs** : contient les fichiers de journalisation.
- **work** : contient des fichiers temporaires nécessaires au fonctionnement du moteur de recherche.

Le second répertoire va nous être utile pour vérifier le bon lancement d'ElasticSearch. Pour cela, il suffit de visualiser le contenu du principal fichier `elasticsearch.log`[29].

Une seconde façon de vérifier qu'ElasticSearch s'est correctement lancé consiste à ouvrir l'URL *http://localhost:9200/* dans un navigateur Web (figure 24). En effet, ElasticSearch écoute par défaut sur le port 9200 pour répondre aux éventuelles requêtes HTTP REST qui lui sont faites. La réponse du serveur confirme le bon fonctionnement d'ElasticSearch et rappelle le code d'authentification et le numéro de la version installée :

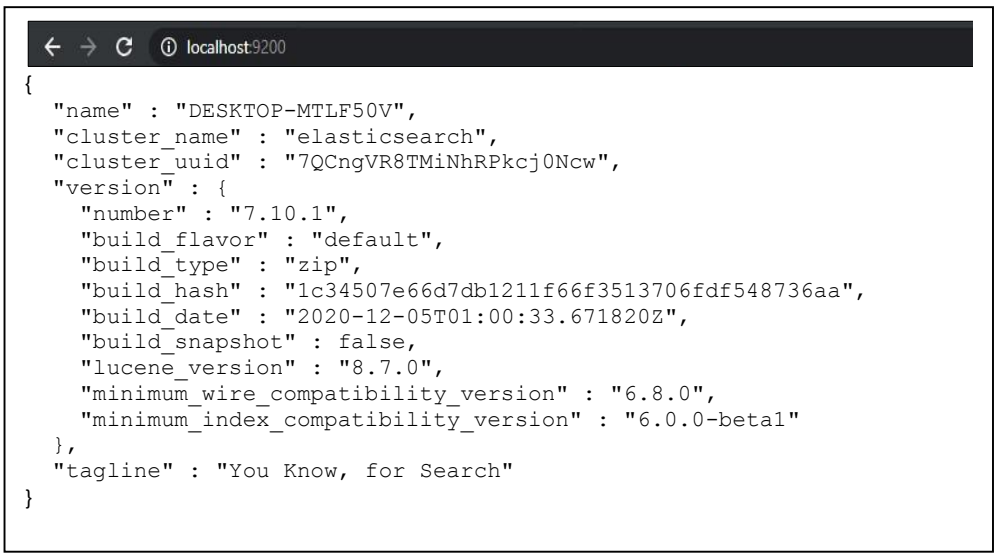


Figure 24 : Démarrage ElasticSearch

[29] https://blog.zenika.com/

Pour importer les corpus convertis en format json sur elasticsearch, il existe plusieurs modes :

- **Pour le Corpus petite taille :** on utilise l'API d'indexation d'Elasticsearch via la commande CURL et ce, après la création d'un index 'mviewer' :

```
$ curl -s -H"Content-Type: application/x-ndjson" -XPOST'http://localhost:9200/_bulk' --data-binary "@Corpus-name.json"
```

- **Pour le Corpus grande taille :** on utilise Logstash du plugin jdbc, c'est l'un des méthodes d'alimentation les plus connues et sur laquelle nous nous sommes appuyés pour importer nos corpus (figure 25), au contraire de la première méthode qui supporte les documents qui ont des petites tailles.

Figure 25 : Importation du corpus par Logstash

- **Logstash** est un outil pour la saisie, le traitement et la sortie des données logs. Sa fonction est d'analyser, filtrer et découper les logs pour les transformer en documents formatés à destination d'ElasticSearch[30]. Il fonctionne au moyen de pipelines : des fichiers de configuration contenant les informations de connexions ainsi que les modifications à effectuer sur les logs à envoyer.

La création d'un fichier de configuration Logstash du Corpus.json est composée de 3 parties :

- **Input** : pour spécifier l'emplacement des Corpus à envoyer, généralement dans le répertoir. C://user/admin/desktop/Data/ar-en.json,
- **Filter** : pour modifier le formatage des logs à envoyer,
- **Output** : pour spécifier les informations de connexion à votre instance ELK.

La figure 26 montre un exemple de contenu du fichier nom-corpus.config.

[30] https://www.oracle.com/

```
input{
  file {
    path => "/Users/admin/Desktop/data/ar-en.json"
    start_position => "beginning"
    sincedb_path => "NUL"
}
}
filter{
   json {
     source => "message"
}
}
output{
  elasticsearch {
hosts => "http://localhost:9200"
    index => "ar-en"
}
  stdout{}
}
```

Figure 26 : Configuration de Corpus.json en Corpus.config (Logstash)

Une fois le fichier de configuration finalisé, on lance la commande ci-dessous, avec l'option -f pour pouvoir troubleshooter et l'importer.

./bin/logstash -f corpus-name.config

Après une durée de 7 heures, le Corpus a été importé et prêt à être utilisé par le moteur de recherche ElasticSearch.

- Client Python ElasticSearch est un Client officiel de bas niveau pour Elasticsearch. Son objectif est de fournir un terrain d'entente pour tout le code lié à ElasticSearch en Python ; pour cette raison, il essaie d'être sans opinion et très extensible. Alors, on installe ElasticSearch package avec **pip** :

*python -m **pip** installer Elasticsearch*

Après l'installation, création de la connexion entre ElasticSearch et Python (figure 27).

```
fromelasticsearchimportElasticsearch
es =Elasticsearch
(
http_auth=(cluster_name, cluster_uuid)
)
es.cluster.health(wait_for_status="yellow")
```

Figure 27 : Création de connexion entre ElasticSearch et Python

2.2.2. Deuxième approche : Utilisation de Tf-Idf (Spark)

Dans cette méthode, nous nous sommes appuyés sur Spark pour traiter plus facilement des données volumineuses (figure 28). Puisque le contenu des corpus utilisés n'est pas modifiable, nous pouvons calculer Tf-IDF une seule fois et avoir un tableau (matrice) contenant toutes les données nécessaires.

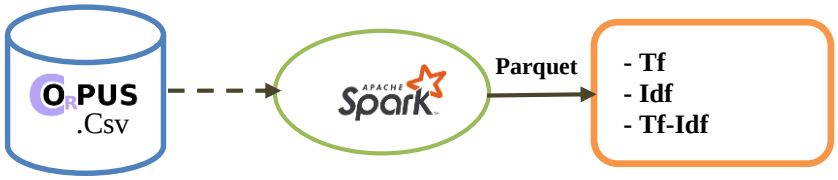


Figure 28 : Traitement de Corpus .Csv par Spark en utilisant Tf-Idf

- Spark est un moteur d'analyse unifié pour le traitement de données à grande échelle. Spark a commencé en tant que projet de l'Université de Californie à Berkeley en 2009 et a rejoint l'Apache Software Foundation en 2013 [32].

Spark a été conçu pour résoudre certains problèmes liés à l'architecture Hadoop lorsqu'elle est utilisée pour l'analyse, tels que le streaming de données, SQL sur des fichiers stockés sur HDFS et l'apprentissage automatique. Il peut distribuer des données sur tous les nœuds de calcul d'un cluster de manière à réduire la latence de chaque étape de calcul. Une autre différence de Spark est sa flexibilité : il existe des interfaces pour Java, Scala, SQL, R et Python, et des bibliothèques pour différents problèmes, telles que MLlib pour l'apprentissage automatique, GraphX pour le calcul de graphes et Spark Streaming, pour les charges de travail en streaming (figure 29).

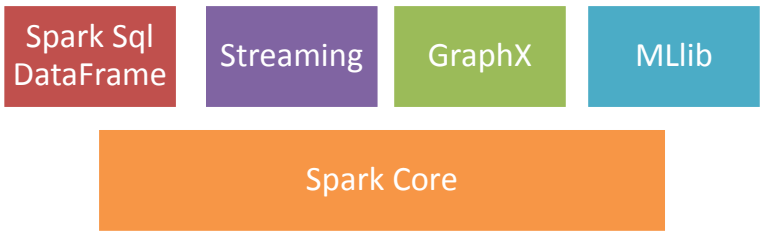


Figure 29 : Composantes du Framework Spark[31]

Spark utilise l'abstraction de travail, avec un processus de pilote qui reçoit l'entrée de l'utilisateur pour démarrer des exécutions parallèles, et des processus de travail qui résident sur les nœuds du cluster, exécutant des tâches. Il dispose d'un outil de gestion de cluster intégré et

[31] https://www.stat4decision.com/

prend en charge d'autres outils, tels que Hadoop YARN et Apache Mesos (et même Kubernetes), s'intégrant dans différents environnements et scénarios de distribution de ressources.

Spark peut également être très rapide, car il essaie d'abord de distribuer les données sur tous les nœuds et de les conserver en mémoire au lieu de s'appuyer uniquement sur les données sur le disque. Il peut gérer des ensembles de données plus volumineux que la mémoire totale disponible, en déplaçant les données entre la mémoire et le disque, mais rendant le processus plus lent que si l'ensemble de données entier s'intégrait dans la mémoire totale disponible de tous les nœuds (figure 30) [32] :

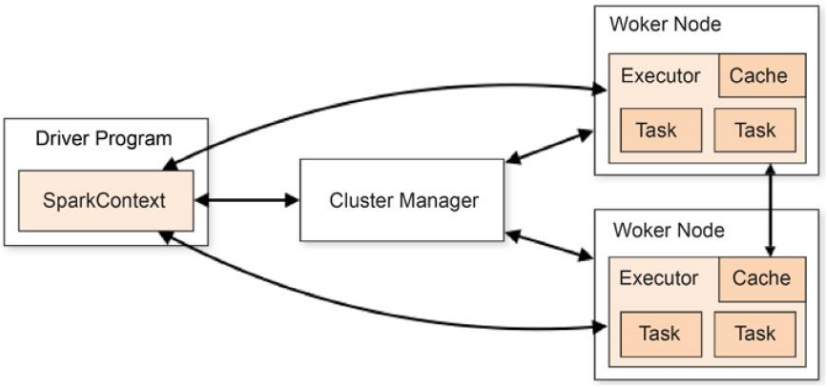


Figure 30 : Mécanisme de travail du Spark

Spark peut être utilisé soit en tant que shell interactif avec Scala, Python et R, soit en tant que plate-forme de soumission de travaux avec la commande spark-submit. La méthode submit est utilisée pour envoyer des tâches à un cluster Spark codé dans un script. L'interface shell Spark pour Python s'appelle PySpark. Il est accessible directement depuis le terminal, où la version Python par défaut sera utilisée ; il est accessible à l'aide du shell IPython ou même à l'intérieur d'un éditeur Jupyter, Pycharm etc.

- Spark SQL et Pandas DataFrames

Le RDD « Resilient Distributed Dataset » (tables de données résilientes), est l'abstraction de base que Spark utilise pour fonctionner avec les données. À partir de Spark version 2.0, l'API recommandée pour manipuler les données est l'API DataFrame. L'API DataFrame est construite au-dessus de l'API RDD, bien que l'API RDD puisse toujours être accessible. Travailler avec des RDD est considéré comme de bas niveau et toutes les opérations sont disponibles dans l'API DataFrame.

Le module SQL permet aux utilisateurs d'interroger les données dans Spark à l'aide de requêtes SQL, similaires aux bases de données relationnelles courantes. L'API DataFrame fait partie du module SQL, qui fonctionne avec des données structurées. Cette interface pour les données permet de créer des optimisations supplémentaires, avec le même moteur d'exécution utilisé, indépendamment de l'API ou du langage utilisé pour exprimer ces calculs.

L'interface pour Spark DataFrames est similaire à l'interface pandas, mais il existe des différences importantes :

• La première différence est que les Spark DataFrames sont immuables : après avoir été créés, ils ne peuvent pas être modifiés.

• La deuxième différence est que Spark a deux types d'opérations différentes : les transformations et les actions.

Les transformations sont des opérations qui sont appliquées sur les éléments d'un DataFrame et sont mises en file d'attente pour être exécutées ultérieurement, sans extraire encore les données.

Ce n'est que lorsqu'une action est appelée que les données sont extraites et que toutes les transformations en file d'attente sont exécutées. C'est ce qu'on appelle l'évaluation paresseuse.

- Ecriture de fichiers Parquet :

Le format de données Parquet est un stockage en colonnes binaire qui peut être utilisé par différents outils, notamment Hadoop et Spark. Il a été conçu pour prendre en charge la compression, afin de permettre des performances et une utilisation du stockage plus élevées. Sa conception orientée colonnes facilite la sélection des données pour les performances, car seules les données dans les colonnes requises sont récupérées. Cela évite de rechercher les données et de supprimer les valeurs dans les lignes qui ne sont pas nécessaires, réduisant ainsi le temps de récupération pour les scénarios de Big Data, où le les données sont distribuées sur disque. Les fichiers Parquet peuvent également être lus et écrits par des applications externes, avec une bibliothèque C++, et même directement depuis les pandas [32].

La bibliothèque Parquet est en cours de développement avec le projet Arrow. Un projet qui permet les traitements multi-systèmes en éliminant les échanges indirects.
Lorsqu'on envisage des requêtes plus complexes dans Spark, le stockage des données au format Parquet peut augmenter les performances, en particulier lorsque les requêtes doivent rechercher

un ensemble de données volumineux. La compression aide à réduire le volume de données qui doit être communiqué lorsqu'une opération est effectuée dans Spark, diminuant ainsi les E/S du réseau.

Le format de données Parquet prend également en charge les schémas et les schémas imbriqués, similaires à JSON, et Spark peut lire le schéma directement à partir du fichier. Le Parquet writer dans Spark a plusieurs options, telles que le mode (ajouter, écraser, ignorer ou erreur, l'option par défaut) et la compression, avec un paramètre pour choisir l'algorithme de de cette compression.

Les algorithmes disponibles sont : gzip, lzo, brottli, lz4, Snappy et Uncompressed. L'algorithme par défaut est snappy.

Rappelant toujours qu'il faut bien comprendre que Spark n'est pas une base de données et ne stocke rien, il se base sur des systèmes de stockage Hadoop ou autre comme, AWS S3, Cassandra, MongoDB, etc.

Le passage aux processus que nous effectuons avec SPARK, en utilisant l'interface des lignes de commandes pour le prétraitement du Corpus et le calcul de Tf-Idf se déroule comme suit :

2.2.2.1. Lecture du Corpus

```
C:\Users\admin>pyspark
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.0.3
      /_/

Using Python version 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020 15:52:53)
SparkSession available as 'spark'.
>>> 21/08/28 16:53:11 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of ProcessTree metrics is stopped

>>> from pyspark.ml.feature import HashingTF, IDF, Tokenizer ,CountVectorizer,IDFModel,VectorAssembler,Normalizer,StopWordsRemover
>>> df = spark.read.parquet("C:/Users/admin/Desktop/test_p")
>>> df=df.select("ar","en")
>>> df.show(5)
+--------------------+--------------------+
|                  ar|                  en|
+--------------------+--------------------+
|) 515 (1832) Ibid...|Ibid., pp. 519, 5...|
|□□□□□□ □□ □□□□□ □...|Everyone should e...|
|□□□ □□□ □□□ □□ □□...|A number of targe...|
|□□□□ □□□□ □□□□ □□...|A Chinese buyer e...|
|43 - □□□□□□□ □□□□...|Draft article 7 d...|
+--------------------+--------------------+
only showing top 5 rows
```

Figure 31 : Lecture du corpus

2.2.2.2. Tokenization

```
>>> token=Tokenizer(inputCol='en',outputCol='token')
>>> df=token.transform(df.na.drop(subset=['en']))
>>> df.show(5)
+--------------------+--------------------+--------------------+
|                  ar|                  en|               token|
+--------------------+--------------------+--------------------+
|) 515 (1832) Ibid...|Ibid., pp. 519, 5...|[ibid.,, pp., 519...|
|□□□□□□ □□ □□□□□ □...|Everyone should e...|[everyone, should...|
|□□□ □□□ □□□ □□ □□...|A number of targe...|[a, number, of, t...|
|□□□□ □□□□ □□□□ □□...|A Chinese buyer e...|[a, chinese, buye...|
|43 - □□□□□□□ □□□□...|Draft article 7 d...|[draft, article, ...|
+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Figure 32 : Segmentation du corpus

2.2.2.3. Elimination des mots vides

```
>>> stop=StopWordsRemover(inputCol='token',outputCol='words')
>>> df=stop.transform(df)
>>> df.show(5)
+--------------------+--------------------+--------------------+--------------------+
|                  ar|                  en|               token|               words|
+--------------------+--------------------+--------------------+--------------------+
|) 515 (1832) Ibid...|Ibid., pp. 519, 5...|[ibid.,, pp., 519...|[ibid.,, pp., 519...|
|□□□□□□ □□ □□□□□ □...|Everyone should e...|[everyone, should...|[everyone, enjoy,...|
|□□□ □□□ □□□ □□ □□...|A number of targe...|[a, number, of, t...|[number, targets,...|
|□□□□ □□□□ □□□□ □□...|A Chinese buyer e...|[a, chinese, buye...|[chinese, buyer, ...|
|43 - □□□□□□□ □□□□...|Draft article 7 d...|[draft, article, ...|[draft, article, ...|
+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Figure 33 : Elimination des mots vides du corpus

2.2.2.4. Calcul de Tf

```
>>> tf = HashingTF(inputCol="words", outputCol="tf")
>>> df = tf.transform(df)
>>> df.show(5)
+--------------------+--------------------+--------------------+--------------------+--------------------+
|                  ar|                  en|               token|               words|                  tf|
+--------------------+--------------------+--------------------+--------------------+--------------------+
|) 515 (1832) Ibid...|Ibid., pp. 519, 5...|[ibid.,, pp., 519...|[ibid.,, pp., 519...|(262144,[48608,10...|
|□□□□□□ □□ □□□□□ □...|Everyone should e...|[everyone, should...|[everyone, enjoy,...|(262144,[2701,416...|
|□□□ □□□ □□□ □□ □□...|A number of targe...|[a, number, of, t...|[number, targets,...|(262144,[2306,126...|
|□□□□ □□□□ □□□□ □□...|A Chinese buyer e...|[a, chinese, buye...|[chinese, buyer, ...|(262144,[47197,51...|
|43 - □□□□□□□ □□□□...|Draft article 7 d...|[draft, article, ...|[draft, article, ...|(262144,[5674,785...|
+--------------------+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Figure 34 : Calcul Tf

HashingTF est un Transformer qui prend des ensembles de termes et les convertit en vecteurs de caractéristiques de longueur fixe. Dans le traitement de texte, un « ensemble de termes » peut être un sac de mots. HashingTF utilise l'astuce de hachage. Une caractéristique brute est mappée dans un index (terme) en appliquant une fonction de hachage.

2.2.2.5. Calcul de Idf

```
>>> idf=IDF(inputCol='tf',outputCol='idf',minDocFreq=2).fit(df)
>>> df=idf.transform(df)
>>> df.show(5)
21/08/28 17:13:04 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|                  ar|                  en|               token|               words|                  tf|                 idf|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|) 515 (1832) Ibid...|Ibid., pp. 519, 5...|[ibid.,, pp., 519...|[ibid.,, pp., 519...|(262144,[48608,10...|(262144,[48608,10...|
|□□□□□□ □□ □□□□□ □...|Everyone should e...|[everyone, should...|[everyone, enjoy,...|(262144,[2701,416...|(262144,[2701,416...|
|□□□ □□□ □□□ □□ □□...|A number of targe...|[a, number, of, t...|[number, targets,...|(262144,[2306,126...|(262144,[2306,126...|
|□□□□ □□□□ □□□□ □□...|A Chinese buyer e...|[a, chinese, buye...|[chinese, buyer, ...|(262144,[47197,51...|(262144,[47197,51...|
|43 - □□□□□□□ □□□□...|Draft article 7 d...|[draft, article, ...|[draft, article, ...|(262144,[5674,785...|(262144,[5674,785...|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Figure 35 : Calcul Idf

2.2.2.6. Sauvegarde Tf et Idf

>>> tf.save("C:/Users/admin/Desktop/spark_tf")

>>> idf.save ("C:/Users/admin/Desktop/spark_idf ")

2.2.2.7. Sauvegarde Tf-Idf en parquet

```
>>> df.write.mode("overwrite").parquet("C:/Users/admin/Desktop/tf_idf")
21/08/28 17:48:59 WARN DAGScheduler: Broadcasting large task binary with size 4.2 MiB
```

2.3. Prétraitement (de la requête)

On a créé une fonction qui contient la segmentation, élimination des mots vides, lemmatisation et racinisation comme montre les figures 36 et 37.

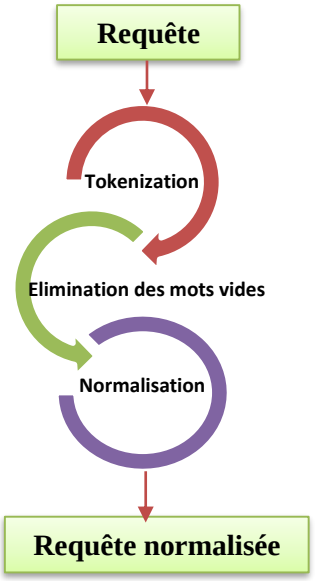


Figure 36 : Schéma de prétraitement de la requête

```
print("+-------------------------------------+")
print('           Prétraitement             ')
print("+-------------------------------------+")
    df_requete = spark.createDataFrame([("0",text)],["id","text"])
    token=Tokenizer(inputCol='text',outputCol='token')
df_requete=token.transform(df_requete.na.drop(subset=['text']))
    stop=StopWordsRemover(inputCol='token',outputCol='words')
df_requete=stop.transform(df_requete)
df_requete=tf.transform(df_requete)
#dfn=dfn.select('text','tf')
df_requete.cache()
#idf=IDF(inputCol='tf',outputCol='idf').fit(dfn)
df_requete=idf.transform(df_requete)
print(df_requete.take(1))
df_requete.show()
```

Figure 37 : Programme Prétraitement de la requête

Pour la langue arabe et la langue anglaise on a créé des fonctionnalités supplémentaires :

- Pour la langue arabe : on a ajouté trois fonctions ; la dérivation des verbes et des mots basés sur le modèle n-gramme, la conjugaison et la voyellation.
- Pour la langue anglaise : le traitement se fait par la sémantique, pour cela on a utilisé l'ontologie WordNet.

3. Conclusion

Dans ce chapitre, nous avons mis l'accent sur la réalisation de la plate-forme traduction automatique par contexte, qui représente l'essentiel de notre travail. L'identification a été effectuée selon plusieurs étapes, en l'occurrence, la collection et préparation des données, traitement de corpus et prétraitement de la requête. La validation de notre solution fera l'objet du prochain chapitre.

Chapitre V : Validation du Système proposé

Chapitre V

Mise en œuvre du Système proposé

1. Introduction

Après avoir développé les étapes de réalisation de notre système depuis la préparation et la collection de corpus jusqu'aux prétraitements et recherches, en identifiant tous les modules et les fonctions utilisés ou créés, maintenant on va présenter et expliquer le fonctionnement et le déroulement de l'application Web.

2. Besoins fonctionnels et non fonctionnels

2.1. Besoins fonctionnels

Le système à réaliser doit répondre au besoin principal de notre projet, à savoir, la traduction de contexte de et vers langue arabe, en focalisant sur le Corpus Parallèle Anglais-Arabe. Pour y arriver, nous devons assurer que le système doit permettre à un utilisateur de saisir la requête (mot ou une expression) en langue choisie et voir la traduction de contexte de cette requête.

2.2. Besoins non fonctionnels

Le système peut :

- Être écrit en Langage Python, et s'exécuter sous un système d'exploitation Windows ;
- Avoir une simple IHM (Interface Homme/Machine), ergonomique et pratique à utiliser ;
- Être optimisé en espace mémoire et surtout en temps d'exécution, où les résultats doivent être retournés dans un temps raisonnable) ;
- Être extensible, afin d'ajouter de futures fonctionnalités.

2.3. Diagramme de cas d'utilisation (DCU)

Le système doit offrir les exigences fonctionnelles citées précédemment. Voici un diagramme de cas d'utilisation (figure 37) illustre bien ces fonctionnalités et comment sera-t-il utilisé par l'utilisateur :

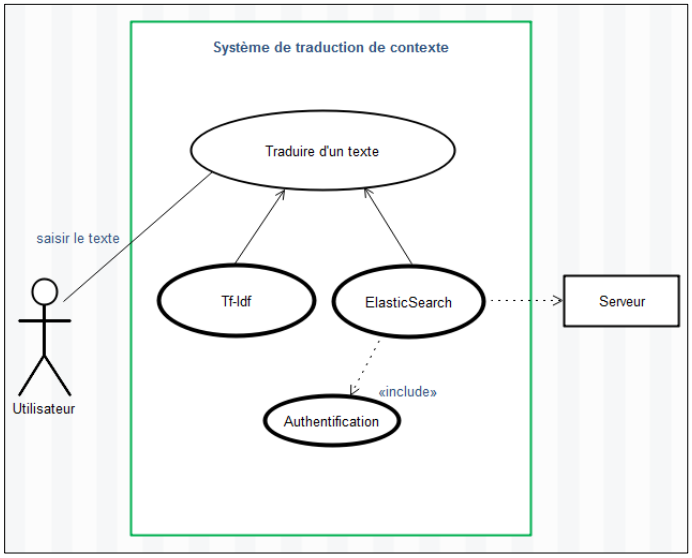


Figure 38 : Diagramme de cas d'utilisation de traduction de contexte

2.4. Diagramme de cas de séquence

Le diagramme ci-dessous (figure 38) permet de décrire comment les éléments du système interagissent entre eux et avec l'utilisateur :

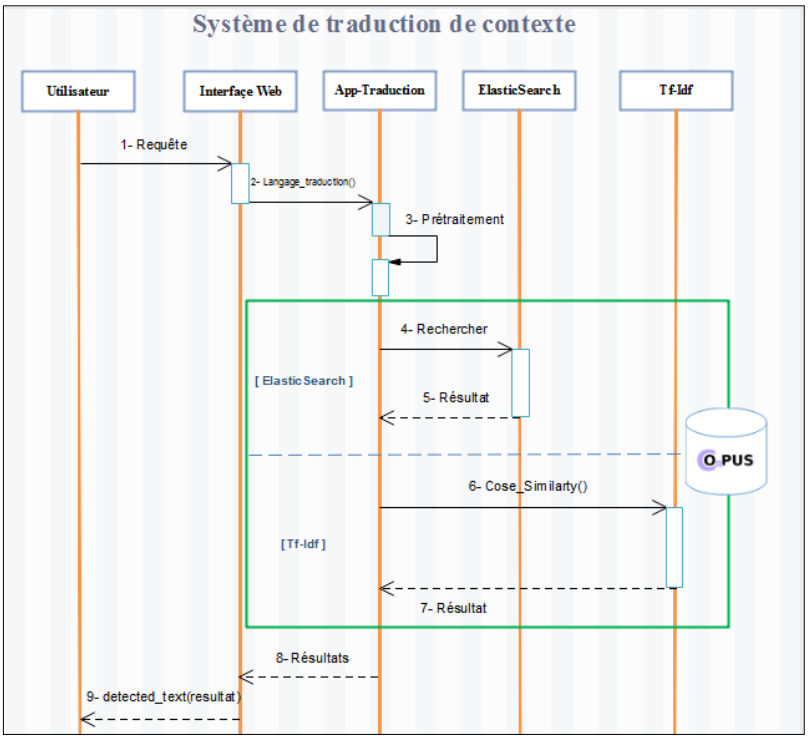


Figure 39 : Diagramme de cas de séquence de traduction de contexte

3. Environnement de développement

3.1. Langages de programmation utilisés

Nous avons implémenté notre solution sous forme d'une application avec une interface Web permettant à l'utilisateur de manipuler le système. Dans cette section, nous introduisons notre environnement de travail, qui est le langage Python sur un environnement de développement Pycharm et le HTML. Ensuite, nous définissons les différentes bibliothèques utilisées et les caractéristiques techniques de l'ordinateur destinées à la mise en place du système.

3.1.1. Python

Python[32] est un langage de programmation open source créé par le programmeur Guido van Rossum en 1991. Il apparut à l'époque comme une façon d'automatiser les éléments les plus ennuyeux de l'écriture de scripts ou de réaliser rapidement des prototypes d'applications.

Depuis quelques années, ce langage de programmation s'est hissé parmi les plus utilisés dans le domaine du développement de logiciels, de gestion d'infrastructure et d'analyse de données. Il s'agit d'un élément moteur de l'explosion du Big Data.

On distingue deux versions de Python : Python 2 et Python 3. Les différences entre ces deux versions sont multiples. Python 2.x est l'ancienne version, qui continuera d'être supportée et donc de recevoir des mises à jour officielles jusqu'en 2020. Après cette date, elle continuera d'ailleurs sans doute de subsister de façon non officielle.

Python 3.x est la version actuelle du langage. Elle apporte de nombreuses fonctionnalités nouvelles et très utiles, telles qu'un meilleur contrôle de concurrence et un interpréteur plus efficace. Cependant, l'adoption de Python 3 a été longtemps ralentie par le manque de bibliothèques tierces prises en charge. Un grand nombre d'entre elles n'étaient compatibles qu'avec Python 2, ce qui rendait la transition compliquée. Cependant, ce problème est aujourd'hui pratiquement résolu et il reste peu de raisons valables de continuer à utiliser Python 2.

Le langage Python doit sa popularité à plusieurs avantages qui profitent aussi bien aux débutants qu'aux experts, dont :

[32] https://www.lebigdata.fr/

- Il est facile à apprendre et à utiliser. Ses caractéristiques sont peu nombreuses, ce qui permet de créer des programmes rapidement et avec peu d'efforts. De plus, sa syntaxe est conçue pour être lisible et directe ;
- Il fonctionne sur tous les principaux systèmes d'exploitation et plateformes informatiques. De plus, même s'il ne s'agit clairement pas du langage le plus rapide, il compense sa lenteur par sa versatilité ;
- Même s'il est principalement utilisé pour le scripting et l'automatisation, ce langage est aussi utilisé pour créer des logiciels de qualité professionnelle. Qu'il s'agisse d'applications ou de services Web, le Python est utilisé par un grand nombre de développeurs pour créer des logiciels.

3.1.2. PyCharm

PyCharm[33] est un environnement de développement intégré (EDI), utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django.

C'est un logiciel multi-plateforme, développé par l'entreprise tchèque JetBrains, qui fonctionne sous Windows, Mac OS X et GNU/Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache.

3.1.3. HTML

Le HyperText Markup Language, généralement abrégé HTML ou dans sa dernière version HTML5, est le langage de balisage conçu pour représenter les pages web. HTML est inspiré du Standard Generalized Markup Language (SGML). Il s'agit d'un format ouvert et permet de :

- ✓ Structurer sémantiquement la page ;
- ✓ Mettre en forme le contenu de la page ;
- ✓ Créer des formulaires de saisie ;
- ✓ Inclure des ressources multimédias dont des images, des vidéos et autres.

[33] https://fr.wikipedia.org

3.2. Bibliothèques utilisées

Pour parfaire ce système, plusieurs bibliothèques ont été intégrées comme indique le tableau 8 suivant :

Bibliothèque	Rôle
Numpy	Permet d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres.
Tashaphyne	Stemmer et segmenteur léger arabe : تاشفين: التجذيع الخفيف للنصوص العربية. Il supporte principalement le light stemming (suppression des préfixes et suffixes) et donne toutes les segmentations possibles. Il utilise un automate à états finis modifié qui permet de générer toutes les segmentations.
Libqutrub.conjugator	Logiciel de conjugaison des verbes arabes قطرب: تصريف الأفعال العربية
Pyarabic	Une bibliothèque de langue arabe spécifique pour Python, fournit des fonctions de base pour manipuler les lettres et le texte arabes, comme détecter les lettres arabes, les groupes et les caractéristiques de lettres arabes, supprimer les signes diacritiques, etc. مكتبة برمجية للغة العربية بلغة بيثون، توفر دوالا للتحكم في الحروف والنصوص، مثلا تحديد نوع الحرف، حذف الحركات، مقارنة التشكيل.
Qalsadi	Analyseur morphologique arabe pour Python
nltk	Une plate-forme de premier plan pour la création de programmes Python fonctionnant avec des données en langage humain. Il fournit des interfaces faciles à utiliser vers plus de 50 corpus et ressources lexicales telles que WordNet, ainsi qu'une suite de bibliothèques de traitement de texte pour la classification,tokenization, stemming, tagging, parsing, et le raisonnement sémantique, des wrappers pour les bibliothèques Traitement Naturel du Langage (NLP) de puissance industrielle, et un forum de discussion actif.
Spacy	L'une des principales bibliothèques du langage Python pour NLP, Cette technologie permet notamment d'analyser automatiquement des textes en langage courant, afin d'en comprendre le sens, d'identifier rapidement les informations principales, ou encore de trouver des similitudes entre plusieurs textes. Cette technologie permet notamment d'analyser automatiquement des textes en langage courant, afin d'en comprendre le sens, d'identifier rapidement les informations principales, ou encore de trouver des similitudes entre plusieurs textes.
Wordnet	Base de données lexicale pour la langue anglaise, qui a été créée par Princeton, et fait partie du corpus NLTK (Natural Language Toolkit).Vous pouvez utiliser WordNet avec le module NLTK pour trouver la signification des mots, des synonymes, des antonymes, etc.
Pandas	Est un package Python qui fournit des structures de données rapides, flexibles et expressives conçues pour rendre le travail avec des données « relationnelles » ou « étiquetées » à la fois facile et intuitif. Il vise à être le bloc de construction fondamental de haut niveau pour effectuer une analyse de données pratique et réelle en Python. De plus, il a pour objectif plus large de devenir l'outil d'analyse / manipulation de données open source le plus puissant et le plus flexible disponible dans n'importe quelle langue.
Spark	bigdata freamwork
Flask	CGI freamwork
Jinja2	Est un moteur de templates pour le langage de programmation Python

Tableau 8 : Bibliothèques utilisées dans le Projet

3.3. Caractéristiques techniques

Nous avons réalisé notre application dans un environnement possédant par les caractéristiques suivantes :

- Système d'exploitation : Windows professionnel 10.0.1836;
- Processeur I5-7200U, CPU @ 2.50 Ghz 2.70 Ghz ;
- RAM : 8 G0.

4. L'interface graphique du système

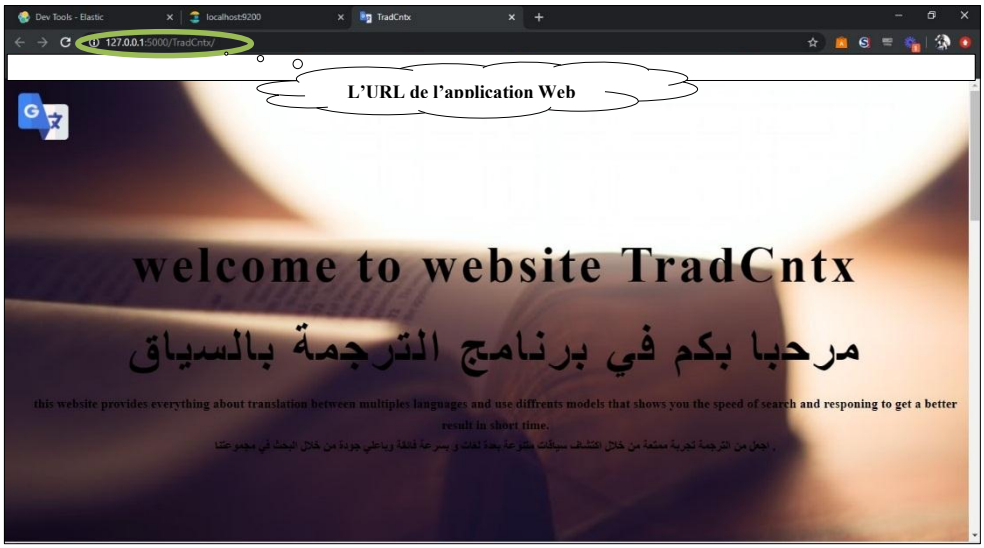


Figure 40 : Page d'accueil de l'application Web

Figure 41 : Interface Graphique de l'Application Web

5. Guide d'utilisation du Système

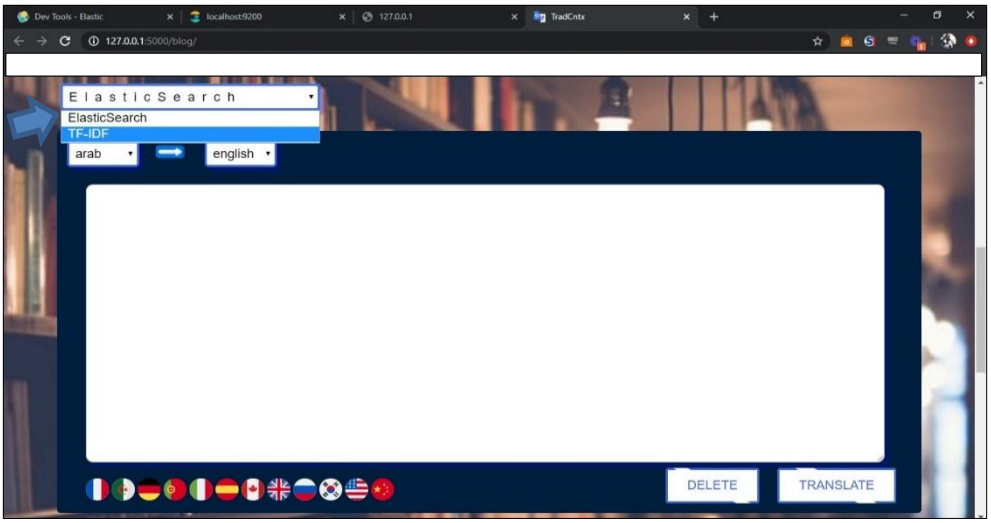


Figure 42 : Choix de l'approche ElasticSearch ou TF-Idf

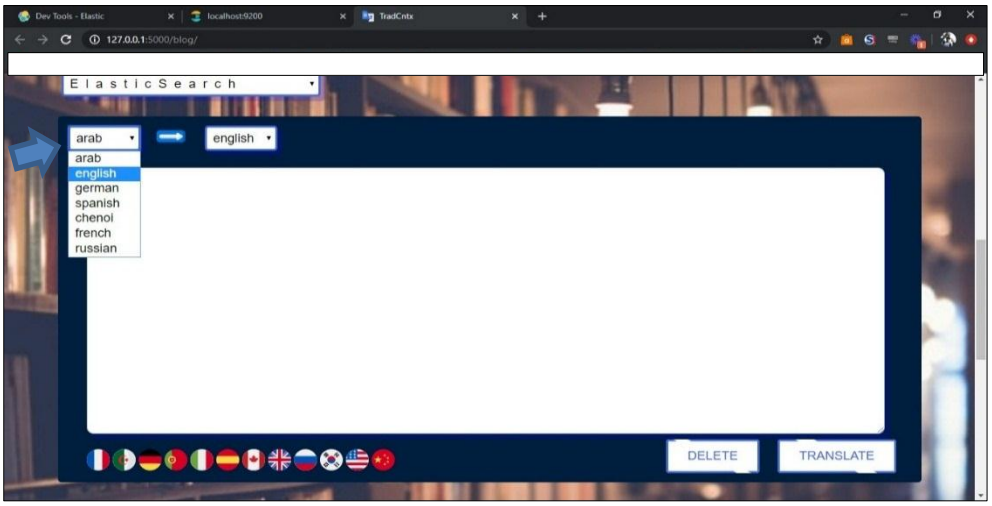


Figure 43 : Choix de la langue source

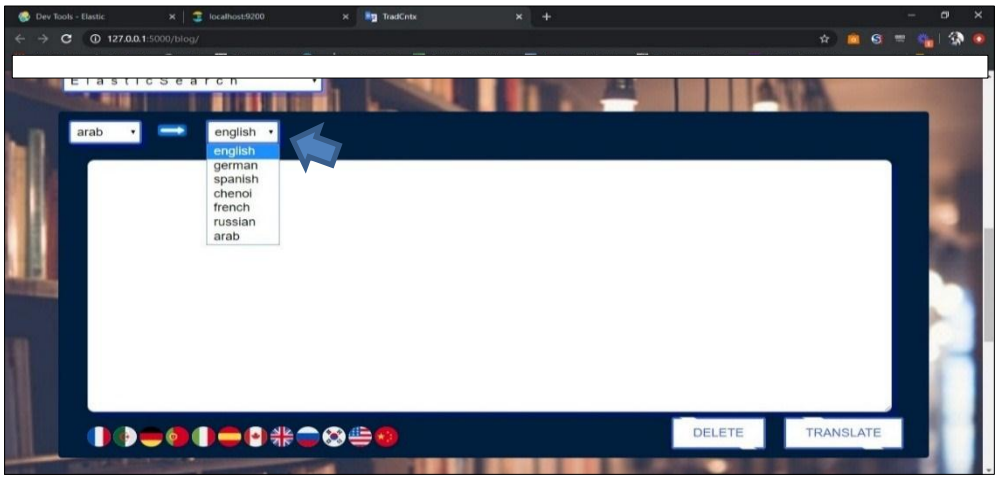


Figure 44 : Choix de la langue cible

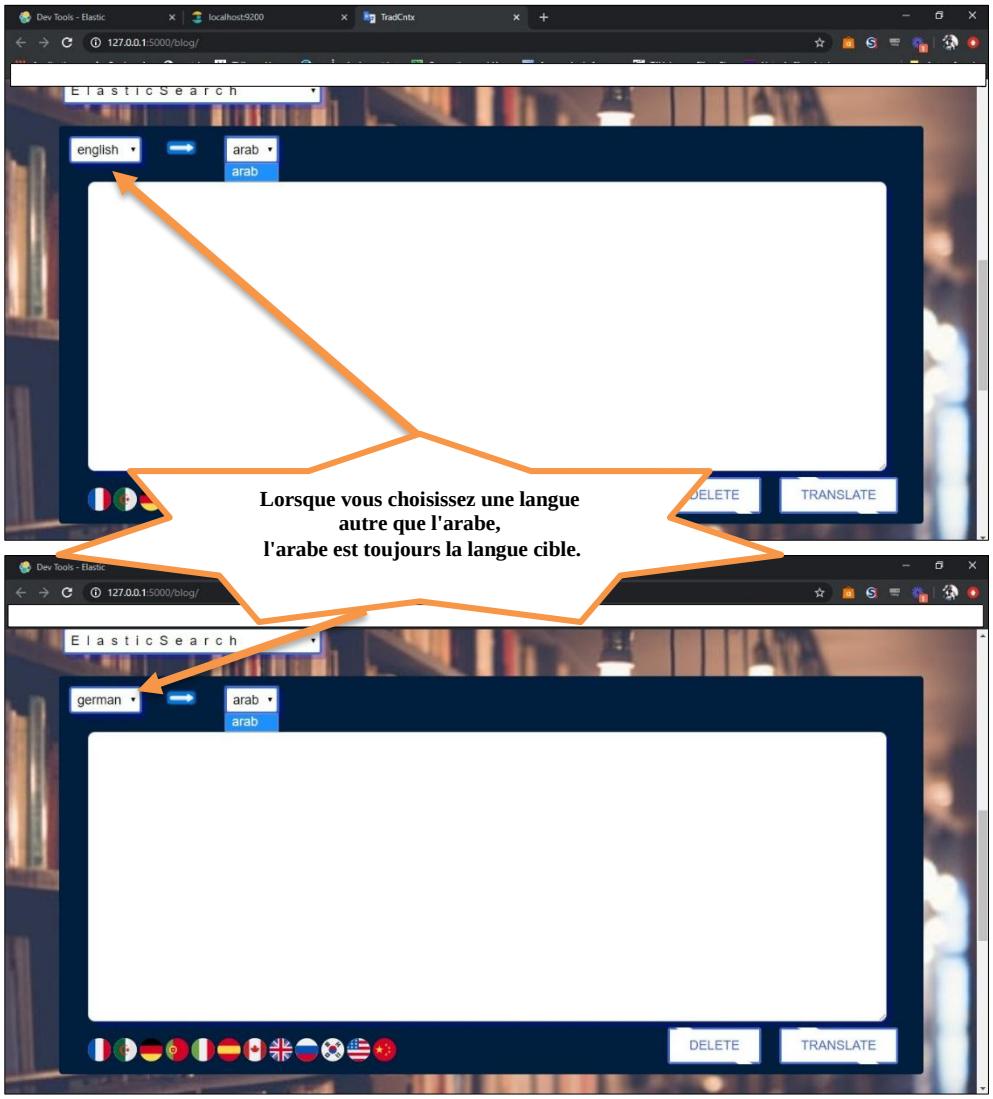


Figure 45 : Traduction de et vers la langue arabe

6. Exemples de traduction :

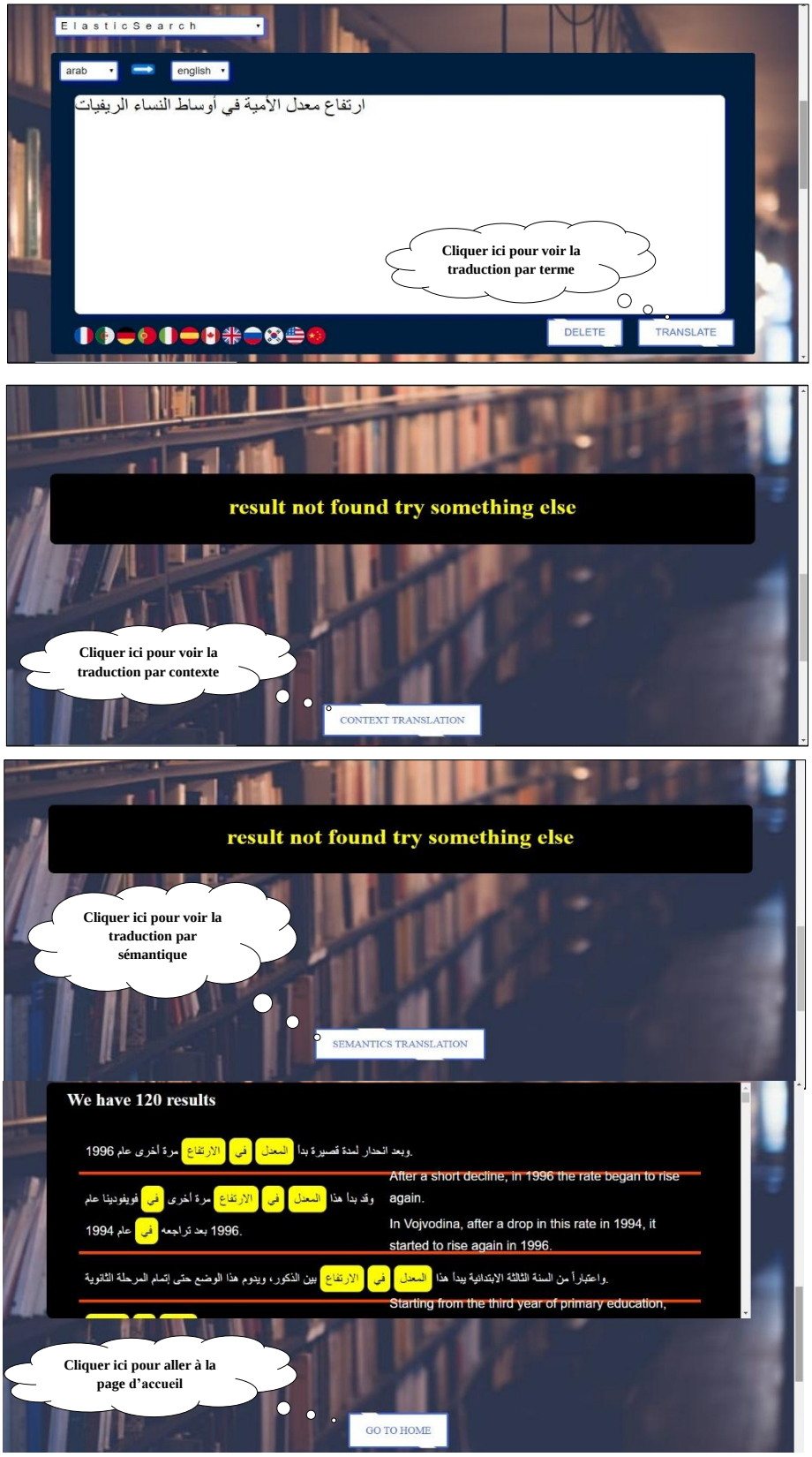


Figure 46 : Traduction d'un texte par ElasticSearch

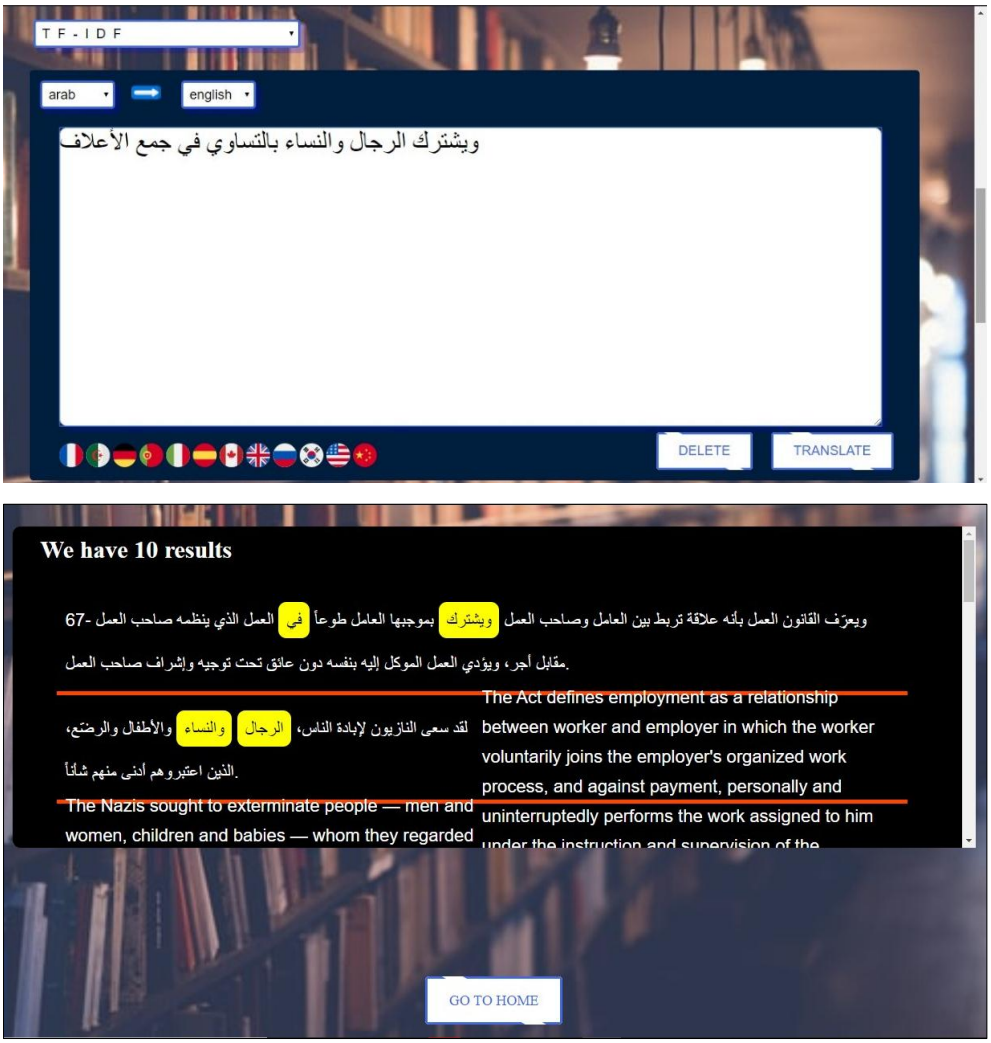


Figure 47 : Traduction d'un texte par Tf-Idf du Spark

7. Conclusion

Dans ce dernier chapitre, nous avons donné un aperçu sur l'utilisation et le fonctionnement de notre application, en donnant différents exemples de traduction de mots ou de textes de l'arabe vers l'anglais, avec la possibilité d'utiliser la traduction également dans toutes les autres langues restantes.

Comme le montrent les exemples donnés, l'utilisation d'ElasticSearch et de Tf-Idf conduit au même résultat, sauf que nous avons remarqué que l'utilisation de la première approche est très efficace en termes de temps d'exécution, contrairement à la première qui prend beaucoup de temps.

Conclusion générale

Conclusion générale

Ces dernières années, la traduction automatique s'est développée de manière surprenante en raison du grand développement technologique qui a eu lieu dans le domaine de l'informatique ainsi que dans la facilité d'accès à Internet, faisant du monde un petit village. Cela a conduit à la convergence des cultures des pays et des différentes langues, ce qui s'est reflété positivement sur la traduction automatique pour permettre la communication entre les individus de manière simple.

L'avènement de la traduction automatique statistique « TAS » a été l'une des méthodes de traduction automatique les plus populaires et les plus efficaces pendant des années jusqu'à la révolution du Deep Learning. Ce type de traduction dépend d'une grande quantité de données connues sous le nom du corpus parallèle. Les corpus parallèles signifient la collection d'un grand nombre de textes dans la langue source et la langue cible de sorte que chaque ligne du premier fichier soit la traduction correcte de la même ligne dans le deuxième fichier. Pour construire un traducteur automatique efficace, nous avons besoin de millions de lignes parallèles. La collecte de ces textes de haute qualité est donc l'un des plus grands défis auxquels sont confrontés les développeurs dans ce domaine, en particulier lorsqu'il s'agit de langues dotées de fonctionnalités spéciales telles que l'arabe.

Nous avons dans ce projet réalisé une traduction de contexte de et vers l'arabe, en créant un moteur de recherche capable de trouver des mots et des textes à partir d'énormes données, à caractère politique, téléchargées depuis le site Web d'OPUS. Ces données sont représentées par un ensemble de paires : l'arabe et les six autres langues : anglais, français, allemand, chinois, espagnol et russe. Dans notre étude, nous nous sommes concentrés sur le corpus arabe-anglais, en tenant compte des cinq langues restantes.

Au début, nous avons créé un moteur de recherche qui fouille dans ce corpus, mais après plusieurs tentatives de programmation, nous avons obtenu des résultats médiocres, avec un temps de recherche très long. Cela est dû à l'énorme volume de données d'une part, et d'autres parts, aux performances de l'ordinateur qui ne correspondaient pas aux niveaux requis. C'est pourquoi nous avons décidé d'utiliser deux autres approches à savoir : Elasticsearch et Tf-Idf du Spark.

Après avoir développé toutes les fonctions de prétraitement et de recherche et toutes les procédures facilitant la traduction de contexte, notamment pour les langues arabe et anglaise, nous avons pu obtenir de bons résultats en un temps record, notamment avec ElasticSearch.

Parmi les difficultés que nous avons rencontrées, nous citons :

- ✓ La complexité de la conversion des fichiers tmx aux formats json et csv, malgré la structure étroite entre eux ;
- ✓ Le temps pris pour importer les Corpus téléchargés dans ElasticSearch ; Environ sept (7) heures pour un seul Corpus ;
- ✓ L'obtention de résultats de la recherche, dans certains cas, complètement erronés, dû : d'une part, à la nature du Corpus parallèle qui se rapporte à un domaine particulier, et d'autres parts, à certains textes parallèles non identiques entre eux ;
- ✓ Le temps d'exécution très long, ainsi que les mauvais résultats obtenus lors de l'utilisation d'une autre méthode Tf-Idf que la méthode Tf-Idf du Spark ;
- ✓ La plupart des problèmes liés à la langue arabe dans ce système :
  - L'arabe n'est pas voyellée dans les Corpus que nous avons téléchargés et dans la plupart des autres corpus parallèles qui existent sur le Web.
  - Difficulté de segmentation des mots et des phrases.
  - L'absence de significations sémantiques en raison du manque de création de leur propre ontologie, comme dans la langue anglaise.
  - Complexité structurelle de certaines phrases.
- ✓ Les caractéristiques techniques de l'ordinateur sont limitées et insuffisantes pour traiter le gros volume de Corpus.

Malgré ces complexités et obstacles rencontrés, notamment dans le traitement de la langue arabe, nous avons pu surmonter certains d'entre eux, et d'autres restent comme des défis ou des paris. Nous espérons y trouver des solutions dans leurs traitements, comme les solutions atteintes par les chercheurs en langue anglaise, et pourquoi pas !

Nous proposons dans les perspectives du projet les actions suivantes :

- Créer des Corpus parallèles pour la langue arabe avec une traduction précise sans erreurs, et dans plusieurs domaines. Dans ces Corpus, il vaut mieux que la langue arabe soit voyellée, afin qu'elle soit plus facile à traiter,

- Créer ou contribuer à de grands projets pour faire avancer la traduction en langue arabe, comme le projet ArabicWordNet, Glove etc., afin que l'on puisse avoir une connaissance sémantique des mots de la langue arabe, et une statistique qui attribue une valeur potentielle à chaque chaîne de mots au moyen d'une distribution de probabilité.

Enfin, il faut noter que le travail mené dans ce mémoire a joué un très grand rôle pour nous permettre d'explorer le monde de la traduction d'une manière générale et la traduction automatique statistique en particulier, à travers l'expérience et la mise en œuvre de techniques approuvées ou éprouvées qui contribueront à améliorer la qualité de la traduction de contexte. Nous espérons voir ce domaine s'ouvrir à grande échelle et se développer par les moyens de projets de fin d'études afin d'atteindre les objectifs souhaités, car c'est un fait d'une grande importance.

Bibliographie

Références Bibliographiques

[1] Thierry Poibeau, « Machine Translation », Livre, Edition MIT Press, 2017.

[2] Zoulikha BENBLAL et Fatima BELOUAFI, « Intégration d'un lemmatiseur arabe dans le cadre d'un système de recherche d'information », Université d'Adrar, Algérie, 2014.

[3] J. LEON, « Le traitement automatique des langues », Université Paris 7, France 2001.

[4] F. YVON, « Une petite introduction au traitement Automatique du langage naturel », support de cours, Ecole Nationale Supérieur des télécommunications, 2007.

[5] P. BOUILLON, « Traitement automatique des langues naturelles », édition Duculot, France 1998.

[6] Athmane CHELLOUF et Yacine AYACHI, « Proposition et conception d'un système de détection d'évènements sur les réseaux sociaux dédié à la langue Arabe », Ecole Militaire Polytechnique, Algérie, 2018.

[7] Jean VERONIS, « Informatique et Linguistique » unité d'enseignement INF Z18, université de Provence, centre informatique pour les lettres et sciences humaines, France, 2001.

[8] J-H. JAYEZ, « Compréhension automatique du langage naturel le cas du groupe nominal en français », Masson, 1985.

[9] B.Rafik, S.Abdelhak, « Proposition d'un système de traduction automatique (anglais-arabe», Université de Blida, Algérie, 2019.

[10] Haïtem AFLI, «Approche mixte pour la traduction automatique statistique », Grenoble, France, 2010

[11] Abdelkarim MARS, « Le TAL au service des enseignants des langues : mise en œuvre d'une plate-forme pour l'enseignement du français et de l'arabe, langues étrangères », Université Grenoble Alpes, France, 2016.

[12] Siham BOULAKNADEL, « Traitement Automatique des Langues et Recherche d'Information en langue arabe dans un domaine de spécialité : Apport des connaissances morphologiques et syntaxiques pour l'indexation », Université de Nantes, France, 2008.

[13] Nebia BENZATER, « Analyse Morphologique du Texte Arabe pour Son Indexation Sémantique », Université des sciences et de la technologie, Oran, Algérie, 2015.

[14] Authoul Abdul Hay, « Constitution d'une ressource sémantique arabe à partir de corpus multilingues alignés », Université de Grenoble, France 2012.

[15] BALLAOUI Hammad, « le traitement automatique de la langue arabe (tala) pour la recherche d'information sur le web », Université Chouaïb Doukkali D'El Jadida, Maroc, 2017.

[16] Houda SAADANE, « Le traitement automatique de l'arabe dialectalisé : aspects méthodologiques et algorithmiques », Université Grenoble Alpes, France, 2015.

[17] Tahar DILEKH, « Un modèle sémantique pour la recherche d'information en langue arabe », Université Batna 2, Algérie, 2019.

[18] Jian-Yun Nie, « Le domaine de recherche d'information – Un survol d'une longue histoire », Département d'informatique et recherche opérationnelle, Université de Montréal.

[19] Abdelkarim HERZALLAH, « support de cours sur recherche d'information », Université de Sfax, Tunisie, 2014.

[20] Haifa ZARGAYOUNA, « Indexation sémantique de documents XML », Université Paris XI Orsay, 2005.

[21] Kamel GARROUCH, « Modèles de Recherche d'information basés sur les Réseaux Bayésiens et les Réseaux Possibilistes », Université de Sfax, 2014.

[22] Djalila BOUGHERB, « Recherche d'information multicritères », Université de Annaba, Algérie, 2014.

[23] BENAOUDA-ZOUAOUI Khalil et AYADI Ramzi, « Agrégation des résultats dans la recherche d'information XML », Université de Blida, Algérie.

[24] BOUHADIBA Mohamed El Amine, « Utilisation des ressources textuelles semi-structurées dans la recherche intelligente sur le web », Université de Oran, Algérie, 2014.

[25] F.BOUBEKEUR, M.BOUGHANEM, L.TAMINE, M. DAOUD, « De l'utilisation de WordNet pour l'indexation conceptuelle des documents», Université de Tizi-Ouzou (Algérie) et Université Paul Sabatier (France), 2010.

[26] TOUIHR Faiza Safia, « Indexation sémantique d'une base textuelle », Université de Mostaganem, Algérie, 2016.

[27] W.ABBASSI, B.MEFTAH, « Indexation sémantique de documents textuels», Université de Ouargla, Algérie, 2013.

[28] Lynda DOUFENE, « Un Modèle De Reformulation Des Requêtes Pour La Recherche D'information Sur Le Web », Université de Tizi-Ouzou, Algérie, 2012.

[29] A.DYHIA, « Indexation sémantique latente de documents textuels», Université de Tizi-Ouzou, Algérie, 2012.

[30] Yaël Champclaux, « Un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amélioration du processus de recherche d'information », Université Paul Sabatier (Toulouse), France, 2009.

[31] Gerard. Salton, « Automatic Information Organization and Retrieval». McGraw Hill, 1968.

[32] ByIvan Marin, Ankit Shukla et Sarang VK, « Big Data Analysis with Python », Livre, Edition Packt, 2019.

[33] François-Régis Chaumartin, « WordNet et son écosystème : un ensemble de ressources linguistiques de large couverture », Colloque BD lexicales, Montréal, Canada, 2007.