RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

FACULTÉ DES SCIENCES ET DE LA TECHNOLOGIE
DÉPARTEMENT DES MATHMATIQUES ET D'INFORMATIQUE

M É M O I R E

pour obtenir le titre de

Master en Informatique

Spécialité : Systèmes Intelligents pour l'Extraction de Connaissances (SIEC)

Présenté par

Abdelghafour BENKHELIFA **et** Ouail Abdeldjalil DJEBRIT

Contribution au développement d'un outil de prétraitement du texte Arabe

---

soutenu publiquement le 28 juillet 2018

Jury :

Président :	S. OULAD NAOUI	MC.	Univ. Ghardaia
Examinateur :	A. BOUHANI	MA.	Univ. Ghardaia
Examinateur :	A. KERRACHE	MA.	Univ. Ghardaia
Encadreur :	S. BELLAOUAR	MC.	Univ. Ghardaia

Dédicace

Je dédie ce modeste travail à :

Ma très chère Mère,que dieu la protège.

Mon très cher Père,que dieu le protège.

Pour leur patience, leur amour, leur soutien et leurs encouragements.

Mes très chers frères et soeurs.

A toute ma famille.

A tout mes amies et mes camarades.

A tous les enseignants et les enseignantes qui ont contribué a ma formation tout au long de ma vie A tout ceux qui mont aidé dans lélaboration de ce travail.

Abdelghafour.

Dédicace

Je dédie ce modeste travail à :

Mes parents et mes grands-parents vos prières vos bénédiction m'ont été d'un grand secoure pour mener a bien mes études , Aucune dédicace ne saurait être assez éloquente pour exprimer ce que vous méritez pour tous les sacrifices que vous n'avez cessés de me donner

Mes chères surs, mon frère , mes tantes et mes oncles.

En témoignage de l'attachement ,de l'amour et de l'affection que je porte pour vous, je vous dédie ce travail avec tous mes vux de bonheur, de santé et de réussite A tous les autres membres de ma famille sans exception.

A mes amis et mes collègues dans et en dehors l'université.

Wail.

Remerciements

En préambule à ce mémoire, la grande louange à ﷲ qui nous aide et nous donne la bonne santé, la patience et le courage durant l'élaboration de ce modeste travail.

Nous tenons à remercier tous ceux qui nous ont aidés, d'une manière ou d'une autre, pendant ce travail d'étude et de recherche.

Nous tenons d'abord à remercier très chaleureusement Monsieur **SLIMANE BELLAOUAR** qui nous a permis de bénéficier de son encadrement. Les conseils qu'il nous a prodigué, la patience, la confiance qu'il nous a témoignés ont été déterminants dans la réalisation de notre travail de recherche.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre modeste travail Et de l'enrichir par leurs propositions.

Nos remerciements s'étendent également à tous nos enseignants durant les années des études.

Merci à Tous et à Toutes.

Résumé

La forte augmentation de texte disponible en format numérique a fait ressortir la nécessité de concevoir et de développer des outils de traitement automatique du texte dans le but d'extraire l'information pertinente. Les textes arabes ne font pas exception quant à leur disponibilité. Cependant le développement d'outils de traitement pour la langue arabe n'a pas suivi la même allure comme pour les autres langues.
De plus, la phase de prétraitement du texte est considérée comme l'entrée principale de la plupart des applications et tâches de traitement automatique de langue naturelle (TALN) qui est l'une des branches importantes de l'intelligence artificielle.
Comme contribution au développement d'outil pour le TALN arabe, notre objectif consiste à développer un flux de prétraitement du texte arabe. Pour ce faire nous développons des nœuds dans la plate-forme KNIME (Konstanz Information Miner) qui s'occupe de filtrage des mots vides, de ponctuations des caractères non arabe et essentiellement de la racinisation.
En fin nous montrons la validité de notre flux de prétraitement du texte arabe en l'intégrant dans une tâche de classification de texte. Les résultats révèlent que le prétraitement du texte arabe contribue à l'amélioration des tâches de l'apprentissage automatique.

Mots clé : Intelligence artificielle, Analyse de sentiments, Traitement automatique des langues naturelles, Prétraitement du texte arabe, Racinisation.

Abstract

The considerable increase of text available in digital format has highlighted the need to design and develop automated text processing tools for the purpose of extracting necessary information's. Arabic texts are not the exception as to their availability. However, the development of processing tools for the Arabic language did not follow the same pace as for other languages.
In addition, the preprocessing phase of the text is considered to be the main entry for most Natural language processing (NLP) applications and tasks, which is one of the major branches of artificial intelligence.
As a contribution to tool development for the Arabic NLP, our goal is to develop a pre-processing flow of Arabic text. To do this we develop nodes in the KNIME platform which deals with the filtering of empty words, punctuations of non-Arabic characters and essentially stemming.
Finally, we show the validity of our preprocessing flow of Arabic text by integrating it into a text classification task. The results reveal that the pre-processing of the Arabic text contributes to the improvement of the tasks of machine learning.

Keywords : Artificial intelligence, Sentiment analysis, Natural Language Processing, Arabic text Preprocessing, Stemming.

ملخص :

أبرزت الزيادة الكبيرة في النصوص الإلكترونية الحاجة إلى تطوير أدوات لمعالجة النصوص بشكل آلي من أجل استخلاص المعلومات منها. النصوص العربية ليست استثناء من حيث وفرتها، ولكن تطوير أدوات معالجة النصوص العربية لم يكن بنفس الوتيرة مع بقية اللغات.
تعتبر المعالجة المسبقة للنصوص كمدخل أساسي لأغلب تطبيقات معالجة اللغات الطبيعية الذي يعتبر فرعا هاما من فروع الذكاء الاصطناعي.
مساهمتنا تتمثل في تطوير أداة تندرج ضمن فرع المعالجة الآلية للغات الطبيعية، هدفنا هو تطوير سلسلة مهام للمعالجة المسبقة للنصوص العربية على منصة نايم، من أجل ذلك قمنا بتطوير بعض العقد على منصة نايم تسعى لتصفية النصوص العربية من الكلمات الفارغة والترقيم والحروف غير العربية والأهم من ذلك هو استخراج جذر للكلمة العربية.
في النهاية أثبتنا صلاحية عملنا المتمثل في سلسلة مهام للمعالجة المسبقة للنص العربي عن طريق إدماجه في عملية تصنيف نصوص، وكشفت النتائج أن المعالجة المسبقة للنص العربي تساهم في تحسين مهام التعلم الآلي.

الكلمات المفتاحية: الذكاء الاصطناعي ، تحليل المشاعر ، معالجة اللغات الطبيعية ، المعالجة المسبقة للنص العربي ، التجذير.

Table des matières

Liste des tableaux

Table des figures

Introduction générale

L'homme est une créature sociale qui aime naturellement communiquer avec les autres races humaines, apprendre à connaître leurs actualités, acquis de leurs cultures. En particulier, il souhaite prendre diverses sciences dans différents domaines de différentes langues. Ces désirs sont compensés par de nombreux obstacles tels que le temps énorme nécessaire pour rechercher des informations spécifiques dans un domaine particulier, ressources humaines nécessaires pour classer le contenu d'une grande bibliothèque, ou bien les ressources financières pour couvrir les frais d'un traducteur linguistique, sachant que toutes ces données textuelles sont très grandes et qui ne cessent d'augmenter jour après jour. Parallèlement à la révolution de l'intelligence artificielle, une branche spécialisée dans le traitement de texte est apparue dans de différentes langues. C'est la branche connue sous le nom de Traitement Automatique des Langues Naturelles (TALN).

TALN est l'un des domaines d'investigation les plus importants et les plus évolutifs en intelligence artificielle. TALN s'occupe de la création des programmes capables de traiter et de comprendre les langues humaines. Un langage naturel est un phénomène très compliqué, et son étude implique de nombreux niveaux d'analyse liés à la morphologie et aux règles syntaxiques et à la sémantique du langage [Allen, 1995 ; Manning and Schütze, 1999]. Le prétraitement de texte est une tâche de base du TALN. Il vise à créer une forme intermédiaire à partir du texte entré basé sur l'extraction des mots, l'analyse morphologique, et l'annotation de texte [Awajan, 2007].

Dans ce mémoire, nous nous focalisons sur la quatrième langue parlée dans le monde, plus de 435 million personnes parlent l'arabe dans le monde, et plus de 219 millions utilisateurs d'internet en arabe avec une croissance de 8616 % entre 2000 et 2018 selon IWS [1].

La langue arabe est à la fois stimulante et intéressante. Elle est intéressante en raison de son histoire, de l'importance stratégique de son peuple et de la région qu'elle occupe [Bakalla, 2002]. En plus de sa brillance et son utilisation c'est aussi un défi en raison de

1. https://www.internetworldstats.com/stats7.htm

sa structure linguistique complexe [Attia and Somers, 2008].
L'objectif de ce travail est de contribuer au développement d'un outil de prétraitement du texte arabe, ce prétraitement n'est pas une seule étape. Mais c'est un flux de tâches à suivre commençant par le filtrage des chiffres, symboles, caractères étrangers (non Arabe), ponctuation et se terminant par l'extraction des racines des mots. Le processus d'extraction des racines des mots est très délicat, compte tenu de la complexité morphologique de la langue Arabe [Otair, 2013].
Nous choisissons la plateforme KNIME (Konstanz Information Miner) qui ne supporte pas l'arabe pour préparer une chaine de prétraitement du texte arabe qui peut être utilisée plus tard dans d'autres tâches d'intelligence artificielle, telles que la classification de texte, la traduction ou l'analyse du discours, etc.

Pour ce faire, nous utilisons des nœuds prêts et nous développons les nœuds dont nous aurons besoin (un nœud pour le filtrage des mots vide et deux nœuds pour la racinisation du mots). KNIME est un environnement modulaire qui permet un assemblage visuel facile et une exécution interactive d'un pipeline de données [Berthold et al., 2009].

Le reste de ce mémoire est organisé comme suit :
Le chapitre (1) de l'intelligence artificielle et le TALN Arabe, introduit le domaine de l'intelligence artificielle ainsi que le sous-domaine de TALN. Ensuite,nous avons expliqué la morphologie de la langue arabe.

Le chapitre (2) de prétraitement de texte arabe, se propose de faire un état de l'art sur le prétraitement du texte arabe. Il approfondi, en particulier, le concept de racinisation.

Le chapitre (3) de l'implémentation d'un processus de prétraitement du texte arabe sur KNIME, présente le détail des étapes à suivre pour développer un nœud et l'intégrer dans KNIME. par la suite, le flux de prétraitement du texte développé est présenté.

Afin de valider les nœuds développés, le flux de prétraitement du texte arabe est utilisé dans une tâche de classification de texte (analyse de sentiments).
La conclusion générale couronne notre mémoire pour une synthèse du travail entretenu ainsi que des perspectives envisagées.

CHAPITRE 1

Intelligence artificielle et le TALN Arabe

1.1 Introduction

L'Intelligence artificielle est l'imitation de processus de l'intelligence humaine par des machines, en particulier des systèmes informatiques. Ces processus comprennent le raisonnement, l'apprentissage et l'extraction de connaissance, etc. L'intelligence artificielle est un grand domaine scientifique qui ne peut pas être limité en un chapitre, mais nous allons la présenter brièvement.
Ce chapitre a pour objectif de présenter l'intelligence artificielle et son histoire. Par la suite nous abordons le traitement automatique du langage naturel (TALN), son but et ses tâches principales.
Finalement, nous nous intéressons à la morphologie de la langue arabe

1.2 Intelligence artificielle

L'intelligence artificielle (IA) détient actuellement le rôle le plus important dans l'informatique, et il progresse rapidement. Alors que la science-fiction dépeint souvent l'IA comme des robots ayant des caractéristiques humaines, l'IA peut englober des moteurs de recherche, systèmes de détection du fraude,voitures, armes, etc.

1.2.1 Introduction à l'intelligence artificielle

L'intelligence peut être simplement définie comme un ensemble de propriétés de l'esprit. Ces propriétés incluent la capacité de planifier, résoudre les problèmes, et en général, le raisonnement [Jones, 2008].
L'intelligence est la capacité de prendre la bonne décision au sein de nombreuses options possibles, faits et variables [Negnevitsky, 2005]. Cette définition (prendre la bonne décision) ne concerne pas seulement les humains, mais aussi les animaux qui ont prouvé qu'ils ont des comportements rationnels. Par exemple, les humains ont la capacité de communiquer par langues, mais certains animaux aussi. Les humains peuvent aussi résoudre des problèmes, mais la même chose peut être dite de certains animaux. Une différence alors est que les humains incarnent de nombreux aspects d'intelligence plus complexe que l'intelligence des animaux.
par conséquent, nous pouvons étendre le concept intelligence aux systèmes informatiques. Par exemple, il est possible de créer une application qui joue parfaitement le jeu d'échecs, mais ce programme ne sait rien du jeu de Dames, du mot croisé ou d'autre jeu. Une application d'exploration de données peut aider à identifier la fraude, mais ne peut pas naviguer dans un environnement complexe. De ce point de vue, les applications les plus complexes et les plus intelligentes peuvent être considérées comme intelligentes d'une perspective [Jones, 2008].

1.2.2 Historique de l'intelligence artificielle

L'histoire moderne de l'IA remonte à l'année 1956 lorsque John McCarthy a proposé le terme comme sujet d'une conférence tenue à Dartmouth College dans New Hampshire. Les objectifs initiaux pour le terrain étaient trop ambitieux et les premiers systèmes d'IA n'ont pas réussi à livrer ce qui avait été promis. Après quelques échecs, Les chercheurs de l'IA ont commencé à se fixer des objectifs plus réalistes. Dans les années 1960 et 1970, la recherche sur l'IA se concentrait principalement sur le développement des systèmes KBS (knowledge-based system) ou les systèmes experts.
Au cours de ces années, la technologie des systèmes experts a été appliquée à un large éventail de problèmes et de domaines allant du diagnostic médical à l'inférence de la structure moléculaire à la compréhension du langage naturel. La même période a également été témoin de travaux préliminaires sur les réseaux des neurones RN (Neural Network, NN) qui ont montré comment une structure distribuée d'éléments pouvait représenter

collectivement un concept individuel avec l'avantage supplémentaire de la robustesse et du parallélisme. Cependant, la publication du livre (Perceptrons) [Papert and Minsky, 1969], qui plaidait pour les capacités de performance limitées du NN, a conduit à la disparition de recherche en RN dans les années 1970.
La fin des années 1980 et les années 1990 ont vu un regain d'intérêt pour la recherche NN lorsque plusieurs chercheurs [Rumelhart et al., 1986] ont réinventé l'algorithme d'apprentissage de la rétro- propagation (bien que l'algorithme ait été découvert en 1969). Cet algorithme a été bientôt appliqué à beaucoup de problèmes d'apprentissage provoquant une grande excitation au sein de la communauté de l'IA.
Les années 90 ont également connu des changements spectaculaires dans le contenu et la méthodologie de la recherche sur l'IA. L'objectif du domaine a été de s'orienter vers l'ancrage des méthodes d'IA sur une base mathématique rigoureuse, ainsi que de s'attaquer aux problèmes du monde réel et pas seulement aux exemples de jeux. Il y a également un mouvement vers le développement de systèmes hybrides intelligents [Goonatilake and Khebbal, 1994] (c'est-à-dire, des systèmes qui utilisent plus d'une méthode IA) provenant de la reconnaissance du fait que de nombreuses méthodes IA sont complémentaires. Les systèmes intelligents hybrides ont également commencé à utiliser des paradigmes plus récents qui imitent le comportement biologique tels que les GA (Genetic Algorithms) [Sadek, 2007].

1.2.3 Définition de l'intelligence artificielle

Nous pouvons diviser la définition de l'intelligence artificielle en quatre types [Russell and Norvig, 2016] :

- **Penser humainement**
  - Le nouvel effort passionnant pour faire penser les ordinateurs, machines avec esprit, dans le sens plein et littéral [Haugeland, 1985].
  - L'automatisation des activités que nous associons à la pensée humaine, des activités telles que la prise de décision, la résolution de problèmes, l'apprentissage [Bellman, 1978].
- **Penser rationnellement**
  - L'étude des facultés mentales à travers l'utilisation de modèles calculatoire [Charniak and McDermott, 1985].
  - L'étude des calculs qui permettent de percevoir de raisonner et d'agir [Patrick,

1992].

- **Agir humainement**
  - L'art de créer des machines qui exécutent des fonctions qui exigent de l'intelligence lorsqu'elles sont exécutées par des personnes [Kurzweil et al., 1990].
  - L'étude d'orienter les ordinateurs de faire des choses auxquelles les gens sont meilleurs pour le moment [Rich and Knight, 1991].
- **Agir rationnellement**
  - L'intelligence calculatoire est l'étude de la conception d'agents intelligents [Poole et al., 1998].
  - l'intelligence artificielle est concerné par le comportement intelligent dans la machine, ce comportement à son tour implique la perception, le raisonnement, l'apprentissage, la communication et l'action dans des environnements complexes [Nilsson, 1998].

1.3 Traitement automatique du langage naturel (TALN)

Le traitement automatique du langage naturel (Natural Language Processing, NLP) est une branche de l'intelligence artificielle (Figure 1.1). C'est un domaine de recherche et d'application qui explore comment les ordinateurs peuvent être utilisés pour comprendre et manipuler des textes ou des discours en langage naturel afin de faire des objectifs utiles. Les chercheurs du TALN cherchent à rassembler des connaissances sur la façon dont les êtres humains comprennent et utilisent le langage, afin que des outils et des techniques appropriés puissent être développés pour que les systèmes informatiques comprennent et manipulent les langues naturelles pour accomplir les tâches désirées. Les fondements de TALN se trouvent dans un certain nombre de disciplines, à savoir, sciences informatiques, linguistique, mathématiques, génie électrique, électronique, biomédical et psychologie, etc.[Chowdhury, 2003].

1.3.1 Définition de traitement automatique du langage naturel

Définition 1

Le Traitement automatique du langage naturel est une gamme théoriquement motivée de techniques de calcul permettant d'analyser et de représenter des textes naturels à un ou plusieurs niveaux d'analyse linguistique dans le but d'obtenir un traitement du langage semblable à celui d'un être humain pour diverses tâches ou applications [Liddy, 2001].

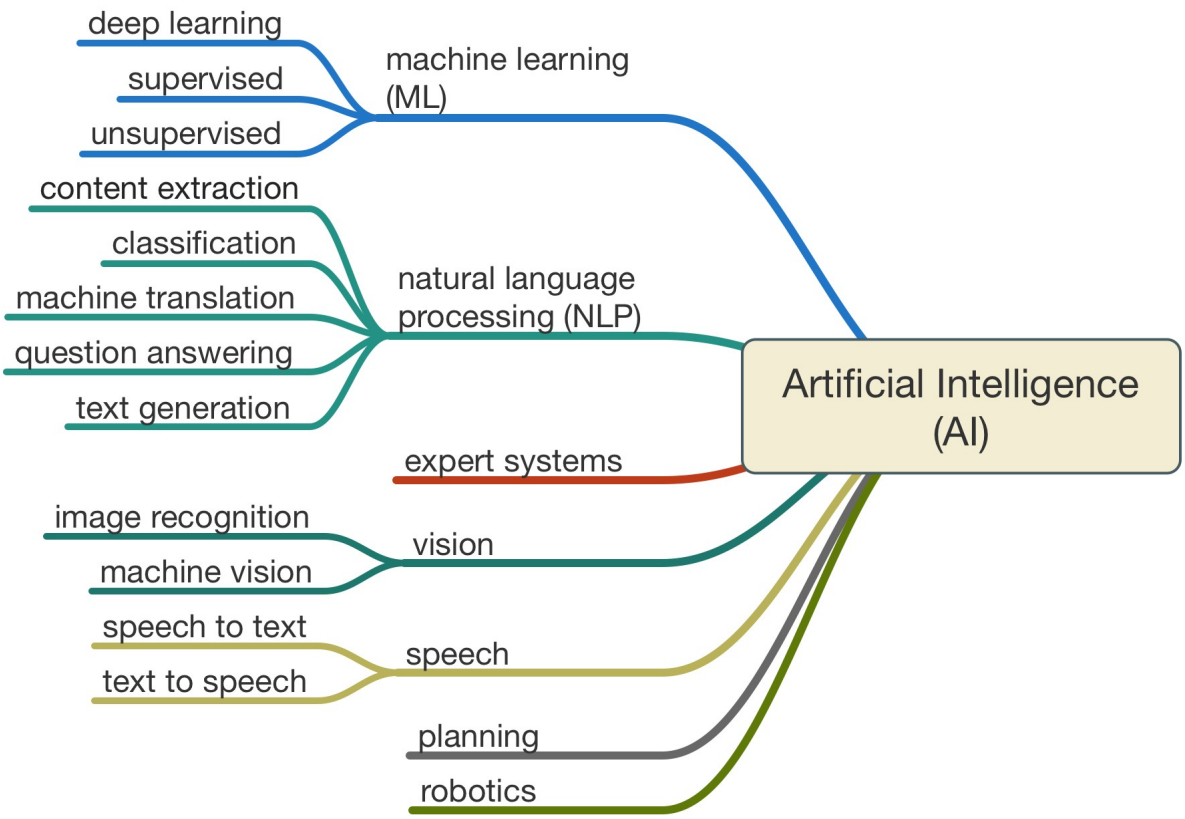

FIGURE 1.1 – Partie des branches de l'intelligence artificielle. Extrait de [MSE, 2017]

Définition 2

Le traitement automatique du langage naturel est un domaine de l'informatique et de la linguistique calculatoire qui s'intéresse aux interactions entre les ordinateurs et les langages humains (naturels). Le TALN est liée au domaine de l'interaction homme-machine. De nombreux défis dans Le TALN impliquent la compréhension du langage naturel, permettant aux ordinateurs de dériver le sens d'entrée en langage naturel, et d'autres impliquent la génération de langage naturel [Bharti and Malhotra, 2016].

1.3.2 But et tâches principales de traitement automatique du langage naturel

Le but de TALN est d'accomplir un traitement du langage semblable à l'humain. Le choix du mot « traitement » est très délibéré et ne devrait pas être remplacé par « comprendre ». Bien que le domaine de TALN ait été à l'origine appelé CLN, l'abréviation de compréhension de la langue naturelle (en anglais : Natural Language Understanding) dans les premiers jours de l'IA [Liddy, 2001] .

Ce qui suit est une liste de certaines tâches les plus fréquemment recherchées dans le TALN [Bharti and Malhotra, 2016].
Notez que certaines de ces tâches ont des applications directes dans le monde réel :

- **Résumé automatique**
  Résumer avec précision du texte (des articles, livres, films, conversations, réunions, courriels, etc) est le processus d'abstraction de grands textes en quelques paragraphes tout en préservant son contenu d'information, et en capturant les points de vus, sentiments, opinions, couleurs, etc). [Azmi and Al-Thanyyan, 2012].
- **Traduction automatique**
  Traduire automatiquement le texte d'une langue humaine à une autre. C'est l'un des problèmes les plus difficiles, et fait partie d'une classe de problèmes familièrement appelés (AI-complete) [Yampolskiy, 2012], c'est-à-dire nécessitant tous les différents types de connaissances que possèdent les humains (grammaire, sémantique, faits sur le monde réel, etc..) afin de résoudre correctement ce probléme [Koehn, 2009].
- **Analyse de sentiments et d'opinion**
  — Analyser la couverture médiatique et les médias sociaux pour comprendre les sentiments et les tendances sur les marques, les actions, les célébrités, les politiciens, les médias, et être capable de comprendre le contenu complexe qui fait référence à plusieurs entités qui utilise le sarcasme, l'humour et le contexte.[Vinodhini and Chandrasekaran, 2012]
  — Utilisez l'analyse des sentiments pour détecter le bien-être personnel et interpersonnel (Est-ce qu'une personne est stressée, anxieuse, fâchée, déprimée).
- **Répondre aux questions**
  la tâche de répondre aux questions (question answering, QA) classé dans (AI-complete), à l'objectif de comprendre des questions humaines complexes, être capable d'identifier une source crédible pour la réponse, puis trouvez la bonne réponse [Weston et al., 2015].
  Par exemple, la capacité de répondre à cette question complexe : « Je cherche un lieu de vacances du ski pour 6 personnes, nous voulons un hôtel 4 ou 5 étoiles, je veux sortir des montagnes de ski à pied, et assure qu'il y ait de fortes chutes de neige à la fin du mois de mars ».
  Etre capable de répondre à des questions complexes à partir des emails, photos, documents, feuilles Excel, messages de chat, historique de localisation, etc.

- **Sous-titrage d'image (collaboration avec la reconnaissance visuelle)**
  Être capable de décrire avec précision les détails de toute entrée visuelle, et de répondre à des questions complexes à ce sujet. Par exemple pouvoir trouver n'importe quelle photo qui contient la phrase « ma photo avec mon fils en France quand il avait 3 ans ».
- **Classification et Clustering**
  La classification et le clustering de documents est l'activité qui consiste à classer de façon automatique des ressources documentaires, généralement en provenance d'un corpus.
  le clustering est le rassemblement de documents dans des catégories homogènes [Aggarwal and Zhai, 2012a], alors que La classification est associée à chaque texte une catégorie (classe nommée) existante [Aggarwal and Zhai, 2012b].

1.3.3 Importance du traitement automatique du langage naturel

Avec l'utilisation accrue de l'Internet et l'utilisation fréquente des courriers électronique, des articles de journaux ou de sites Web, des rapports internes, des transcriptions d'appels téléphoniques, des documents de recherche, des entrées de blogue et des demandes de brevet, pour n'en nommer que quelques-uns, il est devenu difficile, presque impossible de traiter tous ces textes et données manuellement. Ces données sont considérées comme une richesse ne peut en aucun cas être négligée. La Figure 1.2 montre l'explosion des données (non structurées) entre 2015 et 2017.

Grâce au web et aux médias sociaux, plus de 7 millions de pages web du texte sont ajoutées à notre dépôt collectif tous les jours. Nous pouvons maintenant commencer à voir l'utilité d'un logiciel qui peut « lire » entre 15 000 et 250 000 pages par heure, comparé à seulement soixante pages pour les humains [Guernsey, 2003].

1.4 Morphologie de la langue arabe

L'alphabet de la langue arabe compte 28 lettres de l'alphabet (Figure 1.3). L'arabe s'écrit et se lit de droite à gauche. Les lettres changent de forme de présentation selon leur position (au début, au milieu ou à la fin du mot). Toutes les lettres se lient entre elles sauf (ا، و، ر، ز، د، ذ) qui ne se joignent pas à gauche.

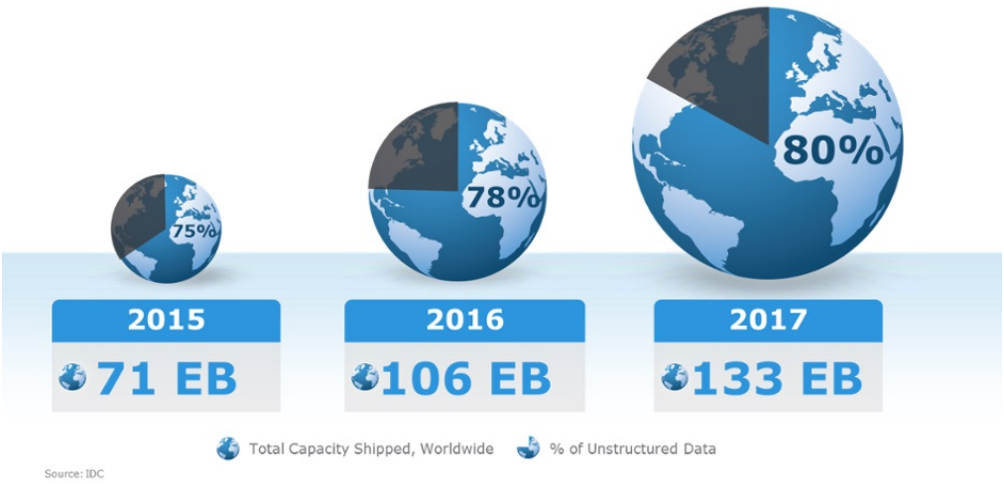

FIGURE 1.2 – Croissance des données non-structurées. Extrait de [IDC, 2017].

خ Kh	ح Haa'	ج Jiim	ث Thaa'	ت Taa'	ب Baa'	أ 'Alif
ص Saad	ش Shiin	س Siin	ز Zaayn	ر Raa'	ذ (Th)aal	د Daal
ق Qaaf	ف Faa'	ع Ghayn	ع 'Ayn	ظ (Th)aa'	ط Taa'	ض Daad
ي Yaa'	و Waaw	ه Haa'	ن Nuun	م Miim	ل Laam	ك Kaaf

FIGURE 1.3 – les 28 lettres arabes.

Un mot arabe s'écrit avec des consonnes et des voyelles. Toutes les lettres dans la langue arabe sont des consonnes. mais les consonnes arabes ne peuvent pas produire un son complet sans les marques de voyelles. Les voyelles sont ajoutées au-dessus ou au-dessous des lettres (Figure 1.4), elles sont nécessaires à la lecture et à la compréhension correcte d'un texte, elles permettent de différencier des mots ayant la même représentation.

La figure 1.5 donne un exemple pour les mots كتب et مدرسة. Les voyelles ne sont utilisées que pour des textes sacrés (Le Coran) et didactiques. Les textes courants rencontrés dans les journaux et les livres n′en comportent habituellement pas. De plus certaines lettres comme ا Alef peuvent symboliser le آ , أ , إ De même que pour les lettres ى et ه qui symbolisent respectivement ي et ة.

L'arabe est une langue sémitique, la plupart de ses mots sont construits à partir de

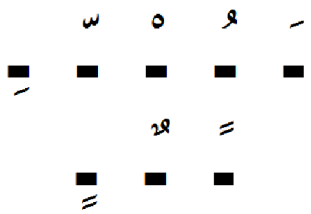

FIGURE 1.4 – Les diacritiques de la langue arabe.

Mot sans voyelles	1ère Interprétation		2ème Interprétation		3ème Interprétation	
كتب	كَتَبَ	il a écrit	كُتِبَ	Il a été écrit	كُتُبْ	des livres
مدرسة	مَدْرَسَةٌ	école	مُدَرِّسَةٌ	enseignante	مُدَرَّسَةٌ	enseignée

FIGURE 1.5 – Ambiguïté causée par l'absence de voyelles pour les mots كتب et مدرسة.

racines en suivant certains modèles connus et en ajoutant des préfixes, des suffixes et des infixes ou nous pouvons analyser ces mots en racines, ces racines sont des mots composés de trois à cinq lettres de consonnes. L'arabe est construit sur trois genres : masculin, féminin et neutre. Il contient également trois personnes, le locuteur, la personne à laquelle on s'adresse et la personne qui n'est pas présente. L'arabe peut décrire le nombre d'objets en utilisant trois modèles : singulier, dual et pluriel. Tous ces attributs sont pris en compte lors de la construction de l'ensemble de motifs[Saad and Ashour, 2010].

1.5 Conclusion

Dans ce chapitre, nous essayons de faire un bref aperçu sur l'intelligence artificielle et sur le TALN et leur importance. En suite la langue arabe et sa morphologie.
Dans le deuxième chapitre, nous nous focalisons sur le processus de prétraitement du texte arabe, et les travaux liés à ce processus.

CHAPITRE 2

Prétraitement de texte arabe

2.1 Introduction

Il existe différentes études intéressées par le problème de prétraitement des textes dans toutes les langues, soit l'anglais ou le français ou bien autre langue maternelle. Mais il y a moins d'études intéressées par le prétraitement du texte arabe. Ce chapitre a pour objectif de présenter les étapes et le processus de prétraitement du texte, avec une revue générale sur les travaux connexes lié au prétraitement des textes arabe. le plan de chapitre est sur la figure (2.1)

2.2 Processus de prétraitement de texte arabe

Le traitement du texte varie d'une langue à une autre. En fonction de la morphologie de la langue cible. De même au niveau de chaque langue il y a plusieur différences dans le processus de traitement selon l'utilisation et le besoin.
En générale le processus de prétraitement de texte arabe est basé sur les étapes qui suivent [Azmi and Al-Thanyyan, 2012] :

2.2.1 Tokenisation

La tokenisation est le processus de découpage du texte courant en mots et en phrases (Figure 2.2). Le texte électronique est une séquence linéaire de symboles (caractères ou

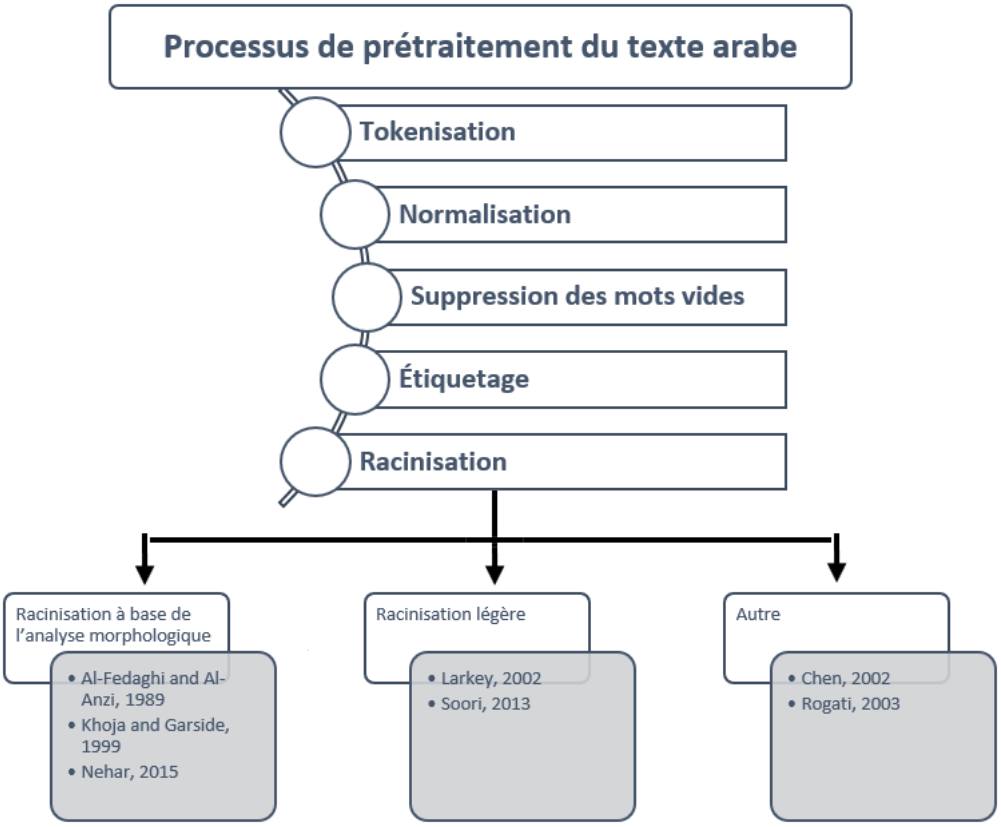

FIGURE 2.1 – Plan de chapitre 2.

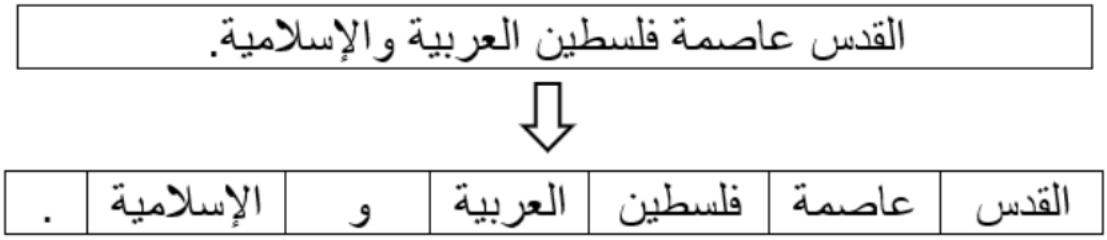

FIGURE 2.2 – Tokenisation de la phrase (al-quds est la capitale de la palestine arabe et islamique).

,	¡	:	"	'	÷
×	º	>	<	|	\
¿	!	@	#	$	%
^	&	*	)	(	_
-	+	=	.	,	;
~	ø	؟	،	§	?

FIGURE 2.3 – Liste de ponctuation

mots ou phrases). Naturellement, avant tout traitement de texte réel, le texte doit être segmenté en unités linguistiques telles que les mots, la ponctuation, les nombres, les caractères alphanumériques, etc [Hassler and Fliedl, 2006].

2.2.2 Normalisation (Standardisation)

La normalisation est le processus de transformation du texte dans un format standard plus facilement manipulable. Le texte arabe est normalisé comme suit [Khoja and Garside, 1999] :

— **Suppression des caractères spéciaux et les chiffres.**
A ce niveau, on supprimer les chiffres, les signes diacritiques (Figure 1.4) et les ponctuations (Figure (2.3)). Il est a noter que dans certains élements sont remplacés par des espaces.

— **Suppression des mots et des caractères étrangers.**

— **Suppression les lettres simples**
les mots d'une lettre en arabe et les abréviations.
Par exemple : la numérotation (paragraphe «B» , الفقرة ب) abréviations de date (ميلادي: م , هجري: ه).

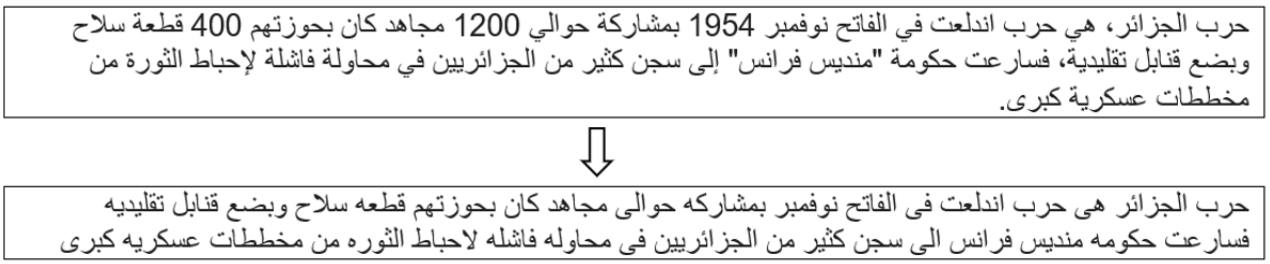

FIGURE 2.4 – Exemple de normalisation d'un texte arabe.

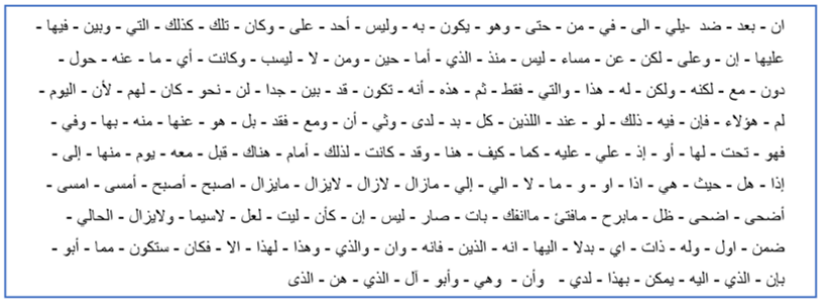

FIGURE 2.5 – liste de mots vides arabe.[Khoja and Garside, 1999]

— **Remplacement de إ, آ et أ avec ا**
— **Remplacement de la lettre finale ي avec ى**
— **Remplacement de la lettre finale ة avec ه**

Cette étape est nécessaire à cause des variations qui peuvent exister lors de l'écriture d'un même mot arabe. Un exemple de Normalisation de texte arabe est montré par la la Fugire 2.4 .

2.2.3 Suppression des mots vides

Cette étape consiste à éliminer tous les mots vides (Stop-words) correspondant aux termes non porteurs d'information utile qui figurent dans un texte. En effet, les mots vide disposent d'une occurrence très fréquente et qui n'apportent aucune valeur ajoutée. Parmi ces mots vides :

— Les pronoms (أنا , نحن , انتما, هن...)
— Les prépositions (من , عن , على , في , كـ...)
— Les conjonctions (ثم , أم , أو , حتى...)

Généralement [Khoja and Garside, 1999] ont proposé une liste de mots vides de 168 mots représenté dans la Figure (2.5) connue et utilisée dans plusieurs algorithmes de prétraitement du texte arabe.

2.2.4 Étiquetage (Marquage)

L'étiquetage (Tagging) est l'opération qui vise à étiqueter chaque mot avec une étiquette morphosyntaxique unique qui indique son rôle syntaxique, par exemple, pluriel, adverbe, etc [Toutanova et al., 2003].
Si nous nous assemblons tous les étiquettes morphosyntaxiques de l'arabes, ce sera plus de 200 [Mace, 1999]. Pour cette raison [Khoja et al., 2001] choisi seulement les classes et sous-classes principales et elle les a représentés dans la Figure 2.6.

2.2.5 Racinisation (l'extraction de radicaux)

La racinisation (Stemming) est l'étape la plus importante dans le prétraitement de texte arabe ou bien d'autre langues. C'est une tâche délicate car l'arabe est une langue inflexible. Le manque de signes diacritiques dans la plupart des textes écrits en arabe crée une ambiguïté et exige donc des règles morphologiques complexes. De plus, les mots en lettres majuscules ne sont pas utilisés en arabe, ce qui rend difficile l'identification des noms propres, des acronymes et des abréviations.
La racinisation est le processus qui consiste à trouver la racine du mot. La racine est extraite après avoir supprimé les suffixes et les préfixes attachés à un mot donné. De nombreuses stemmer ont été développées pour l'anglais et d'autres langues. Les algorithmes de racinisation les plus connus de la langue anglaise sont introduits [Lovins, 1968] et [Porter, 1980]. Par ailleur il existe d'autres algorithmes de racinisation comme [Paice, 1980], [Dawson, 1974] , [Krovetz, 1993] et N-Gram Stemmer. Cependant, ils sont peu utilisés en se comparant aux deux premiers. Ces algorithmes sont classés dans trois groupes : méthodes de troncature, méthodes statistiques, et méthodes mixtes (Figure 2.7) [Jivani, 2011].

Pour la langue arabe, il y a plusieurs approches de racinisation peut être regroupé en 3 groupes : la racinisation à base de l'analyse morphologique de la langue (en anglais : root-based stemmer) et la racinisation légère (en anglais : light stemmer) et enfin autre techniques de racinisation (statistique, clustering..) [Al-Omari, 2014].

2.2.5.1 Racinisation à base de l'analyse morphologique

Ce groupe de racinisation utilise l'analyse morphologique pour extraire la racine d'un mot arabe donné. Plusieurs algorithmes ont été développés pour cette approche. La racine est extraite après suppression des suffixes et des préfixes attachés au mot donné, ensuite

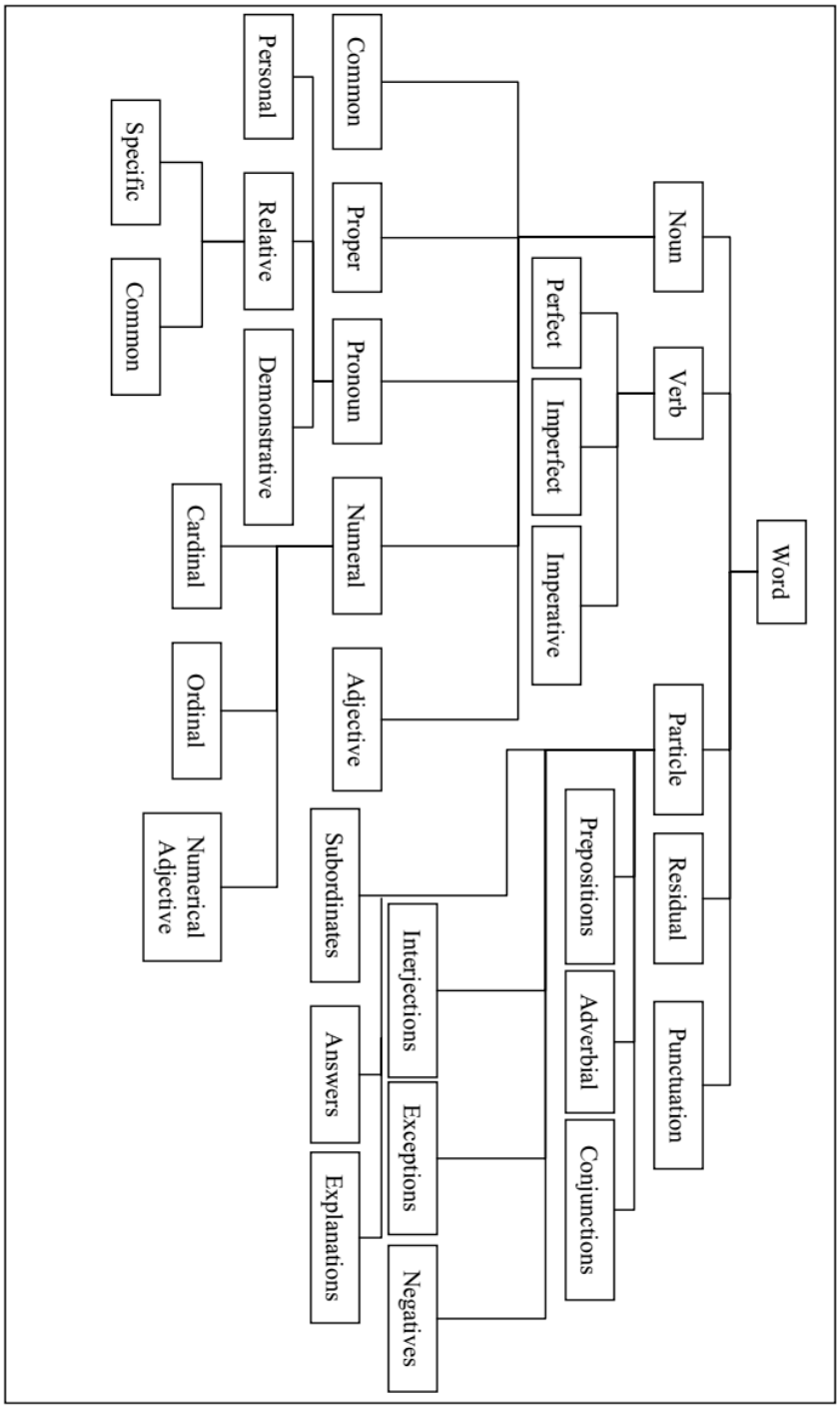

FIGURE 2.6 – Hiérarchie des tags morphosyntaxiques arabe.

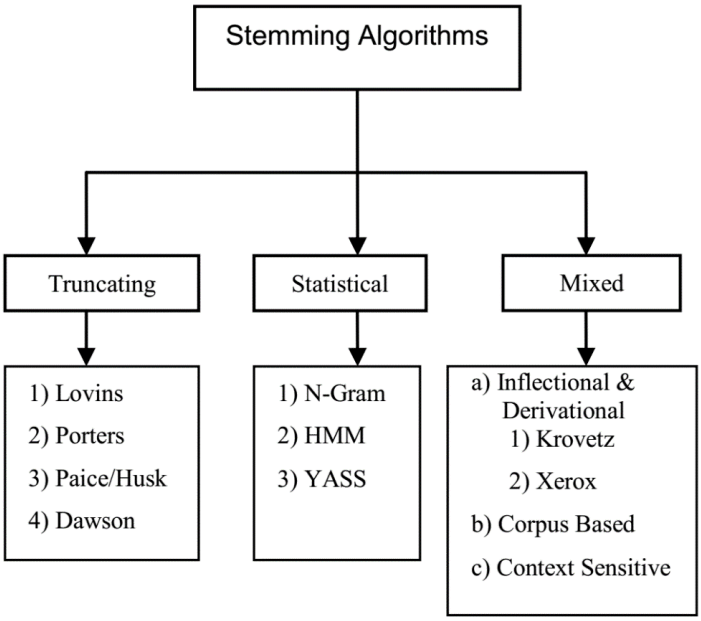

FIGURE 2.7 – Types d'algorithmes de stemming de langue anglaise.

le processus d'extraction de racine est démarré en faisant correspondre les positions des lettres de mot de surface qui correspond à un modèle. L'exemple de Figure 2.8 décrit le processus de correspondance de modèle. Après avoir extrait les lettres correspondant au modèle (فعل), ces lettres représentent la racine [Syiam et al., 2006].

Certains chercheurs décrivent la langue arabe comme un langage complexe, qui signifie que le processus d'analyse morphologique est dur et difficile. Les noms et les verbes en arabe sont générés à partir d'un ensemble de racines - environ 11 347 racines – classifiées selon leurs longueur comme suit [Kanaan, 2008] :

— Racines de 2 lettres : 115 racines.
— Racines de 3 lettres : 7198 racines.
— Racines de 4 lettres : 3.739 racines.
— Racines de 5 lettres : 295 racines.

Ceci dit, il existe plusieurs problèmes en inhérent de la langue arabe qui compliquent le processus de racinisation. Quelques exemples de ces problèmes sont [Al-Omari, 2014] :

1. Un mot peut avoir plusieurs significations selon sa position et le contexte du mot dans le texte, par exemple le mot (عين Ayn) :
   — (شربت من عين الماء) Ce qui signifie : j'ai bu d'une fontaine.

— (عين الله ترعاك) Ce qui signifie : dieu te bénisse.
— (أمراض كثيرة تصيب العين) Ce qui signifie : de nombreuses maladies infectent les yeux.
— (أصبت عين الحقيقة) Ce qui signifie : tu a visé le centre de la vérité.
— (عين الوزير مساعده) Ce qui signifie : le ministre a nommé son assistant.

2. Certains mots ont la même racine mais avec significations différentes. Par exemple, tous les mots suivants ont la même racine (حسب) :
   — (حسب) Ce qui signifie : calculer.
   — (حاسوب) Ce qui signifie : ordinateur.
   — (محاسبة) Ce qui signifie : Comptabilité.
   — (حساب) Ce qui signifie : arithmétique.
3. Certaines lettres originales dans un mot sont changées dans leur forme, ces lettres dans la plupart des cas sont les voyelles (ا,و,ي) :
   — (صام) de (صوم) où (و) s'est convertie en (ا).
   — (سار) de (سير) où (ي) s'est convertie en (ا).
   — (رعى) de (رعي) où (ي) s'est convertie en (ى).
4. Certaines lettres sont originales dans un mot et ont une fonction dans un autre mot. Par exemple la lettre (و) dans les mots suivants :
   — La lettre ici est originale :
     — (ورق) Ce qui signifie : papier.
     — (وصل) Ce qui signifie : arriver.
     — (لون) Ce qui signifie : Couleur.
   — La lettre signifie ici (Et) :
     — (درس وكتب) Ce qui signifie : Étudier et écrire.
     — (أكل وشرب) Ce qui signifie : Manger et boire.

Parmi les travaux connexes a l'approche de racinisation à base de l'analyse morphologique, nous pouvons citer les travaux de [Al-Fedaghi and Al-Anzi, 1989] et [Khoja and Garside, 1999] et [Nehar, 2015].

2.2.5.1.1 Al-Fedaghi and Al-Anzi [1989]

[Al-Fedaghi and Al-Anzi, 1989] tentent de trouver la racine du mot en faisant correspondre le mot avec tous les modèles possibles avec tous les affixes possibles qui s'y rattachent.

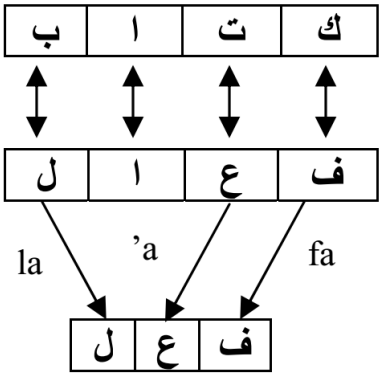

FIGURE 2.8 – Extraire les lettres racines du mot (كتاب) par correspondance de modèle.

L'algorithme ne supprime aucun préfixe ou suffixe. Il appartient aux approches dites Approches combinatoires (Combinatorial approaches). L'algorithme consiste à comparer les mots testés avec des listes préparées des racines, modèles, affixes. La comparaison est basée sur un algorithme combinatoire qui teste toutes les combinaisons de trois lettres d'un mot donné afin d'extraire la racine (Figure 2.9).

En général, de telles approches sont simples mais prennent beaucoup de temps et nécessitent de très grandes listes. L'approche a été appliquée sur (Quran, Tawfeeq, Aqad, Textes d'actualité en arabe) et a obtenu une Précision de 72% (moyenne).

2.2.5.1.2 Khoja and Garside [1999]

[Khoja and Garside, 1999] a développé un algorithme qui supprime les préfixes et suffixes, tout en vérifiant qu'il ne supprime pas une partie de la racine, puis correspond le mot restant aux motifs de la même longueur pour extraire la racine.

L'algorithme de Khoja suit cette procédure :

— Supprimer les signes diacritiques (Figure 1.4).
— Supprimer les mots vide, la ponctuation et les chiffre.
— Supprimer l'article défini (ال).
— Supprimer la conjonction inséparable (و).
— Supprimer les suffixes.
— Supprimer les préfixes.
— Faire correspondre le résultat à une liste de modèles (أوزان). Si une correspondance est trouvé, extraire les caractères dans le modèle représentant la racine.
— Remplacer les lettres faibles (ا,و,ي) par (و).
— Remplacer toutes les occurrences de hamza (ء,ؤ,ئ) par (أ).

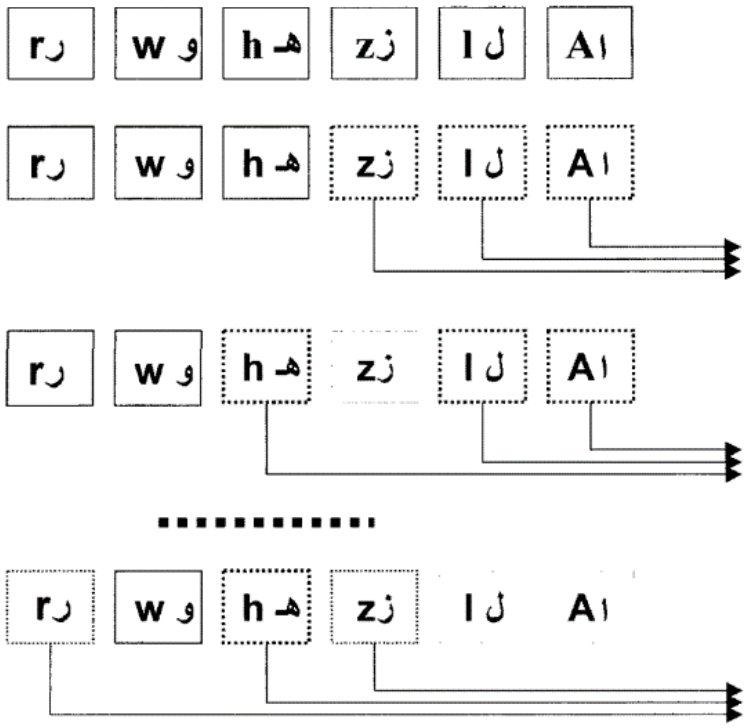

FIGURE 2.9 – Exemple de l'approche combinatoire.

— Les racines à deux lettres sont vérifiées pour voir si elles doivent contenir un double caractère. Si oui, le caractère est ajouté à la racine.

Les problèmes rencontrés par cet algorithme sont :

— Si la racine contient une lettre faible (ا,و,ي), la forme de cette lettre peut changer pendant la dérivation. Pour faire face à cela, le stemmer doit vérifier pour voir si la lettre faible est dans la forme correcte. Si ce n'est pas le cas, le stemmer produit la forme correcte de cette lettre faible, qui donne alors la forme correcte de la racine.
— Certains mots n'ont pas de racines. Par exemple (نحن, بعد, تحت). Si le stemmer rencontre un de ces mots, il ne fait rien.
— Parfois, une lettre racine est supprimée lors de la dérivation. Ceci est particulièrement vrai pour les racines qui ont des lettres en double (c'est-à-dire que les deux dernières lettres sont identiques). Le stemmer peut le détecter et renvoyer la lettre qui a été supprimée.
— Si une racine contient un hamza(ء) qui pourrait changer de forme pendant la dérivation. Le stemmer le détecte et renvoie la forme originale de ce hamza.

L'algorithme de Khojaa été appliquée sur textes arabe classique et il rapporte de bons résultats en terme de précision (96%).

2.2.5.1.3 Nehar [2015]

[Nehar, 2015] propose une approche Tstemmer qui fait appel aux transducteurs pour modéliser les préfixes, motifs des mots et suffixes. Elle permet d'extraire la racine d'un mot en générent les racines potentielles du mot c'est-à-dire modéliser toutes les racines qui répondent aux formes préfixe-modèle-suffixe.
Les transducteurs de mots sont une forme généralisée des automates finis. En effet, chaque transition dans l'automate fini est augmentée par une étiquette de sortie en plus de son étiquette d'entrée. Les étiquettes en sortie sont assemblées le long d'un chemin de l'état initial vers un état final, pour former une chaine de sortie.
Un mot en arabe répond, en général, à une forme préfixes-modèle-suffixes. Pour cela, Tstemmer modélise aussi les préfixes et suffixes des noms et verbes. Il y a 4 préfixes de verbes (ت,ا,ي,ن) et 12 préfixes de noms (م,ن,و,ي,ل,لل,ف,س,ت,ب,ال,ا) et plus de 20 suffixes : (ها,ان,كما,تما
ي,ات,ا,ت,ة,ه,ك,ن,ني,تي,ته,هن,هم,ين ,ون,ما,تا,نا,كن ,تن,كم ,تم,وا). (Figure 2.10-2.11)

Il faut suivre les étapes suivantes pour construire le transducteur d'extraction des racines :

1. Construire le transducteur de tous les préfixes de noms (resp. Préfixes de verbes), Figure 2.10.
2. Construire le transducteur de tous les modèles de noms (resp. Modèles de verbes), Figure 2.12.
3. Construire le transducteur de tous les suffixes de noms (resp. Suffixes de verbes), Figure 2.11 .
4. Concaténer les transducteurs obtenus dans les étapes 1 et 3 avec le transducteur des noms (resp. Le transducteur des verbes) dans l'ordre de leur construction.
5. Concaténer les deux transducteurs obtenus de l'étape 4.

Il est clair que la composition de Tstemmer avec un transducteur d'un mot donné Tterme donne lieu à un transducteur pouvant inclure de nombreux chemins, par conséquent, de nombreuses racines possibles. En effet, un mot arabe pourrait correspondre à plus d'un modèle en même temps. Prenons le mot انتصر (gagner). Ce mot correspond, au

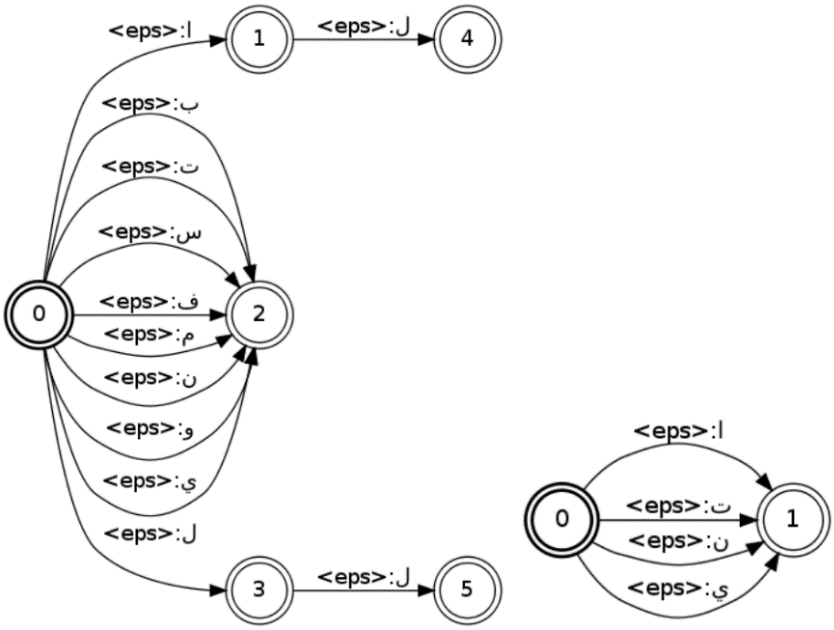

FIGURE 2.10 – Transducteurs des préfixes de noms (gauche) et les préfixes des verbes (droite).

moins, à deux modèles : انفعل et افتعل donnant les racines تصر et نصر respectivement. Ainsi, l'utilisation de Tstemmer conduit à un ensemble contenant une ou plusieurs racines possibles. Cependant, il est assuré que la racine correcte appartient à cet ensemble des racines possibles. Afin de contourner cette situation d'indéterminisme, le transducteur Tstemmer d'extraction nécessite une pondération.

Le Tstemmer a été appliquée sur 3 collections de mots (table 2.1). Les résultats obtenus de la comparaison entre Tstemmer et Khojastemmer en terme d'exactitude sont présentés dans la Table 2.2.

Collection	Mots	Note
Gold1	679	Un échantillon prélevé du corpus de l'arabe contemporaine [Sawalha, 2011].
Gold2	844	Des ensembles recueillis manuellement
Gold3	1,000	Des ensembles recueillis manuellement
Total	2,523	

TABLE 2.1 – Détails des collections de mots.

Les problèmes rencontrés par cet algorithme :

— Comme dans l'algorithme de khoja, Tstemmer fait des erreurs si la racine contient une lettre faible (أ,و,ي) .

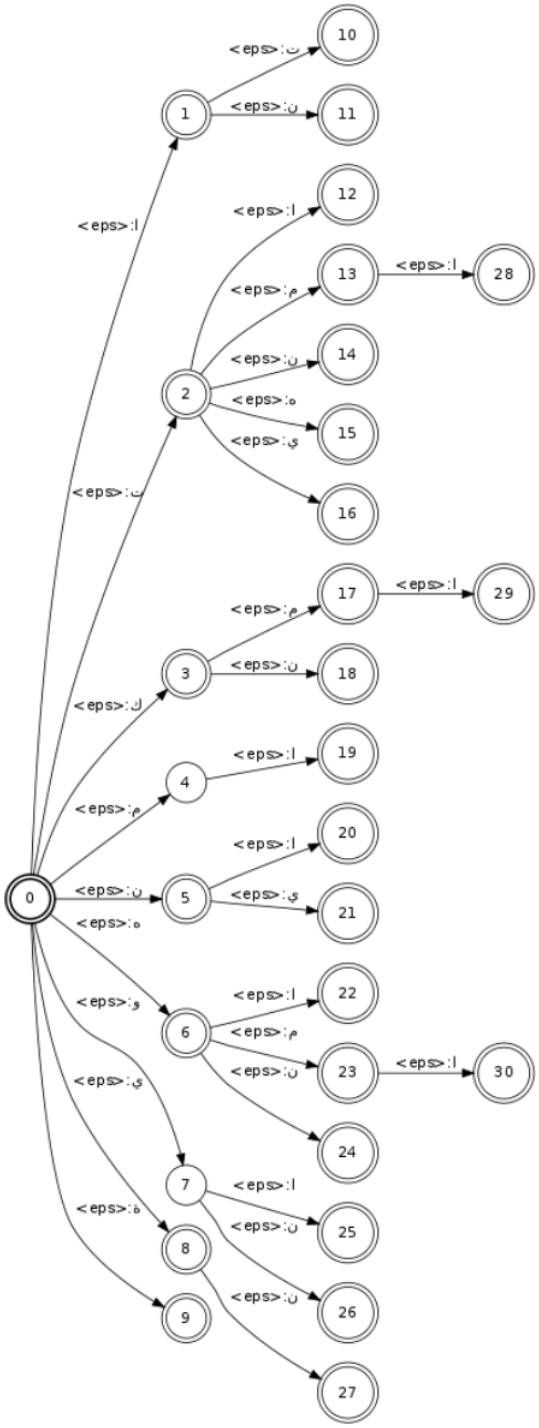

FIGURE 2.11 – Transducteur des suffixes de noms et des verbes.

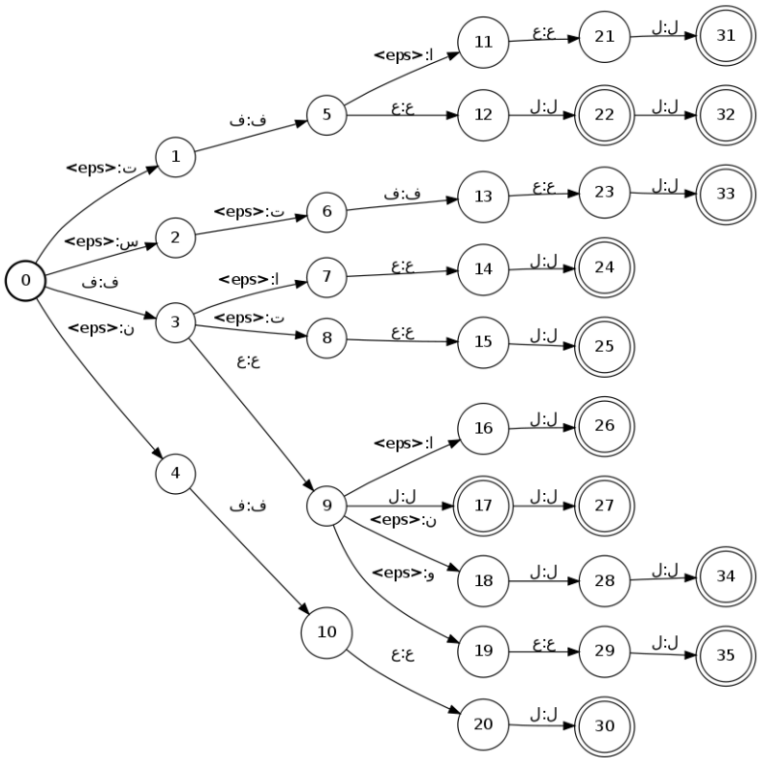

FIGURE 2.12 – Transducteur des modèles des verbes.

Collection	Tstemmer	Khojastemmer
Gold1	71,68	82,77
Gold2	74,82	85,55
Gold3	80,30	87,60
Moyenne	75,60	85,30

TABLE 2.2 – Résultat de comparaison de Tstemmer et Khojastemmer en terme d'exactitude [Nehar, 2015].

— stemmer échoue pour les mots ayant des préfixes/suffixes composés (plus d'un préfixe/suffixe). Par exemple (Humanité الإنسانية) .

2.2.5.2 Racinisation légère

Cette section s'intéresse aux méthodes d'extraction de racine dites légères.
Selon [Porter, 1980] La racinisation consiste à extraire la racine des mots et assembler les mots similaires dans la même racine. Donc, le nombre de termes sera réduit, mais le plus gros problème accompagne cette approche est l'ambiguïté. Pour cela la notion de racinisation légère (en anglais : light stemming) est évoquée dans plusieurs travaux, elle consiste à éliminer juste les préfixes et les suffixes d'un mot donné d'une manière simple

, sans avoir à remonter à sa racine.

2.2.5.2.1 Larkey [2002]

[Larkey, 2002a] a créé un groupe d'algorithmes de racinisation légère (light stemmers), à savoire Light 1, 2, 3, 8, et 10. Le dernier Light10 stemmer a surpassé toutes les versions précédentes.

Le principe de Larkey était d'essayer d'enlever les affixe qui peuvent être trouvés au début ou à la fin d'un mot arabe.

[Larkey, 2002a] propose plusieurs versions de stemming, selon les étapes suivantes :

— Supprimez (و) (et) pour light2, light3 et light8 si le reste du mot comporte 3 caractères ou plus. Bien qu'il soit important d'enlever (و) , c'est aussi un probléme, car beaucoup de mots arabes commencent par ce caractère. d'où le critère de longueur plus strict ici que pour les articles définis.

— Supprimer un des article défini si cela laisse 2 caractères ou plus.

— Parcourer la liste des prefixes et suffixes une fois dans l'ordre (de droite à gauche) indiqué dans la Table 2.3.

Version	**Prefixes**	**Suffixes**	**mots vides**
light1	ال،وال،بال، كال،فال	aucun	aucun
light2	ال،وال،بال، كال،فال،و	aucun	aucun
light3	aucun	ه،ة	aucun
light8	aucun	ها,ان,ات,ون,ين, يه, ية,ه,ة,ي	aucun
light10	ال،وال،بال، كال،فال،لل،و	ها,ان,ات,ون,ين,يه,ية,ه,ة,ي	168 mots

TABLE 2.3 – Les affixes enlevées par [Larkey, 2002a].

Le stemmer de [Larkey, 2002a] a été appliquée sur le corpus arabe TREC-2001, appelé aussi corpus AFP_ARB, composé de 383.872 articles de journaux en arabe de l'Agence France Presse.

La Figure 2.13 montre la comparaison entre des variantes du Light stemmer et des variantes de Khojastemmer.

2.2.5.2.2 Soori [2013]

[Soori, 2013] propose un simple algorithme de racinisation légère. Cet algorithme est divisé en cinq étapes :

1. Normaliser les mots en utilisant un processus qui permet de convertir tous les différents types de (أ) et (إ) en (ا) et changer (ى) par (ي).

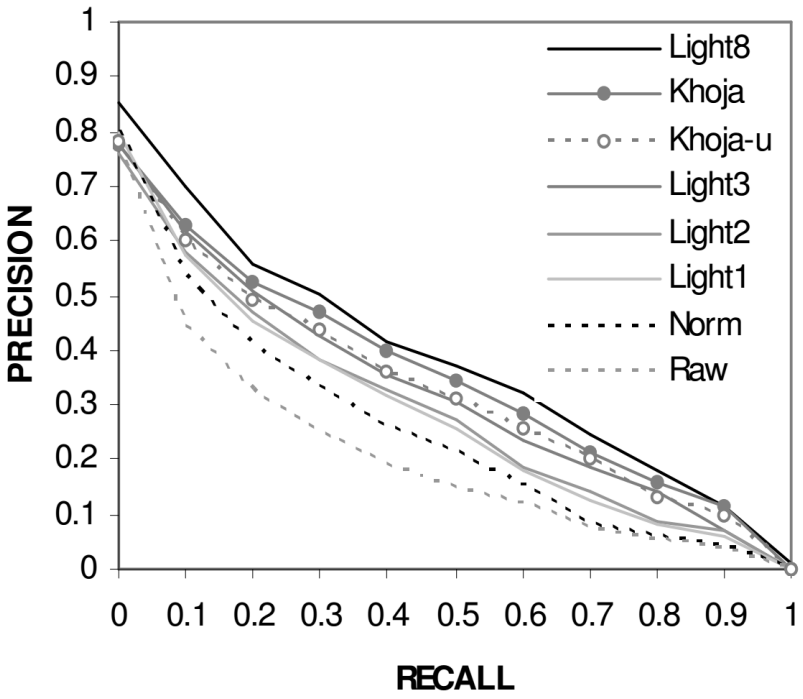

FIGURE 2.13 – Comparaison entre Larkeylight et Khojastemmer [Larkey, 2002a].

Recall = rappel ; precision = précision ; khoja-u = Khoja stemmer avec l'ajout de la liste incassable des termes exemptés de stemming ; Raw = pas de normalisation ; norm = avec normalisation.

2. Supprimer tous les préfixes possibles (préposition, articles de définition et conjonctions) comme suit :
   - Supprimer (ب, و) de début de chaque mot, si le reste est 3 caractères ou plus.
   - supprimez les caractères liés suivants au début d'un mot, si le nombre de caractères restant est supérieur à trois caractères.
     — Supprimer (ا) et (ل).
     — Supprimer (ا),(و). et (ل).
     — Supprimer (ا),(ب). et (ل).
     — Supprimer (ا),(ك). et (ل).
     — Supprimer (ا),(ف). et (ل).
3. Supprimer tous les suffixes possibles, si le reste est composé de deux caractères ou plus (ها, ان, ات, ون, ين, يه, ية).
4. Modifier les adjectifs et les adverbes de la forme féminine à la forme masculine, Supprimer (ة, ه) si trouvé comme le dernier caractère d'un mot.
5. Changer les noms du pluriel en forme singulière. Dans cette règle, les caractères liés suivants sont supprimés si le reste est supérieur à 2 caractères. Si le mot ne comporte que trois caractères, supprimez les caractères suivants si les conditions

supplémentaires sont remplies

- Supprimez (ا) au début de chaque mot.
- Supprimez (ا, و) s'il est trouvé comme le caractère avant dernier.
- Supprimez (ا) du début de chaque mot, seulement si le caractère avant ernier est aussi (ا).
- Supprimer (ا) s'il est trouvé en tant que troisième ,si le mot comporte quatre caractères.

6. Supprimer les caractères suivants à la fin du verbes, si le reste est d'au moins 2 caractères :
   - Supprimer (ت, ي, ا).
   - Supprimer (ا) si trouvé comme un deuxième caractère dans un mot,seulement si (ا) est trouvé comme un dernier caractère d'un mot.
   - Supprimer (و, م).
   - Supprimer (نا, تا, تم, تي, ما, نتا,).
   - Supprimer (ا) si c'est la seconde dans un mot, Supprimer (ت) à la fin du mot.
   - Supprimer (تا) .

L'algorithme de [Soori, 2013] a été appliqué sur quelques articles de presse, de la BBC Arabic et Al Jazeera Arabic news portails et a obtenu une exactitude de 78%.

2.2.5.3 Autre techniques de racinisation

Ce groupe de racinisation comprend les techniques de statistique ou clustering, les mots apparentés sont groupés en fonction de diverses mesures de similarités. Les classes d'équivalence peuvent être formées à partir de mots qui partagent une lettre initiale n-gramme ou en affinant ces classes avec des techniques de clustering.

2.2.5.3.1 Chen [2002]

[Chen, 2002] a construir un algorithme base a une machine de traduction automatique (en anglais : Machine Translation Toolkit - MTTK). En fonction de MTTK, le mot arabe est traduit en anglais, ensuite, tous les mots arabes confondus avec la même racine anglaise (après la traduction) forment le même groupe (exemple de clustering dans la Figure 2.14). Tous les mots arabes dans un cluster sont changés par le mot plus court dans le cluster.

[Chen, 2002] utilise Ajeeb et Almisbar (`http://www.almisbar.com`) comme un outil de traduction automatique. Le stemmer de [Chen, 2002] a été appliquée sur Le corpus arabe TREC-2001, et a obtenu une exactitude de 87.94% .

Arabic word	English translation	Arabic word	English translation	Arabic word	English translation	Arabic word	English translation
أطفال	children	اطفالهن	their children	بطفل	by child	فالطفلة	then the child
أطفالا	children	اطفالي	my children	بطفلة	by child	فطفل	then child
أطفالنا	our children	الأطفال	children	بطفلتنا	by our child	كأطفال	as children
أطفاله	and his children	الاطفال	children	بطفلته	by his child	كالطفل	as the child
أطفاله	his children	الطفل	the child	بطفله	by his child	لأطفال	to children
أطفالها	her children	الطفلان	the children	بطفلها	by her child	لطفلها	to her child
أطفالهم	their children	الطفلة	the child	بطفلهما	by their child	للّطفلة	to the child
أطفالهن	their children	الطفلتان	the children	بطفلين	by children	وأطفالنا	and our children
أطفالي	my children	الطفلتين	the children	بطفليها	by her children	والأطفال	and the children
اطفال	children	الطفله	the child	طفل	child	وبطفل	and by child
اطفالا	children	الطفلين	the children	طفلا	child	وبطفلين	and by children
اطفالك	your children	بأطفال	by children	طفلان	children	وطفلة	and child
اطفالكم	your children	بأطفاله	by his children	طفلاها	her children	وطفلتان	and children
اطفالكن	your children	بأطفالها	by her children	طفلة	child	وطفلنا	and our child
اطفالنا	our children	بالأطفال	by the children	طفلت	child	وطفلها	and her child
اطفاله	his children	بالطفل	by the child	طفلتان	children	وطفليه	and his children
اطفالها	her children	بالطفلة	by the child	طفلتة	his child	وطفليها	and her children
اطفالهم	their children	بالطفلتين	by the children	طفلتنا	our child	ولأطفالها	and to her children
اطفالهما	their children	بالطفلين	by the children	طفلته	his child	وللّطفل	and to the child

FIGURE 2.14 – Cluster de mots arabes dont les traductions anglaises contiennent le mot «child» ou «children»

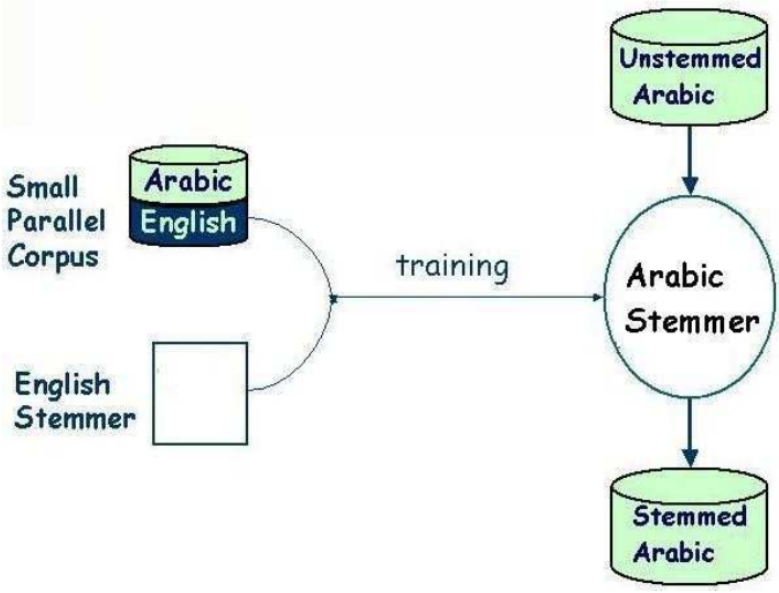

FIGURE 2.15 – Présentation de l'approche de Rogati et al.

2.2.5.3.2 Rogati [2003]

[Rogati, 2003] présente une approche d'apprentissage non supervisée pour la construction d'un algorithme de racinisation non-anglais (arabe) Figure 2.15. Le modèle de racinisation est basé sur la traduction automatique statistique (Statistical Machine Translation - SMT) et utilise un stemmer anglais et un petit corpus parallèle (10K phrases) de délibérations des Nations Unies, journaux bilingues, etc. (la Figure 2.16 montrer un extrait d'une conférence des Nations Unies) comme unique ressource de formation. Aucun texte parallèle n'est nécessaire après la phase d'entraînement. Le texte monolingue et annoté peut être utilisé pour améliorer encore le stemmer en lui permettant de s'adapter à un domaine ou un genre désiré.

L'application de cette approche a donnée des résultats satisfaisants pour la langue arabe, mais l'approche est applicable à toute langue nécessitant une suppression des affixes.

L'approche se traduit par un accord de 87,5% avec un stemmer arabe propriétaire construit à l'aide de règles, listes d'affixes et modeles.

L'approche de [Rogati, 2003] Est basée sur la disponibilité des trois ressources suivantes :

— Petit corpus parallèle.

— stemmer anglais.

— corpus arabe facultatif non annoté.

L'objectif est de former un stemmer arabe en utilisant ces ressources. Le stemmer

Arabic	English	عربية
m$rwE Altqryr	Draft report	مشروع التقرير
wAkdt mmvlp zAmbyA End ErDhA lltqryr An bldhA y$hd tgyyrAt xTyrp wbEydp Almdy fy AlmydAnyn AlsyAsy wAlAqtSAdy	In introducing the report, the representative of Zambia emphasised that her country was undergoing serious and far-reaching changes in the political and economic field.	وأكدت ممثلة زامبيا عند عرضها لتقرير عن بلدها يشهد تغييرات خطيرة وبعيدة المدى في الميدانين السياسي والاقتصادي

FIGURE 2.16 – Un minuscule corpus parallèle arabe-anglais [Rogati, 2003].

résultant soutiendra simplement l'arabe sans avoir besoin de son équivalent anglais.

2.3 Conclusion

Dans ce chapitre, nous avons exposé les étapes principales du processus de prétraitement du texte arabe, et nous avons expliqué certains travaux connexes sur les algorithmes de racinisation du mot arabe.

À partir de l'état de l'art que nous avons abordé dans ce chapitre, nous allons implémenté deux algorithmes de racinisation dans le chapitre qui suit, le premier de [Khoja and Garside, 1999] qui à base de l'analyse morphologique et le deuxième algorithme de [Larkey, 2002a] qui suit la méthode d'extraction de racine légère. Ensuite, nous développons une tache de filtrage des mots vides arabe.

CHAPITRE 3

Implémentation d'un processus de prétraitement du texte arabe sur KNIME

3.1 Introduction

Afin de mettre en pratique les connaissances théoriques acquises précédemment, nous présentons les grandes lignes du processus de prétraitement du texte arabe, Nous implémentons ce processus sur KNIME. KNIME, acronyme de Konstanz Information Miner est une plateforme modulaire pour la conception et l'exécution de flux de travail (workflow) en utilisant des composants prédéfinis appelés nœuds. KNIME est un logiciel gratuite open source développé par l'Université de Constance(Konstanz) en Allemagne.

Dans ce chapitre, nous présentons d'abord, les étapes nécessaires au développement d'un nœud dans KNIME. Ensuite nous expliquons comment exporter un noeud et l'ajouter a la plateforme KNIME. Après, nous présentons notre flux de prétraitement du texte arabe.

Enfin, nous montrons la validité de notre travail en utilisant notre chaîne de prétraitement dans une tâche de classification (Analyse de sentiments).

Nous utilisons la version KNIME Desktop 3.5.3 sur un PC Intel Core i7 avec un processeur de 2.40 GHz et 12 Go de mémoire centrale fonctionnant sous le système d'exploitation Windows 10.

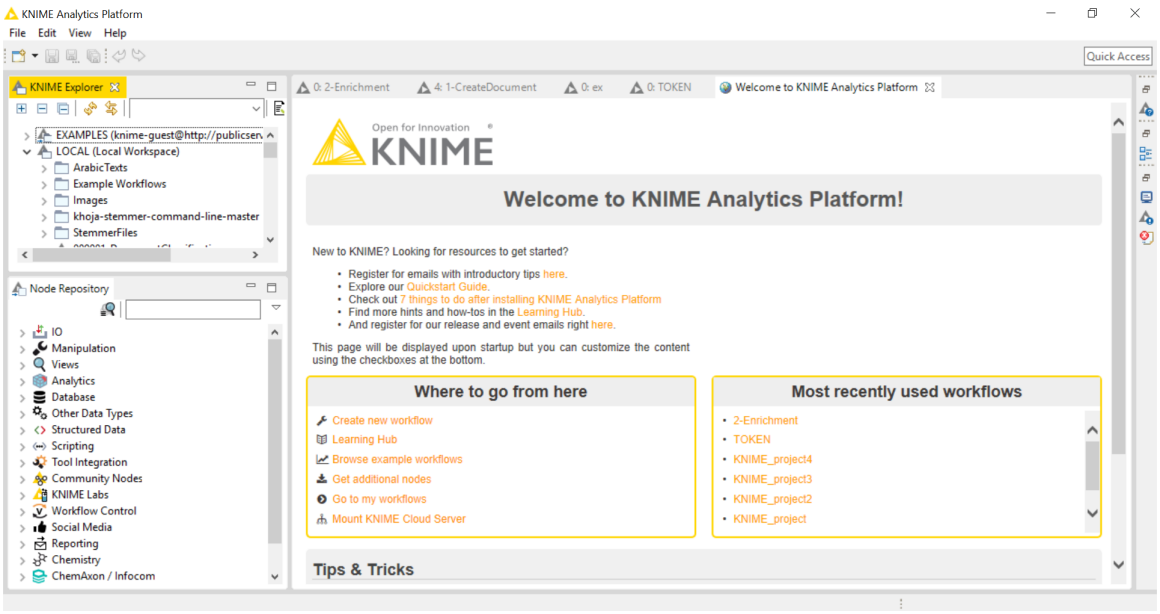

FIGURE 3.1 – Interface graphique de KNIME.

3.2 Outil KNIME

KNIME Analytics Platform est une plateforme d'analyse de données qui permet à l'utilisateur de construire un flot de traitement de données, comparer et modifier des algorithmes de fouille de données. Il dispose d'une interface utilisateur graphique pour combiner les nœuds. Les collections de nœuds sont connues comme des extensions [HASSANI, 2016].

C'est un logiciel open source sous la licence GPL(General Public License), écrit en langage JAVA sous la plate forme Eclipse SDK. KNIME a une interface graphique simple (Figure 3.1) et il a la possibilité d'acquérir des extensions selon le besoin [Berthold et al., 2009].

L'installation de la plateforme KNIME est disponible dans le lien `https://www.KNIME.com/downloads`. Le processus d'instalation est expliqué dans le lien `https://www.KNIME.com/installation-0`

3.3 Etapes de développement d'un nœud dans KNIME

En vu de réaliser le processus de prétraitement du texte arabe, nous avons développé trois nœuds, à savoir, un nœud KhojaStemmer ayant comme tâche de la racinisation

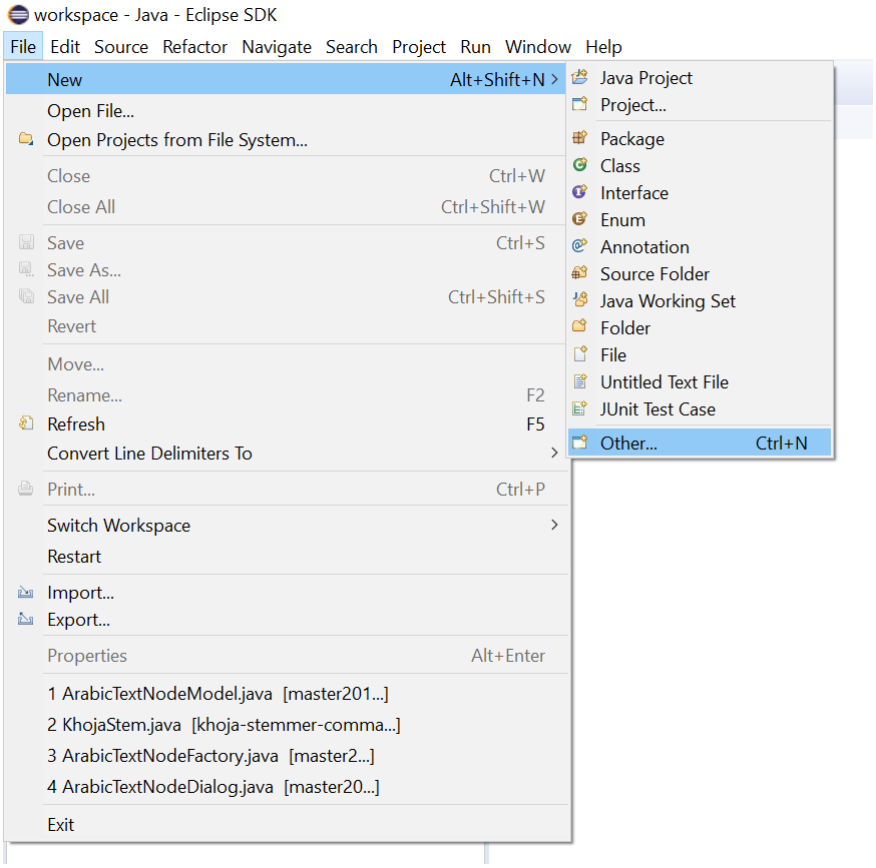

FIGURE 3.2 – Création d'un nouveau projet sur KNIME SDK.

d'un mot arabe en se basent sur l'analyse morphologique. Le deuxième nœud a pour objectif d'implémenter l'algorithme de racinisation légère de Larkey. Dans une perspective d'illustrer le processus de création d'un nouveau nœud dans KNIME, nous allons décrire dans ce qui suit, les étapes nécessaires pour développer le nœud KhojaStemmer.

1. Tout d'abord, télécharger et installer le KNIME SDK sur le lien https://www.KNIME.com/download-KNIME-analytics-platform-sdk. KNIME SDK (Software Development Kit) est fourni par KNIME. C'est une version d'Eclipse qui contient l'extension wizard pour simplifier le développement de nouveaux noeuds.

   Eclipse est un environnement de développement intégré (IDE), basé sur Java, utilisé pour le développement des logiciels. Eclipse est un environnement flexible et extensible. Lorsque KNIME SDK est installé, il ressemble beaucoup à une installation Eclipse.

2. Ouvrire KNIME SDK et créer un nouveau projet : File > New > Other (Figure 3.2)

3. Choisir KNIME > create a new KNIME Node-Extension (Figure 3.3).

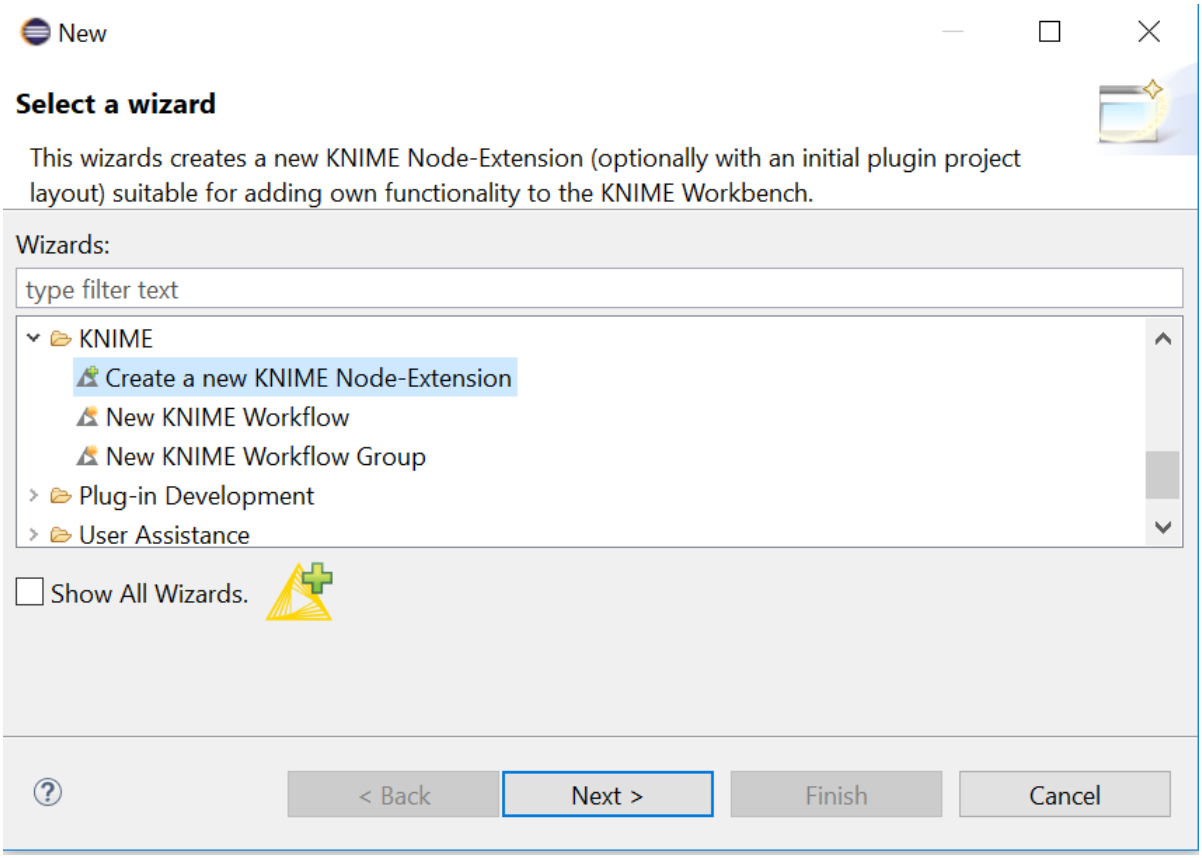

FIGURE 3.3 – Création d'un nouveau nœud sur KNIME SDK.

4. Remplir les données générales (KNIME extension settings) du projet (Figure 3.4).
5. Avec la création du projet, cinq classes sont créées automatiquement (par défaut) par KNIME SDK, ces classes sont monNodeFactory, monNodeView, monNodePlugin, monNodeModel et monNodeDialog, avec le changement de «mon» par le nom «Node class name» sélectionné à l'étape 4. Dans notre projet nos classes sont StemmerNodeModel, StemmerNodePlugin...
6. Ajouter les classes nécessaires pour le projet.
   Dans notre projet nous utilisons trois classes (Methode, ArabicStemmerKhoja, KhojaStem) contentent le code de l'algorithme de racinisation de Khoja. La Figure 3.5 montrer les différents classes de notre projet de création du nœud de racinisation
   - **Classe ArabicStemmerKhoja :** Dans cette classe on définit des listes comme suit :
     — Préfixes : liste des préfixes des verbes et des noms disponibles dans la langue arabe.
     — Suffixes :liste des suffixes des verbes et des noms disponibles dans la langue arabe.

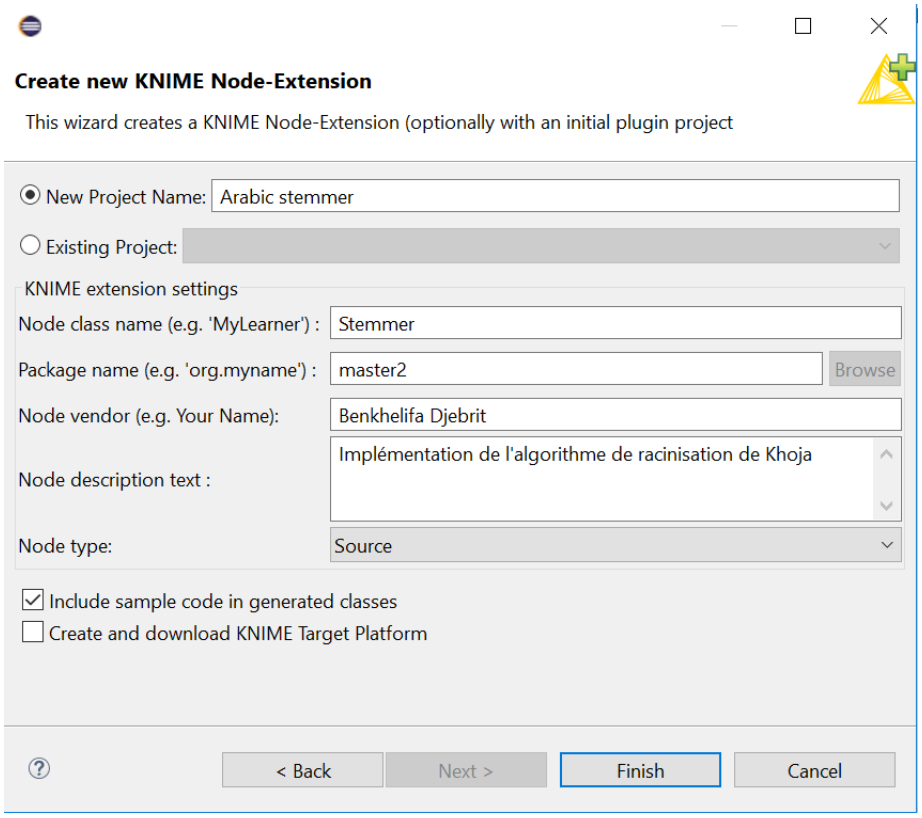

FIGURE 3.4 – Remplissage des informations générales du nœud.

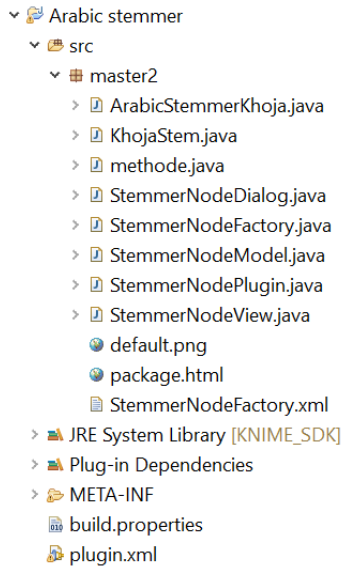

FIGURE 3.5 – Classes de projet de racinisation du text arabe.

— Dupliquer : liste de tous les mots en deux lettres dont le dernier caractère est un doublon (نخ , مد ...).
— Mots vide : liste de 168 mots vide, illustré dans la section 2.2.3.
— Racines de 3 et 4 lettres : liste de toutes les racines de longueur 3 et 4, par exemple (رسم, ترجم).
— Diacritiques : liste de Diacritiques montré dans la Figure 1.4.
— Modèles :liste des modèles disponibles en l'Arabe représentée dans la Table 3.1.

مستفعل	مفعول	مفعلّ	مفتعل	منفعل	متفاعل	متفعل	متفعّل
فعول	مفعال	فعّال	مفعل	مفاعل	مفعل	استفعال	افعلال
فعيل	أفعل	فعلان	فعلاء	تفعيل	فعال	افعال	تفعل
افتعال	انفعال	تفاعل	فعلى	يفتعل	فواعل	مفاعيل	فعيّل
افاعل	يستفعل	تفتعل	استفعل	فعائل	فعّل	فاعل	تفعّل
افعل	تفعل	تفاعل	انفعل	افتعل	افعلّ		

TABLE 3.1 – Liste de modèles de l'Arabe [Khoja and Garside, 1999].

— Etrange : liste de quelques mots étrangers qui ont été arabisés, nom propre ou les noms de pays par exemple (مانديلا, ألمانيا, التلفزة).
— Premier waw, Premier yah : liste des mots possibles qui sont incomplets d'un caractère au début du mot (manque و ou ي). Par exemple (صف son origine وصف et بس son origine يبس) .
— Mid waw, Mid yah : liste de tous les mots possibles qui sont incomplets d'un caractère au milieu du mot (manque و ou ي). Par exemple (هس son origine هوس et شع son origine شيع) .
— Dernier Alef, Dernier hamza, Dernier yah : liste des mots possibles qui sont incomplets d'un caractère a la fin (manque ا ou ء ou ي). Par exemple (طر son origine طرأ et در son origine درء et نس son origine نسي).

- **Classe KhojaStem :** C'est la classe la plus importante de notre projet où sont implémentées les méthodes de racinisation. Nous résumons le travail de cette classe dans un pseudo code présenter dans l'Algorithme 1.

- **Classe Methode :** Dans cette classe, nous avons segmenté le texte en mots 2.2.1. Pour chaque mot, nous appelons la procédure **StemWord** dans la classe

de KhojaStem.

7. Exécution de l'algorithme de racinisation a partir de KNIME.
   - Dans KNIME, les données se déplacent d'un noeud à l'autre sous forme de tables. Autrement dit, chaque nœud traite les donnes en entrée (table d'entrée) et produire des résultats (table de sortie).
   - Dans notre cas, pour exécuter la tâche de racinisation, nous utilisons la classe StemmerNodeModel. Spécifiquement nous faisons a la méthode *execute(BufferedDataTable[] inData, ExecutionContext exec)* ou une instance de classe «Methode» est passée comme argument a la classe StemmerNodeModel.
   - La méthode (*execute(BufferedDataTable[] inData, ExecutionContext exec)*). Cette méthode comporte 2 paramètres :
     - BufferedDataTable[] : la table qui contient les données, cette table est généralement le résultat d'un noeud précédent ou d'un noeud d'entrée.
     - ExecutionContext : Pour créer une table de sortie de type BufferedDataTable.

     Au cours du processus de création de la table, nous appelons la méthode *getstem()* de notre **classe Methode**, qui prend le contenu de la cellule de colonne précédente, et segmente le text aux mots. Elle applique la méthode stemWord de la classe de KhojaStem à chaque mot. Finalement, il réassemble les mots résultant du processus racinisation dans un seul texte, et le met dans la nouvelle cellule de nouvelle colonne, elle passe à la ligne suivante et effectue la même opération.
   - Dans notre projet, nous créons 3 nœuds tous basés sur les mêmes étapes précédentes. Un nœud **Khojastemmer** (Algorithme 1) de racinisation à base de l'analyse morphologique de la langue. Le deuxième noeud **Larkeystemmer** (Algorithme 2) a pour objectif de la racinisation légère. Enfin le noeud **Arabic stop words** (Algorithme 3) qui supprime les mots vides arabe.

Algorithm 1 Pseudo code de racineur de khoja

Input : Dataset, Stop-word list, Assets and patterns files
Output : Stemmed Data set

1 : Remplacer (إ, أ, آ) par (ا).
2 : Supprimer les mots vides.
3 : Supprimer la ponctuation, les non-lettres et les signes diacritiques.
4 : Supprimer les articles définis au début du mot.
5 : Supprimer la lettre (و) de début du mot et (ة) de la fin du mot.
6 : Supprimer le plus long préfixe et suffixe.
7 : Comparer le mot résultant avec la liste des modelés, si la racine résultante n'a pas de sens, le mot d'origine est renvoyé sans modifications.

Algorithm 2 Pseudo code de racineur de larkey

Input : data set,Stop-word list
Output : Stemmed data set

1 : Supprimer des mots vide.
2 : Remplacer (أ , إ , آ) par (ا).
3 : Convertir (ى) en (ي) et (ة) en (ه)
4 : Supprimez la lettre (و) au début du mot seulement si le mot résultant est plus de 3 lettres.
5 : Supprimer les articles définis.
6 : Supprimer les suffixes (illustré dans la Table 2.3).

Algorithm 3 Pseudo code de filtrage des mots vide

Input : data set
Output : data set without stop words

1 : Filtrer tous les mots vide définis dans la section 2.2.3, basé sur une liste de 903 mots assemblé par [Zerrouki, 2010].
2 : Supprime la ponctuation arabe.
3 : Supprimer les caractères de A-Z et a-z.
4 : Supprimer Les chiffres de format (٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩).

3.4 Exporter et déployer les nœuds sur Knime

Maintenant pour exporter et déployer le nœud en tant que plugin sur KNIME il faut suivre les étapes suivantes :

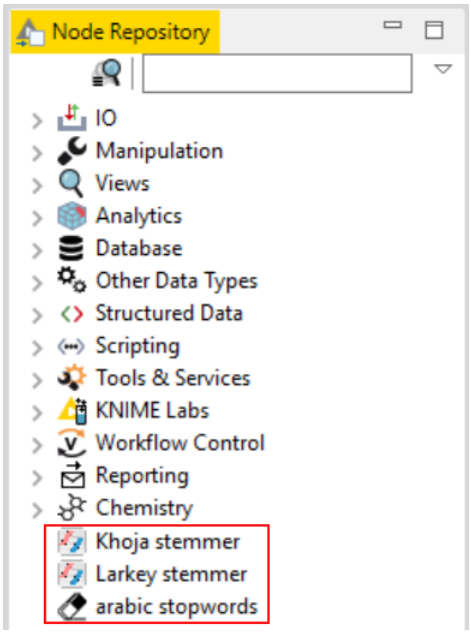

FIGURE 3.6 – Dépôt de nœuds de KNIME.

— Un clic droit sur le projet et on choisit : **Export >Plug-in Development > Deployable plug-ins and fragments**.
— Après avoir choisi le projet et l'endroit ou l'enregistrer, nous obtenons un fichier JAR (Java archive).
— Copier le fichier .JAR au dossier «plugins» sur le chemin d'installation de KNIME. Le chemin par default est : (*C :/ Program Files/KNIME/plugins*).
— Redémarrer KNIME.

La figure 3.6 montre l'addition des nœuds que nous avons développés dans l'entrepôt des nœuds de KNIME.

3.5 Flux de prétraitement du texte arabe

L'objectif de la présente section est de mettre en exergue notre contribution au développement d'un flux de prétraitement du texte arabe. En effet nous avons développé des nœuds KNIME dont la tâche est spécifique au traitement du texte arabe.

Les nœuds Khojastemmer et Larkeystemmer proposent deux méthodes de racinisations du texte arabe. Alors que le nœud Arabic stop words est développer afin de filtrer les mots vides arabes.

La Figure 3.7 montre l'utilisation de ces nœuds au sein des nœuds KNIME prédéfinis pour réaliser un flux de prétraitement du texte arabe complet.

Pour mieux comprendre le flux de prétraitement du texte arabe, nous décrivons brièvement l'ensemble des nœuds dons la figure 3.7.

- **Flat File Document Parser**

  Ce nœud vous permet de lire des fichiers texte plats et de créer un document pour chaque fichier. Le titre du document sera la première phrase du fichier et le texte intégral du texte restant contenu dans le fichier. Le répertoire spécifié sera recherché pour tous les fichiers, si la recherche récursive est activée, les sous-répertoires seront également recherchés. Tous les fichiers contenus sont analysés. Les données du document sont stockées dans la table de données sortante. Non seulement ce nœud dédié à l'entrée de texte et de donnée, Il existe de nombreux nœuds personnalisés à cet effet représentés sur la Figure 3.8.

- **Arabic StopWords**

  Nous avons programmé ce nœud pour avoir 4 filtres à la fois :

  1. Filtrer tous les mots vide définis dans la section 2.2.3, basé sur une liste de 903 mots assemblé par [Zerrouki, 2010].
  2. Dans l'étape précédente nous avons utilisé un nœud (Punctuation Erasure) pour filtrer la ponctuation. Ce nœud est prédéfini dans KNIME et supprime uniquement la ponctuation latine. Donc nous avons besoin d'ajouter une liste de ponctuation arabe tels que le point d'interrogation (؟) sachant que celui latin s'écrit (?), et le prolongement d'un caractère (الجزائـــــــــــر).
  3. Supprimer les caractères de A-Z et a-z.
  4. Supprimer Les chiffres de format (٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩), parce qu'il y a ceux qui utilisent ces chiffres, surtout au Moyen-Orient

  Ce processus est très efficace et donne un excellent résultat à 100% de précision. En effet nous avons divisé le mot en caractères et comparé chaque caractère individuellement avec la iste de ponctuation ou la liste des lettres (A-Z,a-z), donc même si la ponctuation ou la lettre est à l'intérieur du mot, il sera supprimé.

- **Number Filter** Filtre tous les termes contenus dans les documents d'entrée composés de chiffres, y compris les séparateurs décimaux «,» ou «.» et possible «+» ou «-» .

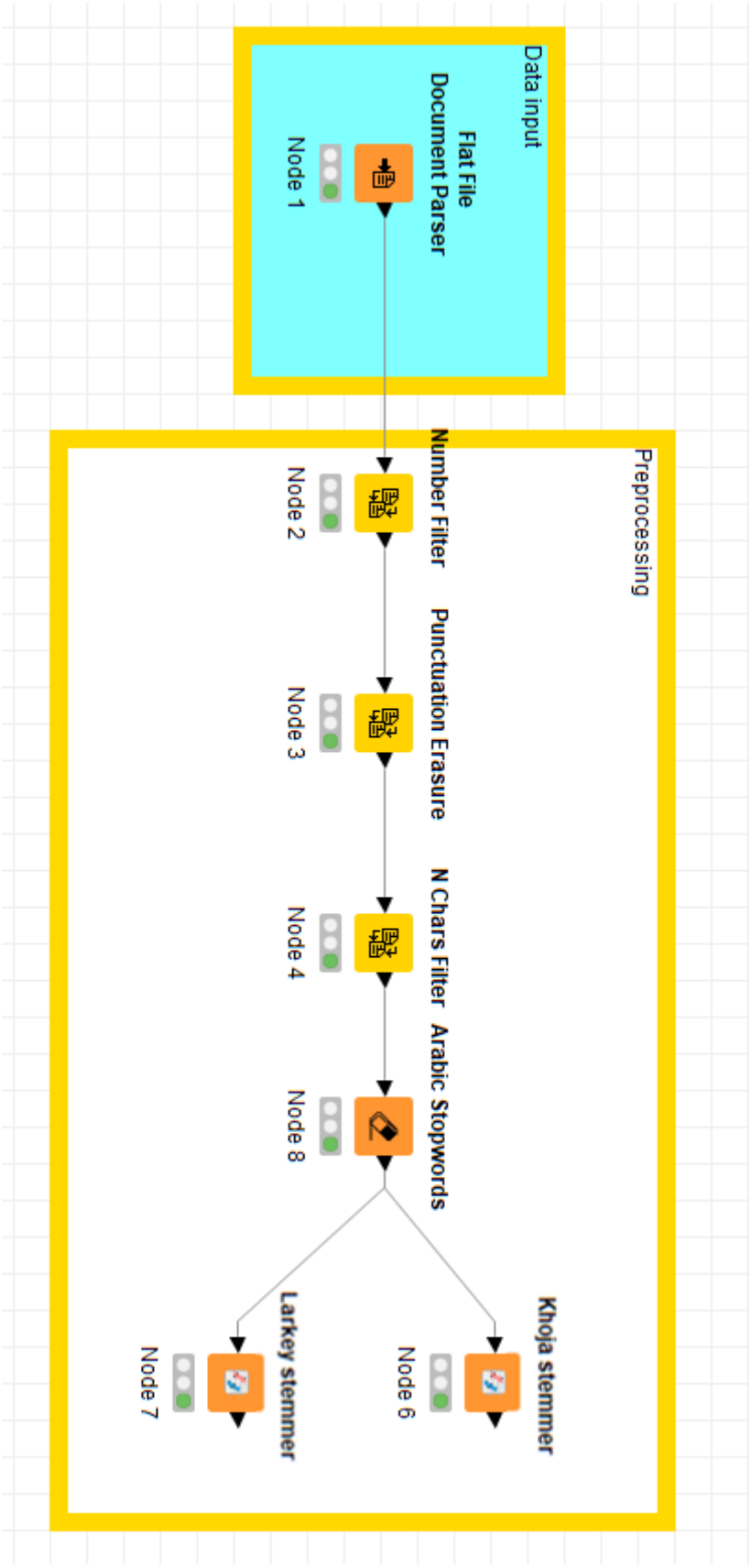

FIGURE 3.7 – Chaîne de prétraitement du texte arabe.

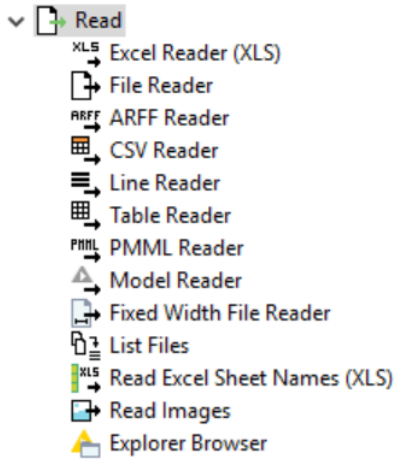

FIGURE 3.8 – Nœuds d'entrée de données.

- **Punctuation Erasure** Supprime tous les caractères de ponctuation des termes contenus dans les documents d'entrée.

- **N Chars Filter** Ce nœud a pour objectif de filtrer tous les termes contenus dans les documents d'entrée avec un nombre inférieur a un entier N donné.
  On a utilisé ce nœud pour supprimer les caractères individuels dans le texte, par exemple : الفقرة ب , 2018م .

- **Arabic stemmer**
  la racinisation est l'étape la plus importante dans le processus de traitement des textes arabes, parce qu'après les opérations de filtrage précédentes, la sortie de ce processus peut être utilisée dans différents domaines du TALN tels que la vérification orthographique, la compression et l'analyse du texte, etc [Khoja and Garside, 1999].
  Nous avons proposé deux nœud KhojaStemmer et LarkeyStemmer qui implémentent respectivement les algorithmes de racinisation à base de l'analyse morphologique de [Khoja and Garside, 1999] et la racinisation légère de [Larkey, 2002a]. La Figure 3.9 montrer le resultat d'utilisation de deux noeuds.

3.6 Expérimentation

Dans cette section, nous montrons la validité de travail en utilisant notre flux de prétraitement du texte arabe dans une tâche de classification.

S text	S Larkey stemmer	S Khoja stemmer
ممتازة بصراحه مريحة خاصتا في الخرجات الي من غير عربية	ممتاز بصراح مريح خاصتا في خرج ال من غير عرب	ميز صرح روح خوص خرج عرب
عملي	عمل	عمل
جيد نوع	جيد نوع	جود نوع
سلعه ودفعت ثمنها من الماستر كارد وتم ابلاغي انه تم ارجا...	اشتريت سلع دفعت ثمن من ماستر كارد وتم ابلاغ ان تم ارجاع ولا اعلم سبب	شري سلع دفع ثمن ستر كرد بلغ يتم ولي سبب
شنطة جميلة وأنيقة وسعرها مناسب جدا	شنط جميل انيق سعر مناسب جدا	شنط جمل نوق سعر نسب
حلوه	حلو	حلي
أو المحجر كما يسمى) ممتاز ... يوم لعب بأمان خاصة ل...	و محجر كما يسمى) ممتاز ... يوم لعب بام خاص اطفال رضع اقل من 3 سنوات ...	سرر حجر يسمى ميز يوم لعب مون طفل رضع قلل 3 سنوات
مفيده	مفيد	فود

FIGURE 3.9 – Résultats des racineurs Khoja et Larkey

Nous avons choisi l'analyse des sentiments car elle a fait l'objet de nombreuses études de recherche en raison de la grande variété de ses applications potentielles [ElSahar and El-Beltagy, 2015].
Dons ce qui suit, nous présentons les grandes lignes de notre expérimentation :

3.6.1 dataset utilisés

Nous utilisons un data set de 4272 commentaires (avis, témoignages..) extraits du site de souq.com affilié dans Amazon.com. Le data set est disponible sur le lien : https://github.com/hadyelsahar/large-arabic-sentiment-analysis-resouces/blob/master/datasets/PROD.csv. Il est utilisé principalement pour l'analyse des sentiments clients concernant les produits du site web de concerné.

Les commentaires (avis) sont regroupés en trois groupes, Le premier groupe contient les commentaires des clients qui ont interagi positivement avec le produit (إيجابي), ce groupe contient 3101 commentaires positifs, soit 72,6 % du nombres totale de commentaires. Le deuxième groupe est pour les commentaires des clients qui se sont dérangés par les marchandises (سلبي), ce groupe contient 863 commentaires négatifs, soit 20,2% du nombre total de commentaires. Le dernier groupe est pour les clients qui n'ont pas annoncé leurs sentiments (محايد), ce groupe contient 308 commentaires neutres, soit 7,2% du nombre total de commentaires.

3.6.2 Apprentissage et validation

Pour mener nos expérimentations, nous utilisons deux méthodes de classification, Arbre de décision (AD) et les machines à vecteurs de support (support vector machine, SVM). Pour chaque méthode, nous considérons le dataset sans prétraitement, avec prétraitement de racineur de Khojaet le prétraitement de racineur de Larkey. Par conséquent, nous conduisons six expérimentations, les Figures (3.10, 3.11, 3.12) montre les différents flux pour réaliser ces expérimentation.

Il est a préciser que nous avons utilisé la méthode de validation 30%, 70% (méthode de validation standard dans KNIME). Aussi nous avons choisi l'option stratified sampling qui préserve approximativement la distribution des avis par rapport au data set d'origine.Par conséquent, nous avons obtenu les dataset pour le test comme suit (Table 3.2).

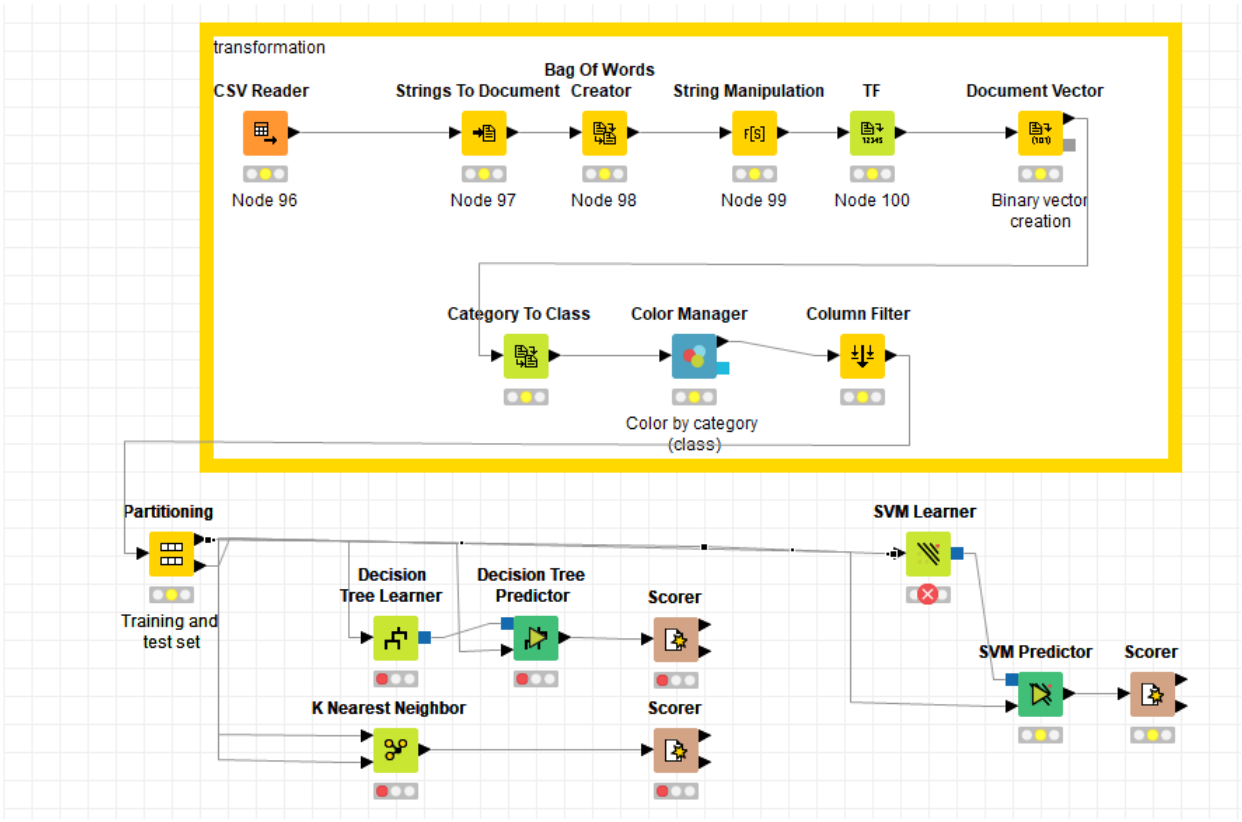

FIGURE 3.10 – Flux de classification de texte sans prétraitement dans KNIME

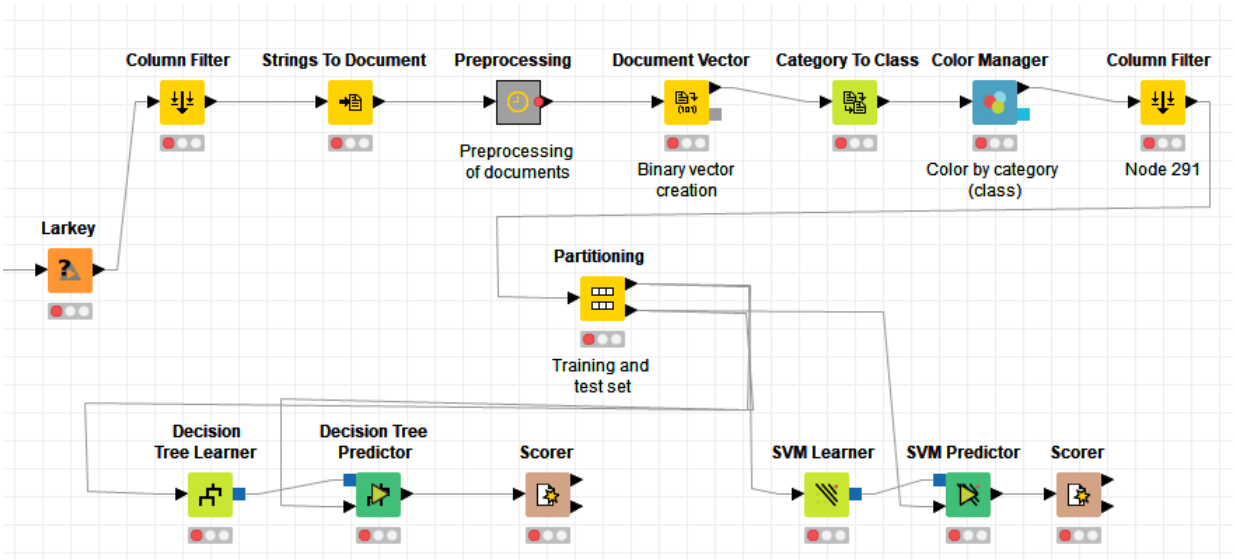

FIGURE 3.11 – Flux de classification de texte avec le racineur de Larkey dans KNIME

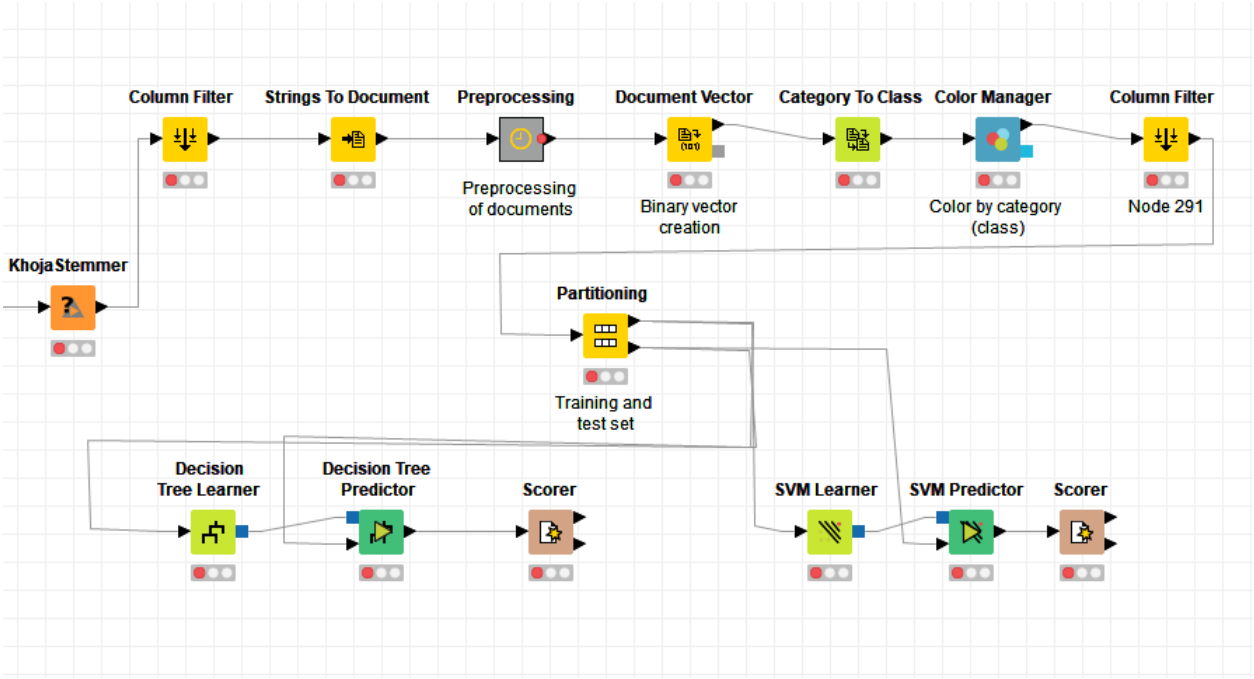

FIGURE 3.12 – Flux de classification de texte avec le racineur de Khoja dans KNIME

3.6.3 Metriques utilisés

Afin de mesurer les performances de méthode de classification ainsi que l'impact de prétraitement du texte arabe sur ces tâches, nous utilisons la matrice de confusion qui nous permet de calculer les différents métrique (exactitude, précision, rappel).

- **Matrice de confusion :**
  Table 3.3 avec :
  - VP (Vrais positifs) : Les documents appartenant à la classe A que le classifieur a attribué à la classe A.
  - FP (Faux positifs) : Les documents appartenant à la classe B que le classifieur a attribué à la classe A.
  - VN (Vrais négatifs) : Les documents appartenant à la classe B que le classifieur a attribué à la classe B.
  - FN (Faux négatifs) : Les documents appartenant à la classe A que le classifieur a attribué à la classe B.
- **La précision :** C'est le rapport entre le nombre de documents correctement classés dans la classe (C) sur le nombre de document auxquels la classe (C) est assignée.

$$\textbf{Précision (C)} = \frac{\textbf{VP}}{\textbf{VP+FP}}$$

Data set de test	Nbr total cmts	classe	Nbr cmt class	taux
Sans prétraitement	1282	إيجابي	931	73%
		سلبي	259	20%
		محايد	92	7%
Taritement avec khoja	1109	إيجابي	813	73%
		سلبي	218	20%
		محايد	78	7%
Taritement avec Larkey	1124	إيجابي	821	73%
		سلبي	224	20%
		محايد	79	7%

TABLE 3.2 – Distribution approximative de data set du test

		Classes Prédites	
		Classe A	Classe B
Classes Réelles	Classe A	VP	FP
	Classe B	FN	VN

TABLE 3.3 – Matrice de confusion

- **Le rappel :** C'est le rapport entre le nombre de documents correctement classés dans la classe (C) sur le nombre de documents appartenant à la classe (C).

$$\mathbf{Rappel\,(C)} = \frac{\mathbf{VP}}{\mathbf{VP+FN}}$$

- **F-mesure (F1) :** C'est un indicateur qui combine le rappel et la précision elle est donnée par la formule suivante :

$$\mathbf{F1} = \frac{\mathbf{2\times Précision\times Rappel}}{\mathbf{Précision+Rappel}} = \frac{\mathbf{2»VP}}{\mathbf{2\times VP+FP+FN}}$$

- **l'exactitude (accuracy) :** C'est la proportion de décisions correctes prises par le classificateur.

$$\mathbf{Exactitude} = \frac{\mathbf{VP+VN}}{\mathbf{VP+VN+FP+FN}}$$

- **Temp d'exécution**.

Row ID	إيجابي	سلبي	محايد
إيجابي	798	91	42
سلبي	115	126	18
محايد	66	21	5

FIGURE 3.13 – Matrice de confusion de classifieur AD sans prétraitement

Row ID	إيجابي	سلبي	محايد
إيجابي	764	47	2
سلبي	120	93	5
محايد	69	8	1

FIGURE 3.14 – Matrice de confusion de classifieur SVM avec le racineur de Khoja.

3.6.4 Résultats et discussions

A l'issue de la phase de test, nous avons obtenu les résultats suivants organisés sous forme de matrices de confusion (Figure 3.13 - Figure 3.17)

A partir de ces matrices de confusion, nous pouvons construire les tables de résultats impliquant les mesures de précision, éxactitude, rappel et F1 tels que illustré dans les tables (3.4- 3.8).

Dans la plate-forme KNIME, les résultats précités peuvent être obtenu en utilisant les nœuds **Scorer** pour les métriques (Matrice de confusion, précision, rappel..) et le nœud **Timer Info** pour le temps d'exécution.
Les résultats obtenus révèlent l'impact visible du prétraitement du texte arabe sur le temps d'exécution (Table 3.9) et les différents métriques.

A titre l'illustration, nous pouvons constater qu'il n'était pas possible d'extraire le classifieur SVM (dans notre environnement) sur le texte brut, alors que nous avons obtenu des résultats satisfaisants dans le cas de classification suite à un prétraitement du texte. Autrement dit les performances du classification (précision, rappel, le temps d'exécution) s'améliorons dans le cas de la classification après prétraitement .

Row ID	إيجابي	سلبي	محايد
إيجابي	705	88	20
سلبي	97	112	9
محايد	61	15	2

FIGURE 3.15 – Matrice de confusion de classifieur AD avec le racineur de Khoja.

Row ID	إيجابي	سلبي	محايد
إيجابي	790	29	2
سلبي	123	101	0
محايد	61	14	4

FIGURE 3.16 – Matrice de confusion de classifieur SVM avec le racineur de Larkey.

Row ID	إيجابي	سلبي	محايد
إيجابي	739	72	10
سلبي	97	126	1
محايد	64	12	3

FIGURE 3.17 – Matrice de confusion de classifieur AD avec le racineur de Larkey.

3.7 Conclusion

Dans ce chapitre, nous avons fourni les étapes nécessaires pour développer un nœud dans KNIME. Ensuite, nous avons expliqué les nœuds que nous avons développés pour le flux de prétraitement du texte arabe.

Compte tenu des résultats obtenus dans la section 3.6 de l'expérimentation, nous avons prouvé l'efficacité de notre flux de prétraitement de texte arabe dans une tache de classification de texte.

	Précision	Rappel	F1	exactitude
إيجابي	0,815	0,857	0,836	
سلبي	0,529	0,486	0,507	
محايد	0,077	0,054	0,064	
overall				0,725

TABLE 3.4 – Résultat d'évaluation de classifieur AD sans prétraitement.

	Précision	Rappel	F1	exactitude
إيجابي	0,817	0,867	0,841	
سلبي	0,521	0,514	0,517	
محايد	0,065	0,026	0,037	
overall				0,739

TABLE 3.5 – Résultat d'évaluation de classifieur AD avec prétraitement de racineur de Khoja

	Précision	Rappel	F1	exactitude
إيجابي	0,802	0,94	0,865	
سلبي	0,628	0,427	0,505	
محايد	0,125	0,013	0,023	
overall				0,774

TABLE 3.6 – Résultat d'évaluation de classifieur SVM avec prétraitement de racineur de Khoja

	Précision	Rappel	F1	exactitude
إيجابي	0,811	0,962	0,88	
سلبي	0,701	0,451	0,549	
محايد	0,667	0,051	0,094	
overall				0,796

TABLE 3.7 – Résultat d'évaluation de classifieur SVM avec prétraitement de racineur de Larkey

	Précision	Rappel	F1	exactitude
إيجابي	0,821	0,9	0,859	
سلبي	0,6	0,562	0,581	
محايد	0,214	0,038	0,065	
overall				0,772

TABLE 3.8 – Résultat d'évaluation de classifieur AD avec prétraitement de racineur de Larkey

		Texte brute	Traité par Khoja	Traité par Larkey
temps d'exécution (s)	SVM	+3369,904	29,093	28,283
	AD	555,948	7,667	3,663

TABLE 3.9 – Comparaison entre SVM et AD en termes de temps d'exécution

Conclusion générale

Le but de notre mémoire est de développer un outil pour le prétraitement de texte arabe, car la langue arabe marque un manque important d'outils et d'applications de traitement du texte pour l'extraction des connaissances. Bien que l'Arabe soit est la quatrième langue parlée avec une présence très visible dans l'internet.
Nous avons commencé par l'introduction de domaine de l'intelligence artificiel, du TALN ainsi que le processus de prétraitement du texte arabe. Afin d'atteindre notre objectif, nous avons développé trois nœuds essentiels dans le prétraitement de texte arabe sur la plateforme KNIME. Le premier nœud s'occupe le filtrage des mots vides, de la ponctuation et tous les caractères non arabes. Les deux autres nœuds s'occupent, respectivement de l'implémentation du racineur de Khojaqui à base de l'analyse morphologique et le racineur de Larkeyqui suit l'approche de racinisation légère.

Nous avons expérimenté notre flux de prétraitement du texte arabe dans une tâche de classification de texte (analyse de sentiments), dont les résultats témoignent que notre flux de prétraitement améliore les performances de la classification en termes de temps d'exécution et d'exactitude.

Nous envisageons intégrer officiellement notre chaine de prétraitement dans la plateforme KNIME ou même dans d'autre plateforme largement utilisée dans le domaine de l'apprentissage automatique.

Bibliographie

Charu C Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer, 2012a.

Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012b.

Sabah Al-Fedaghi and Fawaz Al-Anzi. A new algorithm to generate arabic root-pattern forms. In *proceedings of the 11th national Computer Conference and Exhibition*, pages 391–400, 1989.

Belal Al-Omari, Asma et Abuata. Arabic light stemmer (ars). *Journal of Engineering Science et Technology*, 9(6) :702–717, 2014.

J Allen. Natural language understanding, the benjamim/cummings publish company, 1995.

Mohammed Attia and Harold Somers. *Handling Arabic morphological and syntactic ambiguity within the LFG framework with a view to machine translation*, volume 279. University of Manchester Manchester, 2008.

Arafat Awajan. Arabic text preprocessing for the natural language processing applications. *Arab Gulf Journal of Scientific Research*, 25(4) :179–189, 2007.

Aqil M Azmi and Suha Al-Thanyyan. A text summarizer for arabic. *Computer Speech & Language*, 26(4) :260–273, 2012.

MH Bakalla. Arabic language through its language and literature. *London : Kegan Paul*, 2002.

Richard Bellman. *An introduction to artificial intelligence : Can computers think?* Thomson Course Technology, 1978.

Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime-the konstanz information miner : version 2.0 and beyond. *AcM SIGKDD explorations Newsletter*, 11(1) :26–31, 2009.

Onam Bharti and Mrs Monika Malhotra. Sentiment analysis on twitter data. 2016.

Eugene Charniak and Drew McDermott. Introduction to ai. *Reading (Mass.) : Addison*, 1985.

Fredric C Chen, Aitao et Gey. Building an arabic stemmer for information retrieval. In *TREC*, volume 2002, pages 631–639, 2002.

Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1) :51–89, 2003.

Hady ElSahar and Samhaa R El-Beltagy. Building large arabic multi-domain resources for sentiment analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 23–34. Springer, 2015.

Suran Goonatilake and Sukhdev Khebbal. *Intelligent hybrid systems*. John Wiley & Sons, Inc., 1994.

Lisa Guernsey. Digging for nuggets of wisdom. *The New York Times*, 16, 2003.

CHENINI Hafsa and DJAFFER Oumima. Noyaux de séquences pour la classification de textes. *Université de ghardaia*, 2017.

MOHAMED HASSANI. vers un modèle formel applicatif pour une nouvelle ingénierie de la langue et de l' information. *Université du Québec à Trois-Rivières*, 2016.

Marcus Hassler and Gunther Fliedl. Text preparation through extended tokenization. *WIT Transactions on Information and Communication Technologies*, 37, 2006.

John Haugeland. Artificial intelligence : The very idea. 1985. *Cited on*, page 26, 1985.

IDC. Growth of unstructured data 2015-2017. *International Data Corporation.*, 2017.

Anjali Ganesh et others Jivani. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, 2(6) :1930–1938, 2011.

M Tim Jones. *Artificial intelligence : a systems approach*. Laxmi Publications, Ltd., 2008.

Riyad et Ababneh Mohamad et Al-Nobani Alaa Kanaan, Ghassan et Al-Shalabi. Building an effective rule-based light stemmer for arabic language to inprove search effectiveness. In *Innovations in Information Technology, 2008. IIT 2008. International Conference on*, pages 312–316. IEEE, 2008.

Shereen Khoja and Roger Garside. Stemming arabic text. *Lancaster, UK, Computing Department, Lancaster University*, 1999.

Shereen Khoja, Roger Garside, and Gerry Knowles. A tagset for the morphosyntactic tagging of arabic. In *Corpus Linguistics 2001 conference, Lancaster*, 2001.

Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

Ray Kurzweil, Robert Richter, Ray Kurzweil, and Martin L Schneider. *The age of intelligent machines*, volume 579. MIT press Cambridge, 1990.

James et Connell Margaret E et Bolivar Alvaro et Wade Courtney Larkey, Leah S et Allan. Umass at trec 2002 : Cross language et novelty tracks. Technical report, MASSACHUSETTS UNIV AMHERST CENTER FOR INTELLIGENT INFORMATION RETRIEVAL, 2002a.

Lisa et Connell Margaret E Larkey, Leah S et Ballesteros. Improving stemming for arabic information retrieval : light stemming et co-occurrence analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research et development in information retrieval*, pages 275–282. ACM, 2002b.

Elizabeth D Liddy. Natural language processing. 2001.

Julie Beth Lovins. Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, 11(1-2) :22–31, 1968.

J. Mace. *Arabic Verbs and Essential Grammar*. Teach yourself books. McGraw-Hill/Contemporary, 1999. URL https://books.google.dz/books?id=Hq2CAAAAIAAJ.

Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

MSE. branches of artificial intelligence. *Management Science & Engineering from Stanford University*, 2017.

Michael Negnevitsky. *Artificial intelligence : a guide to intelligent systems*. Pearson Education, 2005.

Attia NEHAR. Noyaux rationnels pour la classification des données non structurées : Documents web en arabe. *Université Amar Telidji - Laghouat*, 2017.

Djelloul et Cherroun Hadda Nehar, Attia et Ziadi. Rational kernels for arabic stemming and text classification. *arXiv preprint arXiv :1502.07504*, 2015.

Nils J Nilsson. *Artificial intelligence : a new synthesis*. Elsevier, 1998.

Mohammed A Otair. Comparative analysis of arabic stemming algorithms. *International Journal of Managing Information Technology*, 5(2) :1, 2013.

Seymour Papert and ML Minsky. Perceptrons : an introduction to computational geometry. *Expanded Edition*, 1969.

H Patrick. Winston. artificial intelligence, 1992.

David Poole, Alan Mackworth, and Randy Goebel. Computational intelligence : a logical approach. 1998.

Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3) :130–137, 1980.

Elaine Rich and Kevin Knight. Artificial intelligence. *McGraw-Hill, New*, 1991.

Scott et Yang Yiming Rogati, Monica et McCarley. Unsupervised learning of arabic stemming using a parallel corpus. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 391–398. Association for Computational Linguistics, 2003.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088) :533, 1986.

Stuart J Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Malaysia ; Pearson Education Limited,, 2016.

Motaz K Saad and Wesam Ashour. Arabic morphological tools for text mining. *Corpora*, 18 :19, 2010.

AW Sadek. Artificial intelligence in transportation : information for application. *Transportation Research Board Circular (E-C113), TRB, National Research Council, Washington, DC Available online at :< http ://onlinepubs. trb. org/onlinepubs/circulars/ec113. pdf*, 2007.

Majdi Sawalha. Open-source resources and standards for arabic word structure analysis. *Leeds : University of Leeds PhD*, 2011.

Majdi Sawalha and Eric Atwell. Comparative evaluation of arabic language morphological analysers and stemmers. *Coling 2008 : Companion volume : Posters*, pages 107–110, 2008.

Jan et Snášel Václav Soori, Hussein et Platoš. Simple stemming rules for arabic language. In *Proceedings of the Third International Conference on Intelligent Human Computer Interaction (IHCI 2011), Prague, Czech Republic, August, 2011*, pages 99–108. Springer, 2013.

Mostafa M Syiam, Mohamed F Tolba, ZT Fayed, Mohamed S Abdel-Wahab, Said A Ghoniemy, and Mena Badieh Habib. An intelligent system for arabic text categorization. *International Journal of Intelligent Computing and Information Sciences*, 6(1) :1–19, 2006.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

G Vinodhini and RM Chandrasekaran. Sentiment analysis and opinion mining : a survey. *International Journal*, 2(6) :282–292, 2012.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering : A set of prerequisite toy tasks. *arXiv preprint arXiv :1502.05698*, 2015.

Roman V Yampolskiy. Ai-complete, ai-hard, or ai-easy-classification of problems in ai. In *MAICS*, pages 94–101, 2012.

Taha Zerrouki. *Arabic Stop words*, 2010. http://sourceforge.net/projects/arabicstopwords/.