Outils génériques pour l'étiquetage morphosyntaxique de la langue arabe : segmentation et corpus d'entraînement

Dhaou Ghoul

UNIVERSITÉ Stendhal GRENOBLE 3

Outils génériques pour l'étiquetage morphosyntaxique de la langue arabe : segmentation et corpus d'entraînement

Nom : GHOUL

Prénom : DHAOU

UFR Sciences du Langage

Mémoire de master 2 recherche - 30 crédits – Sciences du langage

Spécialité ou Parcours : Modélisation et traitements automatique en industries de la langue : parole, écrit, apprentissage orientation Recherche

Sous la direction d'Olivier Kraif

Composition du jury :

Georges Antoniadis
Lynne Franjié
Olivier Kraif

Année universitaire 2010-2011

Remerciements

En préambule à ce mémoire, je souhaiterais adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Je tiens à remercier sincèrement Monsieur Olivier Kraif, qui, en tant qu'encadreur de mémoire, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Mes remerciements s'adressent également à Monsieur Georges Antoniadis : Responsable du master, de me faire découvrir le domaine du TAL qui m'a bien intéressé et de l'aide qu'il m'a attribué tout au long des années des études en IDL.

Je présente mes remerciements anticipés à Mme Lynne Franjié qui m'a fait l'honneur d'accepter la tâche d'être rapporteuse.

Je tiens à remercier Atef ben Youssef, pour ses remarques pertinentes lors de nos discussions, son soutien et son amitié.

J'exprime ma gratitude à tous les consultants et internautes rencontrés lors des recherches effectuées (particulièrement Monsieur El Haj) et qui ont accepté de répondre à mes questions avec gentillesse.

Je garde une place toute particulière à mes parents, mes frères et mes sœurs qui sont toujours à mes cotés.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et amis, qui m'ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire. Merci à tous et à toutes.

GHOUL DHAOU

Résumé :

L'objectif de ce travail est la réalisation d'un nouvel outil pour l'étiquetage morphosyntaxique de la langue arabe.
Après avoir étudié les spécificités de la langue arabe, et confronté celles-ci avec les différentes approches de l'étiquetage, nous avons mis en œuvre un système basé sur l'étiqueteur stochastique TreeTagger, réputé pour son efficacité et la généricité de son architecture.

Pour ce faire, nous avons commencé par la constitution de notre corpus de travail. Celui-ci nous a d'abord servi à réaliser l'étape de segmentation lexicale. Dans un second temps, ce corpus a permis d'effectuer l'entrainement de TreeTagger, grâce à un premier étiquetage réalisé avec l'étiqueteur ASVM 1.0, suivi d'une phase de correction manuelle. Nous détaillons ainsi les prétraitements requis, et les différentes étapes de la phase d'apprentissage avec cet outil.

Nous terminons par une évaluation sommaire des résultats, à la fois qualitative et quantitative. Cette évaluation, bien que réalisée sur un corpus de test de taille modeste, montre que nos premiers résultats sont encourageants.

Mots clés :

TALN, langue arabe, corpus d'apprentissage, étiquetage morphosyntaxique, segmentation de l'arabe, arbre de décision, lexique, jeux d'étiquette, TreeTagger, ASVM 1.0.

ABSTRACT:

The main goal of this study is the implementation of a new tool for the part-of-speech tagging of Arabic.
After studying the different approaches and problems of arabic tagging, we implemented a tagging system based on TreeTagger, a generic stochastic tagging tool, very popular for its efficiency.

First of all, we began by gathering a working corpus, large enough to ensure a general linguistic coverage. This corpus has been used to implement the tokenization process, as well as to train Treetagger. We first present our method of tokenization, then we describe all the steps of the preprocessing and training process, using ASVM 1.0 to yield a raw POS tagging that was subsequently manually corrected.

Finally, we implemented a straightfoward evaluation of the outputs, both in a quantitative and qualitative way, on a small test corpus. Though restricted, this evaluation showed really encouraging results.

Key words:

NLP, Arabic language, training corpus, POS tagging, tokenization, decision tree, lexicon, tagsets, TreeTagger, ASVM 1.0.

Sommaire

Table de matière

Liste des tableaux :

Liste des Figures :

Introduction générale

L'Internet fait maintenant partie de notre vie quotidienne et il contient une quantité phénoménale d'informations qui sont très utiles. Aujourd'hui la langue arabe est parmi les langues les plus utilisées sur le Web. On peut y trouver des nombreux ouvrages que les auteurs ont décidé de rendre publics.

Par ailleurs, il existe de nombreux logiciels traitant la langue naturelle qui facilitent la recherche et la consultation des documents électroniques, comme par exemple les moteurs de recherche sur Internet, les résumeurs et traducteurs automatiques. Bien que ne donnant pas toujours des résultats totalement satisfaisants, certains aboutissent à des résultats intéressants et exploitables par le grand public, comme les logiciels de traduction automatiques. Pour avoir un système de traduction d'une langue à une autre, mais également pour l'indexation ou l'extraction d'information, on peut avoir besoin d'un système d'étiquetage robuste et performant. C'est pourquoi, beaucoup de recherches ont été menées sur cette tâche**.** Ces différentes recherches ont donné lieu à des propositions d'approches et à des algorithmes, et ont parfois débouché sur des applications.
D'un point de vue général, pour mettre en œuvre des outils du TAL (Traitement automatique de la langue) en arabe, les chercheurs peuvent avoir besoin :

- Des modules de base pour la segmentation en phrases et en mots, l'analyse morphologique, syntaxique voire sémantique ;
- Des ressources langagières (dictionnaires, corpus, bases de données lexicales, …) ;
- Des ressources et modules de comparaisons pour l'évaluation ;
- D'utilitaires de traitement de la langue (outils de recherche de texte, outils statistiques sur les textes et corpus annotés, etc.) ;

Parmi les « modules de base », l'étiquetage morphosyntaxique constitue une étape essentielle pour réaliser la plupart des applications en traitement automatique de la langue car il permet d'identifier la catégorie grammaticale à laquelle appartiennent les mots du texte. Ainsi, les étiqueteurs constituent un module essentiel dans des applications de grand public telles que la correction grammaticale automatique, la génération automatique des

résumés et le repérage d'information. Ils sont aussi très utiles dans le traitement de la parole pour réaliser des systèmes de synthèse ou de reconnaissance vocale. En général, l'étiquetage morphosyntaxique est une étape préalable dont il est difficile de faire l'économie dans la plupart des applications du TAL.

Dans le cadre de notre master, nous avons décidé de nous concentrer sur ce problème spécifique pour la langue arabe.

Dans ce domaine, de nombreux travaux ont été réalisés. Les plus connus sont basés sur des modèles de Markov, des systèmes symboliques à base de règles ou encore des réseaux de neurones. Ces systèmes parviennent à produire seulement 3-4% d'erreurs pour l'anglais (Brill, 1993) mais ce n'est pas le cas pour l'arabe. Ils peuvent atteindre plus de 10% d'erreurs (EL Jihad & Yousfi, 2005) à cause de l'ambiguïté graphique des mots. Malgré les différentes recherches effectuées sur la langue arabe, il existe peu d'outils gratuits et génériques pour cette langue qui pourraient être utile à la communauté des chercheurs. A l'heure actuelle, TreeTagger constitue un des outils les plus populaires et les plus répandus grâce à sa rapidité, à son architecture indépendante des langues, et à la qualité des résultats obtenus. C'est pourquoi, nous avons cherché à élaborer un fichier de paramètres de TreeTagger pour l'arabe.

Cette recherche à été menée au laboratoire LIDILEM de Stendhal sous la direction d'Olivier Kraif. Notre travail porte sur la construction des données et des prétraitements en entrée de cet outil, afin d'appliquer les deux modules principaux, le programme d'entraînement et l'étiqueteur proprement dit. Pour la construction du fichier de paramètres, nous avons commencé par construire un vaste corpus d'apprentissage indispensable pour la phase d'entraînement.

Ce mémoire est composé de deux parties et contient cinq chapitres distincts :

Dans la première partie, qui est divisée en deux chapitres, nous détaillons d'abord les spécificités de la langue arabe, puis nous effectuons une présentation générale des méthodes d'étiquetage, afin d'identifier les problèmes posés par la langue arabe pour ces différents approches. .

Dans la deuxième partie qui s'intitule « développement d'un nouvel outil d'étiquetage pour l'arabe », nous nous consacrons au développement de notre propre solution. D'abord nous détaillons une méthode de segmentation lexicale, car ce problème non trivial pour l'arabe constitue une étape préliminaire indispensable à la mise en œuvre de TreeTagger. Le chapitre suivant est consacré à la construction des ressources nécessaires à l'entraînement

de TreeTagger sur l'arabe. Le dernier chapitre, enfin donne une évaluation de nos premiers résultats.

Partie1 :

Etat de l'art

Chapitre1 : Langue arabe et TALN

1. Introduction

Malgré des nombreuses recherches, la langue arabe est considérée comme une langue difficile à maitriser dans le domaine du traitement automatique de la langue (Aljlayl & Frieder, 2002) à cause de sa richesse morphologique. L'arabe existe et se développe à partir du 7ème siècle grâce à diffusion du Coran[1] qui est considéré comme la base de cette langue. Avec la diffusion de la langue arabe sur le Web et la disponibilité des moyens de manipulation des textes arabes, les travaux de recherche ont abordé des problématiques variées comme la morphologie, la traduction automatique, l'indexation des documents, etc.

Au cours de ce chapitre nous présenterons les particularités de la langue arabe ainsi que certaines de ses propriétés morphologiques et syntaxiques.

2. Particularités de la langue arabe

Au contraire de nombreuses autres langues, l'arabe s'écrit et se lit de droite à gauche. Une autre originalité concerne l'utilisation facultative des voyelles. Les voyelles sont ajoutées au-dessus ou au-dessous des lettres, sous la forme des signes diacritiques (voir tableau 4 p11). Elles sont utiles à la lecture et à la compréhension correcte d'un texte, car elles permettent de distinguer des mots ayant la même représentation graphique. Elles sont utiles, notamment, pour effectuer la correcte interprétation grammaticale d'un mot indépendamment de sa position dans la phrase.

Le tableau suivant donne un exemple pour les mots s'écrivant كَتَبَ (ktb) sous la forme non-voyellée.

.

[1] Nom du livre sacré de l'islam. Les musulmans le considèrent comme la parole de Dieu livré au prophète Mohammed par l'intermédiaire de Gabriel. C'est la base de la langue arabe.

Mot sans voyelles	Interprétation 1	Interprétation 2	Interprétation 3
كَتب ktb	كَتَبَ Kataba « il a écrit »	كُتِبَ kutiba « il a été écrit »	كُتُبٌ kutubN « des livres »

Tableau 1 : Interprétation du mot « كتب »

En général les voyelles ne sont utilisées que pour textes sacrés et didactiques (comme les textes du Coran). En effet, la presse, la littérature et la plupart des textes écrits contemporains ne contiennent pas de voyelles. De plus, même pour des textes non voyellés, il existe des variations d'usage au niveau des diacritiques. Par exemple, **أ** ou **إ** , qui correspondent à des réalisations différents de la lettre **ا(alif),** et qui sont en principe discriminées (et translittérées par les caractères | ou < en translitération Buckwalter), sont souvent écrites **ا** sans les diacritiques. Il est de même pour les lettres **ي(Y)** et **ة(p)** qui s'écrivent parfois **ى(y)** et **ه(h)** (Xu, Fraser & Weischedel, 2002), ce qui est une grande source l'ambiguïté dans l'interprétation des mots**.**

Les voyelles jouent donc un rôle analogue à celui des accents en français, comme par exemple pour le mot « peche » qui peut être interprété comme pêche, pèche ou péché. Mais, ces ambiguïtés sont démultipliées en arabe, car chaque lettre de chaque mot devrait posséder sa voyelle, ce qui augmente les combinaisons possibles.

Signalons une dernière difficulté concernant le système d'écriture, les lettres arabes changent de forme de présentation selon leur position, au début, au milieu ou à la fin du mot.

Le tableau suivant montre par exemple les variations de la lettre ع (ayn). Toutes les lettres sont liées entre elles sauf (ا و ر ز د ذ) qui ne se joignent pas à gauche.

Lettre	A la fin	Au milieu	Au début
ع	ـع	ـعـ	عـ

Tableau 2 : La variation de lettre « ع »

Mais ces variations n'affectent pas le traitement automatique, car elles ne touchent que les glyphes (représentations graphiques définies lors du rendu) et pas les caractères eux même dans le codage informatique des textes.

3. L'alphabet arabe

L'alphabet de la langue arabe compte 28 consonnes appelées « Huruf al_Hija » (voir annexe 1, p 72) et se compose de deux familles contenant le même nombre de consonnes :

- Familles Solaires : contient 14 consonnes.
- Familles Lunaires : contient 14 consonnes.

Le tableau suivant représente la classification des consonnes arabe :

Famille Solaires	Famille Lunaires
ت ث د ذ ر ز س ش ص ض ط ظ ل ن	أ ب ج ح خ ع غ ف ق آ ه م و ي

Tableau 3 : Classification des consonnes arabes

Par ailleurs, l'alphabet arabe compte 6 voyelles qui sont aussi divisées en deux groupes :

- Voyelles courtes : 3 voyelles.
- Voyelles longues : 3 voyelles.

Le tableau suivant représente ces deux groupes :

Voyelles courtes	Voyelles longues
ُ / ـَ / ـِ /	ا ي و

Tableau 4 : Classification des voyelles arabes

4. Morphologie de l'arabe

Dans le lexique de la langue arabe, on distingue trois catégories principales de mots : les verbes, les noms et les particules, qui se subdivisent elles-mêmes en différentes sous catégories : préposition, conjonction, pronom, article, interjection et adverbe. Les verbes et les noms sont le plus souvent dérivés d'une racine à trois consonnes radicales (Baloul,Alissali, Baudry & de Mareuil, 2002). Une famille de mots se référant à une même représentation sémantique, peut être générée à partir d'une seule racine, à l'aide de différents schèmes. Ce phénomène est caractéristique de la morphologie arabe. On dit donc que l'arabe est une langue à racines réelles : à partir de ces racines, on déduit le lexique arabe selon des schèmes qui se traduisent par des adjonctions et des manipulations de la racine (Zouaghi, Zrigui & Ben Ahmed, 2005). Le tableau suivant donne quelques exemples des schèmes appliqués au verbe **عَمل** (Eml) :

Schème	Mot	Traduction en français
فَعل	عَمل (Eaml)	travail
فَاعل	عَامل (Eaml)	ouvrier
فَعَلَ	عَمَلَ (Eamala)	a travaillé
مَفعل	مَعمل (maEml)	atelier
فُعلَ	عُمِلَ (Eumila)	a été travaillé
مَفعول	مَعمول (maEmwl)	applicable

Tableau 5 : Exemples des schèmes appliqués au mot عَمل « travailler »

La plupart des verbes arabes sont formés d'une racine composée de trois consonnes, les dérivés et les formes fléchies du verbe étant obtenues par l'application de certains

schèmes. Au total, l'arabe utilise environ 150 schèmes dont certains plus complexes, tel le redoublement d'une consonne ou l'allongement d'une consonne de la racine (voyelles longues). Une autre propriété syntaxique est le caractère flexionnel des éléments : les déclinaisons, qui s'attachent au début ou à la fin de radical permettent par exemple de distinguer le mode du verbe ou la nominalisation.

4.1. Structure d'un mot

Les mots peuvent avoir une structure composée, résultat d'une agglutination de morphèmes lexicaux et grammaticaux. En arabe un mot peut représenter toute une proposition. La représentation suivante schématise une structure possible de mot complexe. Notons bien que la lecture se fait de droite à gauche.

Enclitique	Suffixe	Corps schématique	Préfixe	Proclitique

- ✓ Les proclitiques sont des prépositions ou des conjonctions.
- ✓ Les préfixes et suffixes expriment des traits grammaticaux, tels que les fonctions de noms, le mode du verbe, le nombre, le genre, la personne…..
- ✓ Les enclitiques sont des pronoms personnels.
- ✓ Le corps schématique représente la base de mot.

Exemple :

أتتفكرونا >tatafakarwunanA (translittération Buckwalter)

Ce mot en arabe représente en français la phrase suivante : « Est-ce que vous vous souvenez de nous ? »

La segmentation correcte de ce mot se fait sous la forme suivante :

نَا nA	ونَـ wna	تَفَكّرُ tafak~ru	تَـ ta	أ >

Tableau 6 : segmentation correcte du mot : « أتتفكرونا »

- Proclitique : أ conjonction d'interrogation.
- Préfixe : تَـ préfixe verbal exprimant l'aspect inaccompli.

- Corps schématique : تَفَكّرُ dérivé de la racine **/f k r/** (ف ك ر) selon le schème تَفعل
- Suffixe : ونَ suffixe verbal exprimant le pluriel.
- Enclitique : نَا pronom suffixe.

Cet exemple montre bien la richesse morphologique de la langue arabe. Pour identifier les différentes formes soudées par ces phénomènes d'agglutination, et envisager un traitement automatique, il va donc falloir mettre en œuvre une phase spécifique de segmentation.

4.2. Catégories d'un mot

La langue arabe comporte trois catégories de mots :

- ✓ Le verbe : entité qui exprime un sens dépendant du temps, c'est un élément fondamental auquel se rattachent directement ou indirectement les divers mots qui constituent l'ensemble.
- ✓ Le nom : l'élément désignant un être ou un objet qui exprime un sens indépendant du temps.
- ✓ Les particules : entités qui servent à situer les événements et les objets par rapport au temps.

4.2.1. Le verbe

En arabe, la majorité des mots dérivent d'un verbe de 3 consonnes qui représente une racine d'un groupe de mots. Comme en français, le mot en arabe se détermine à partir d'un radical en rajoutant des préfixes, des suffixes ou les deux en même temps.

Comme les autres langues, la conjugaison de verbe en arabe dépend de facteurs suivants :

- ➢ L'aspect: accompli (passé) ou inaccompli (présent).
- ➢ Le nombre du sujet : singulier, pluriel ou duel.
- ➢ Le genre du sujet : masculin ou féminin.
- ➢ La personne : première, deuxième ou troisième.
- ➢ La voix : active ou passive.

Voici un exemple qui illustre ces différents facteurs :

La combinaison de ces trois consonnes (ف + ت + ح) produit le verbe **فتح /fth/ (**ouvrir**).**

Comme on l'a dit précédemment, la conjugaison des verbes se fait en rajoutant des préfixes, des suffixes ou les deux.

La langue arabe possède trois temps :

- ✓ L'accompli : indique le passé et les verbes conjugués se distinguent par des suffixes. Pour notre exemple, avec le féminin pluriel, on obtient **فتحنَ**, **/fatahna/** « elles ont ouvert » ; pour le masculin pluriel, on obtient **فتحوا /fatahw/«** ils ont ouvert ».
- ✓ L'inaccompli présent : les verbes conjugués à ce temps se distinguent par les préfixes. Pour notre exemple, au masculin singulier on obtient **يَفتح /Yaftah/** « il ouvre » ; et pour le féminin singulier on obtient **تَفتح /taftah/«** elle ouvre ».
- ✓ L'inaccompli futur : la conjugaison d'un verbe au futur nécessite d'ajouter l'antéposition au début du verbe conjugué à l'inaccompli. En ajoutant l'antéposition à notre exemple **س** on obtient **سَيفتح** « il ouvrira », qui désigne le futur ; on peut également ajouter l'antéposition **سَوف** on obtient **سَوف يَفتح** « il va ouvrir ».

4.2.2. Les noms

En arabe les noms sont divisés en deux familles, ceux qui sont dérivés à partir d'une racine (verbale) et les autres comme les noms étrangers et certains noms fixes.

La première famille est composée des tous les noms qui sont dérivés à partir d'une racine verbale. La variabilité des noms obéit à plusieurs règles, en ajoutant des morphèmes spécifiques :

- ✓ Le féminin singulier : pour obtenir le nom féminin singulier, dans la majorité des cas on ajoute le lettre **ة** (exemple : **طِفل** « enfant » devient **طِفلة** « fille »).
- ✓ Le féminin pluriel externe : pour obtenir le nom féminin pluriel on ajoute les deux lettres **ات** (exemple **: كَاتب** devient **كَاتبات)**.
- ✓ Le masculin pluriel externe: on ajoute les deux lettres **ينْ** ou **ون** qui dépendent de la position du nom dans la phrase (avant ou après le verbe). Exemple : **معلّم** « enseignant » se transforme en **معلّمين**ou **معلّمون** « enseignants ».
- ✓ Le pluriel masculin, féminin et interne : c'est le cas le plus complexe en arabe, la construction de ces types des noms s'obtient en insérant des lettres au début, au milieu ou à la fin (exemple : **طِفل** « enfant » se transforme en **أطفال** « enfants » et **فصل** « saison » se transforme en **فُصول** « saisons »).

Comme l'a noté (Kiraz, 1996 p 2) : « Le phénomène du pluriel irrégulier en arabe pose un défi à la morphologie, non seulement à cause de sa nature non concaténative, mais aussi parce que son analyse dépend fortement de la structure comme les verbes irréguliers. »

Comme en français, les noms en arabe assument des fonctions diverses :

- ✓ Agent : celui qui fait l'action.
- ✓ Objet : celui qui subit l'action.
- ✓ Instrument : signifiant l'instrument de l'action.
- ✓ Lieu : qui désigne en général un endroit (exemple : **منزل** « maison »).
- ✓ Nom d'action : désigne l'action
- ✓ etc …

Notons que, la morphologie des noms arabes dépend de ces fonctions.

4.2.3. Les particules

En général, les particules sont les mots outils pour une langue donnée. En arabe, comme le notent (Kadri & Benyamina, 1992) : « Les particules sont classées selon leur fonction dans la phrase, on en distingue plusieurs types (introduction, explication, conséquence). Elles jouent un rôle important dans l'interprétation de la phrase. »

Les particules représentent en particulier les mots qui expriment des faits ou des choses par rapport au temps ou au lieu. Par exemple :

- Particules temporelles : **منذُ (pendant), قبل (avant)**, **بَعد (après),…**
- Particules spatiales : **حَيث (où)**

Les particules peuvent aussi exprimer des pronoms relatifs (la détermination, avec une valeur référentielle) : **الَّذي (ce), الَّذين (ceux), الَّتي (cette)**,…

Le problème, c'est que certaines particules peuvent également porter des préfixes et suffixes, ce qui complique la phase de segmentation pour les identifier.

4.2.3.1. Les préfixes

Les préfixes sont représentés par un morphème correspondant à une seule lettre en début de mot, qui indique la personne de la conjugaison des verbes au présent. Les préfixes ne se combinent pas entre eux. Le tableau suivant présente la liste des préfixes verbaux en arabe :

Dimension 1	Explication

أ	Indique la première personne au singulier (je)
نَـ	Indique la première personne au pluriel (nous)
تَـ	Indique la deuxième personne féminine, masculine, singulière et duelle
يَـ	Indique la troisième personne masculine au singulier, duel, pluriel, masculin et féminin pluriel.

Tableau 7 : Liste des préfixes arabe

4.2.3.2. Les suffixes

Les suffixes en arabe, sont essentiellement utilisés pour des terminaisons des conjugaisons verbales, ainsi que les marques du pluriel et du féminin pour les noms. Ils ne se combinent pas entre eux. La taille des suffixes varie entre 1 et 6 caractères. Le tableau suivant présente quelques suffixes an arabe avec leurs longueurs (L) :

L=1	L=2	L= 3	L=4	L= 5
ت	تَه	تَها	تَهما	ونَهما
و	تِي	تَهم	وهمَا	تَاهما
ن	تَك	تَهن	نهمَا	اتَهما
ا	هو	تَكم	اهمَا	ينَهما
ي	كو	تَكن	يهمَا	نَاهما
ة	نَه	تون	ونَهم	تَنهما
ه	نَك	تَنا	ونَهن	انَهما
ك	اَه	وهَا	ونَكم	تَموها
	اَك	وهَم	ونَكن	تَموهم
	نو	وهَن	ونَنو	تَمونا
	تَا	وكنَ	ونَنا	تَماها
	اَت	وكمَ	تَاها	تَماهم
		وون		تَماهن

Tableau 8 : Liste des suffixes les plus fréquents en arabe

4.2.3.3. Les proclitiques

Au contraire des préfixes et des suffixes, les proclitiques se combinent entre eux pour donner plus d'informations sur le mot arabe (traits sémantiques, coordination, détermination...). Comme le note (Abbes, 2004, p 244) : « Dans le cas des verbes, les proclitiques dépendent exclusivement de l'aspect verbal. Ils prennent donc tous les pronoms et par conséquent ils sont compatibles avec tous les préfixes pris par l'aspect. Dans le cas des noms et des déverbaux, le proclitique dépend du mode et du cas de déclinaison. »

Voici quelques exemples de proclitiques :

- La coordination par les coordonnants : **ف** « f » et **و** « w ».
- L'interrogation : **أ**
- La marque du futur : **س « s »**
- L'article : **الْ** « Al »
- Les prépositions par les lettres : **بـ** « bi » et **لـ**« li »

A l'écrit, il n'est pas toujours facile de faire la différence entre un proclitique et un caractère appartenant à la racine de certains mots. Par exemple le caractère **س** dans le mot سَرق « il a volé » est un caractère de la racine, par contre dans le mot سَيخرج « il va sortir » c'est un proclitique qui marque le futur.

4.2.3.4. Les enclitiques

Comme les proclitiques, les enclitiques se combinent entre eux pour donner une post-base composée. Ils s'attachent toujours à la fin du mot pour produire des pronoms suffixes qui s'attachent au verbe(CD), au nom et au préposition(complément du nom ou COI).

Exemples :

- ✓ **هُم** « hom » : **أخرجهم** « il l'ont sorti »(isolé ou suffixe).
- ✓ **ي « Y » : أخرجني** « il m'a sorti »
- ✓ **هُما** « homA| » : **أخرجهما**

4.2.3.5. Les pré-bases

Les pré-bases sont obtenues par combinaison entre le(s) proclitique(s) et le préfixe. La génération des pré-bases se fait d'une manière automatique. Le tableau suivant représente un exemple de groupe de pré-base :

Pré-base	Préfixe	Proclitique
أَتَ « Ata »	تَ « ta »	أ « A »
سَتَ «sta »	تَ « ta »	سَ « sa »
أَفَتَ « Afata »	تَ « ta »	أَفَ « Afa »
أَسَتَ « Asata »	تَ « ta »	أَسَ « Asa »
وسَتَ « wsata »	تَ « ta »	وسَ « wsa »
فَسَتَ « fsata »	تَ « ta »	فَسَ « fsa »
أَفَستَ « Afasata »	تَ « ta »	أَفسَ « Afasa »

Tableau 9 : Exemple de groupe de pré-base

4.2.3.6. Les post-bases

En arabe, les post-bases sont obtenues par combinaison entre le suffixe et le(s) enclitique(s). Les compatibilités dépendent des pronoms décrits par chacune des particules (Abbes, 2004) :

- ✓ Les suffixes de la première personne se combinent très souvent avec les enclitiques de la deuxième et la troisième personnes.
- ✓ Les suffixes de la deuxième personne se combinent très souvent avec les enclitiques de la première et la troisième personnes.
- ✓ Les suffixes de la troisième personne se combinent très souvent avec les enclitiques de la première, la deuxième personne et la troisième personne.

Enfin, il existe en arabe des suffixes qui jouent le rôle de caractère terminal du mot. En effet ces types des suffixes ne se combinent avec aucun enclitique(s). Le tableau suivant présente un exemple de groupe de post-bases :

Post-base	Enclitique	Suffixe
وكَ « waka »	ك « ka »	و « w »
وهُم « wahom »	هُم « hom »	و « w »
ونَنا « wananA »	نَا « nA »	ونَ « wna »
ونَني « wananY »	نِي « nY »	ونَ « wna »
أنَكم « Annakom »	كُم « kom »	أنَ « Anna »
أنَهم « Annahom »	هُم « hom »	أنَ « Anna »
تَموه « tamwh »	ه « h »	تَمو « tamw »

Tableau 10 : Exemple de groupe de post-bases

5. Conclusion

Ce bref tour d'horizon de la morphologie arabe montre la richesse et la complexité des phénomènes mis en jeu. Les problèmes d'ambiguïté liés au système d'écriture non-voyellé, ainsi que les difficultés de segmentation lexicale liées à l'agglutination, font que l'arabe reste une langue particulièrement difficile à traiter, sur le plan de l'écrit. Dans le chapitre suivant, nous allons détailler les différents problèmes posés par le traitement de la langue arabe.

Chapitre2 : L'étiquetage d'une Langue

1. Introduction

Reconnaitre la *nature* (i.e. la catégorie morphosyntaxique) d'un mot dans un contexte est une tâche non triviale du traitement automatique de la langue écrite. En effet rendre une machine capable de connaître la catégorie d'un mot exige de mettre en œuvre des méthodes sophistiquées, en particulier pour les mots ambigus, c'est-à-dire susceptibles d'appartenir à plusieurs catégories différents. Les systèmes automatiques dédiés à cette tâche sont appelés des *étiqueteurs* morphosyntaxiques (Part-of-speech tagger, en anglais).

Le processus de l'étiquetage morphosyntaxique se fonde généralement sur l'hypothèse que la catégorie d'un mot dépend de son contexte local, qui peut par exemple se réduire au mot ou aux deux qui le précèdent.
Dans ce chapitre, nous allons présenter différentes méthodes d'étiquetage morphosyntaxique, et effectuer un bref inventaire des étiqueteurs qui existent en particulier pour la langue arabe.

2. Méthodes d'étiquetage

Les différentes méthodes utilisent toutes les mêmes informations pour étiqueter un mot dans un texte : son contexte et sa morphologie. Ce qui diffère, c'est la façon de représenter ces éléments et de hiérarchiser ces informations.

Il existe deux grands types d'étiqueteurs :

- Les étiqueteurs symboliques sont ceux qui appliquent des règles qui leur ont été fournies par des experts humains. Dans ce type d'étiqueteur, il y a très peu d'automatisation ; c'est le concepteur qui manipule toutes les règles d'étiquetage et qui fournit au besoin une liste des morphèmes, La conception n'est pas automatisée mais l'étiqueteur, une fois ses règles élaborées, fournit un étiquetage automatique. La conception d'un tel étiqueteur est longue et coûteuse. De plus, les étiqueteurs ainsi conçus ne sont pas facilement portables, c'est-à-dire qu'ils ne sont efficaces que pour une langue donnée et

un domaine donnée (exemple : le droit, la médecine, etc).

- Les étiqueteurs avec apprentissage automatique (de l'anglais *Machine Learning*) sur lesquels nous allons nous concentrer dans la suite de cette étude. Parmi les étiqueteurs de ce type, il existe deux grandes familles : les étiqueteurs supervisés qui apprennent à partir de corpus pré-étiquetés, et les étiqueteurs non supervisés qui apprennent à partir de corpus bruts sans information additionnelle. Qu'ils soient supervisés ou non, les étiqueteurs avec apprentissage peuvent être regroupés en trois familles: système à base de règles, statistiques ou neuronal.

Les deux types d'approches ont chacune leurs avantages et leurs inconvénients. L'une et l'autre ne seront pas utilisées dans le même type de situation.

La figure suivante résume ces différents types d'étiqueteurs avec apprentissage automatique:

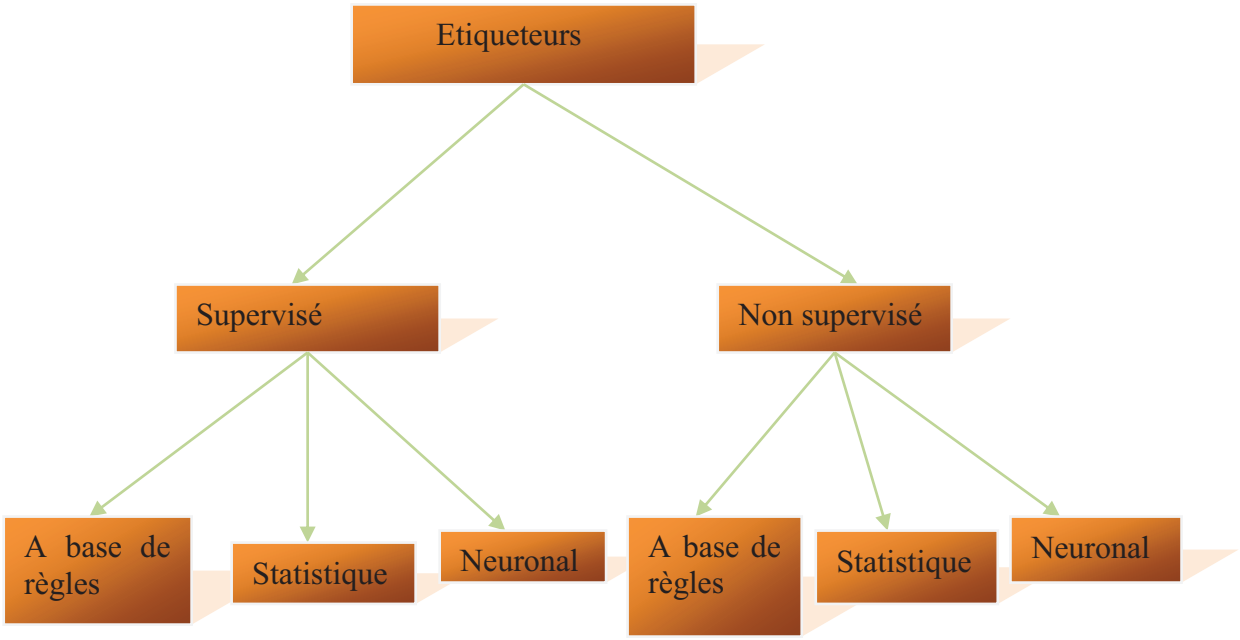

Figure 1 : Les différents méthodes d'étiquetage avec apprentissage automatique

2.1. Etiquetage non supervisé

Au contraire des étiqueteurs supervisés, les étiqueteurs non supervisés ne nécessitent pas de corpus préalablement étiqueté pour la phase d'entraînement. Ils utilisent une analyse distributionnelle afin de regrouper automatiquement les mots en groupes ou classes de mots, c'est-à-dire en catégories grammaticales.

Comme le note (Thibeault, 2004) « En général, le regroupement des mots en classes se fait en fonction de la similarité des contextes. Plus précisément, le fait que des mots soient interchangeables dans des contextes formellement similaires détermine s'ils font partie d'une même catégorie ou classe. »

Exemple : **une** *classe* **de mots**, **une** *catégorie* **de mots**.

2.2. Etiquetage supervisé

Les étiqueteurs supervisés sont entraînés sur des corpus préalablement étiquetés, ce qui permet de préparer toutes les données nécessaires pour l'étiquetage. Ces données sont créées à partir de dictionnaires permettant d'attribuer à chaque mot un ensemble de critères : catégorie ; lemme ; fréquence moyenne d'apparition du mot ; parfois des statistiques sur les étiquettes du mot en contexte ; et des règles pour faciliter l'analyse du mot par la suite. L'étiqueteur de Brill (Brill, 1993), l'étiqueteur APT de Khoja (Khoja, 2001) et l'étiqueteur POS Tagger (Diab, Hacioglu & Jurafsky, 2004) font partie de cette famille.

Les étiqueteurs supervisés ont tendance à donner de meilleurs résultats si on les utilise pour étiqueter le même type de texte que ceux sur lesquels ils ont été entraînés. Toutefois, l'étiqueteur de Brill échappe un peu à la norme. Il est en fait très portable du fait que le corpus nécessaire à l'entraînement n'a pas à être très gros (une centaine de pages). De plus, son approche a été conçue dans une perspective multilingue, ce qui fait que le logiciel peut être entraîné en quelques heures pour tout type de texte et pour toute langue écrite avec un alphabet. Cet étiqueteur applique des règles pour faire l'étiquetage, et a été déjà adapté sur la langue arabe (Haddad, Ben Ghezala & Ghenima, 2007). Par contre, TreeTagger est basé sur des méthodes statistiques qui sont plus efficaces que celles de Brill, mais il n'est pas disponible pour l'arabe. C'est pourquoi, nous avons fait le choix de Treetagger et pas de Brill.

Notons aussi qu'il n'y a pas si longtemps, les corpus étiquetés nécessaires à l'entraînement des étiqueteurs supervisés étaient plutôt rares, dispendieux et pas nécessairement

disponibles pour toutes les langues et tous les genres de texte. Mais, depuis quelques années, plusieurs corpus ont été créés et rendus disponibles (EAGLES ,1996 [2]), (Penn TreeBank Arabic, 2004[3]) et (corpus Buckwalter, 2002[4]), il est donc de plus en plus facile de travailler dans le cadre d'une approche supervisée. Comme on a noté dans le premier paragraphe soit le type des étiqueteurs supervisés ou non, les étiqueteurs avec apprentissage peuvent être regroupés en trois familles qu'on va détailler par la suite.

2.2.1. Etiquetage à base de règles

Ces types d'étiqueteurs s'appuient sur des règles grammaticales ou morphologiques, soit pour affecter une étiquette à un mot, soit pour définir les transitions possibles entre les différentes étiquettes Par exemple, une règle contextuelle peut dire qu'un mot X ambigu ou inconnu précédé d'un nom est un verbe. Cette règle servira à désambiguïser un mot en arabe comme « ذَهب » «***hb***», qui peut être un nom « de *l'or* » ou un verbe « *aller* ». Plusieurs étiqueteurs exploitent la morphologie pour faciliter la désambiguïsation. Par exemple, une règle qui tient compte de la morphologie peut spécifier qu'un mot ambigu ou inconnu qui se termine par «هُم » «**hom**» et qui est précédé par un verbe est un nom.

2.2.2. Etiquetage statistique

Ce type d'étiquetage caractérise les étiqueteurs qui utilisent des fréquences et des calculs de probabilité. La forme la plus simple d'étiqueteur statistique attribue les catégories pour chaque mot en se basant sur les catégories les plus fréquentes dans un texte de référence. Cette méthode est dite *robuste,* c'est-à-dire qu'elle donne toujours des résultats, même si elle produit beaucoup d'étiquettes erronées, car le même mot recevra toujours la même étiquette, quel que soit son contexte.

Il existe une autre approche cherchant à calculer la probabilité qu'une certaine *séquence* d'étiquettes apparaisse. Cette approche est appelée approche par « n-grammes ». Les algorithmes les plus connus pour implémenter une approche de ce type sont des

[2] Expert Advisory Group on Language Engineering Standard
[3] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T20
[4] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L49

algorithmes de programmation dynamique, comme l'algorithme de Viterbi 1967[5]. Une approche encore plus complexe, connue sous le nom de Modèle de Markov caché (HMM), combine les deux approches précédentes. Elle utilise à la fois la probabilité qu'une séquence d'étiquettes apparaisse et la fréquence des étiquettes pour un mot. Le modèle de Markov caché traite le texte comme une séquence d'états cachés (la séquence des catégories), chaque état produisant des *émissions* (les mots observables de la phrase). Dans le modèle de Markov, on considère que le choix d'une étiquette pour un mot doit dépendre des « n » étiquettes précédentes, et du mot lui même. Le passage d'un état à l'autre, c'est-à-dire d'une étiquette de catégorie à la suivante, est appelé une *transition*. Des algorithmes tels que le forward/backward ou l'algorithme de Baum Welch permettent d'identifier la séquence d'étiquettes maximisant la probabilité des transitions pour une suite de mots donnés.

Pour estimer la probabilité que l'étiquette X soit suivie de l'étiquette Y, on peut se baser sur les fréquences observées dans un corpus d'entrainement, avec la formule suivante :

Prob (x|y) = freq(x , y) / freq(y).

2.2.3. Etiquetage à base de réseaux de neurones

Un réseau de neurone peut être implémenté sous la forme d'un arbre où chaque nœud représente un neurone interconnecté avec les autres neurones par des liens inhibiteurs ou activateurs. Les liens pondérés permettent de modéliser l'influence des neurones entre eux. Si la somme des poids des liens des neurones actifs qui vont vers un neurone dépasse un certain seuil (qui varie d'un neurone à l'autre) le neurone produira une valeur en sortie susceptible d'activer d'autres neurones.

Cette méthode d'étiquetage est rarement utilisée dans le domaine du TAL à cause de ses difficultés d'application.

3. L'étiquetage pour la langue arabe

Comme le notent (Thi Minh, Laurent & Xuan, 2003, p 79) « L'étiquetage morpho-syntaxique automatique est processus qui s'effectue généralement en trois étapes :

[5] http://perso.telecom-paristech.fr/~vallet/dom_com/bacuvier/cadreviterbi.html

- La segmentation du texte en unités lexicales.
- L'étiquetage qui consiste à attribuer pour chaque unité lexicale l'ensemble des étiquettes morpho-syntaxiques possibles.
- La désambiguïsation qui permet d'attribuer, pour chacune des unités lexicales et en fonction de son contexte, l'étiquette morpho-syntaxique pertinente. »

Pour ces différentes étape, l'arabe pose différents problèmes, que nous allons essayer de résumer dans les points suivant :

3.1.1. L'ambiguïté

En traitement automatique des langues naturelles, le principal problème à résoudre est l'ambiguïté. Il existe différents types d'ambiguïtés. D'abord, les mots peuvent être ambigus aux niveaux lexical ou grammatical. Le mot « ذهب» « ***hb »** est ambigu lexicalement. Il peut désigner « l'or » en français ou encore le verbe « aller ». « كتب» « **ktb »** quant à lui, est ambigu grammaticalement. Il peut appartenir à plusieurs catégories grammaticales différentes : verbe ou nom. Le sens de ce mot sera très différent selon sa catégorie : nom = kAtib « écrivain », verbe = « écrit ». Il existe aussi des ambiguïtés qui relèvent du niveau syntaxique. Une même phrase peut avoir plusieurs sens possibles en fonction de ses interprétations syntaxiques.

3.1.2. Les mots inconnus

Les mots inconnus, du fait de leurs très grandes ambiguïtés, posent un véritable problème pour le traitement de l'arabe. En effet, généralement un mot inconnu peut avoir plusieurs catégories, contrairement aux mots connus répertoriés dans un dictionnaire. Certains étiqueteurs à base de règles peuvent traiter les mots inconnus. Par exemple (Vasilakopoulos, 2003), rapporte un taux d'efficacité de 87 % pour l'anglais. Pour ce qui est des étiqueteurs statistiques, leur taux efficacité est d'environ 85 % pour l'anglais (Vasilakopoulos, 2003). Mais pour la langue arabe, ce problème reste très difficile à traiter à cause de l'ambiguïté de ces types de mots.

3.1.3. Absence des voyelles

La majorité des textes en arabe sont écrits à l'aide de lettres non voyellées, bien que les voyelles (qui sont réalisées sous la forme de signes diacritiques placés au-dessus ou

au-dessous des lettres pour éviter l'ambiguïté d'un mot), apparaissent dans certains textes religieux (Coran hadith) ou littéraires (poésie classique, notamment). Pour compliquer la situation, et comme le remarque Joseph Dichy, la voyellation peut être partielle et l'usage n'est pas systématique sur ce point : « *L'écriture de l'arabe courante ne note pas les voyelles brèves, la gémination des consonnes, les marques casuelles composées d'une voyelle brève suivie, pour les noms et les adjectifs indéterminés d'une consonne « ن». On parle donc d'une écriture non-voyellée. On notera que la voyellation partielle ne repose pas sur une codification appuyée sur une tradition : elle ne présente pas un caractère systématique* » (Dichy, 1997)

Un mot sans voyelles peut générer plusieurs cas d'ambiguïtés lexicales et morphologiques. Par exemple le mot sans voyelle « ktb » possède 17 voyellations potentielles, représentant 9 catégories grammaticales différentes (voir tableau ci- dessus).

Mot avec voyelles	**TranslittérationBuckwalter**	**Traduction**	**Catégorie**
كَتَبَ	kataba	a écrit	Verbe Accompli voix active, 3ème personne, masculin, singulier
كُتِبَ	kutiba	a été écrit	Verbe Accompli voix passive, 3ème personne, masculin, singulier
كَتَّبَ	kattaba	a fait écrire	Verbe Accompli voix active, 3ème personne, masculin, singulier
كُتِّبَ	kuttiba	a été fait écrire	Verbe Accompli voix active, 3ème personne, masculin, singulier
كَتِّبَ	kattiba	fais écrire	Verbe impératif, 2ème personne, masculin, singulier

كُتُبٌ	kutubuN	Substantif, masculin, pluriel, nominatif, indéterminé
كُتُبٍ	kutubiN	Substantif, masculin, pluriel, génitif, indéterminé
كَتبٌ	katbuN	Substantif, masculin, singulier, nominatif, indéterminé
كَتبٍ	katbiN	Substantif, masculin, singulier, génitif, indéterminé
كَ + تَبٍّ	ka + tabbi	Préposition + Substantif, masculin, singulier, génitif, déterminé
كَ + تَبٍّ	ka + tabbiN	Préposition + Substantif, masculin, singulier, génitif, indéterminé

Tableau 11 : Différentes catégories de mot voyellé « ktb »

3.1.4. L'ordre du mot dans la phrase

La phrase arabe est caractérisée par une grande variabilité au niveau d'ordre de ses mots. En général, dans la langue arabe on met au début de la phrase le mot sur lequel on veut attirer (nom ou verbe) l'attention et on termine sur le terme le plus riche pour garder le sens de phrase. Cette variabilité de l'ordre des mots, provoque des ambiguïtés syntaxiques artificielles dans la mesure ce pour cela il faut donner dans la grammaire toutes les règles de combinaisons possibles d'inversion de l'ordre des mots dans la phrase qui sont réalisées par des linguistes.

3.2. L'étude des étiqueteurs existants

3.2.1. Aramorph

C'est un analyseur distribué par le LDC (Linguistic Data Consortium) qui permet de segmenter un mot en trois séquences (préfixe racine post-fixe). Le préfixe varie entre 0 et 4 caractères, le postfixe varie de 0 à 6 caractères et la racine varie de 1 à plusieurs caractères. Il est réalisé à partir de trois fichiers de lexique : préfixe (598 entrées), postfixe (906 entrées) et racines (stem, 78 839 entrées) (Buckwalter, 2002). Ces trois fichiers sont complétés par trois tables de compatibilité utilisées pour faire la combinaison entre préfixe et racine (2 435), post fixe et racine (1 612 entrées) et préfixe-postfixe (1 138 entrées), pour générer en sortie un fichier qui contient la forme lexicale d'un mot, sa catégorie et sa traduction en anglais.

Les points faibles de cet analyseur se résument dans ces différents points : (Attia, 2006)

- Absence de règles de générations : tous les lemmes sont listés manuellement et tous les lexèmes des formes fléchies associées sont énumérés, ce qui finit par augmenter le coût de maintenance de lexique.
- Problème au niveau du traitement des proclitiques interrogatifs qui se localisent au début des verbes et des noms (exemple : «أأقول », « أمحمد »).
- Manque de spécification de certaines formes à l'impératif : seulement 22 verbes sur un total de 9 198, soit 0,002% ont des formes impératives.
- Manque de spécification de certaines formes à la voix passive : seulement 1 404 verbes sur un total de 9 198, soit 15% sont conjugués à la voix passive au présent et 110 verbes au passé.

3.2.2. ASVM

C'est un analyseur gratuit développé en perl par l'équipe de Mona Diab en 2004. Il s'agit d'une adaptation à l'arabe du système anglais « Yamcha » qui a été entrainé sur le corpus annoté Treebank, en utilisant le modèle *Support Vector Machine* et en se basant sur 24 étiquettes.

L'étiqueteur ASVM est réalisé à partir de trois modules qui permettent de générer la sortie attendue :

- TOKrun.pl pour la tokenisation.
- LEMrun.pl pour la normalisation des mots féminins uniquement (ce n'est pas une vraie lemmatisation).
- POSrun.pl pour l'étiquetage.

D'après (Diab, Hacioglu & Jurafsky, 2004) l'évaluation d'ASVM se fait selon le corpus TreeBank arabe qui se compose de 4 519 phrases. Le corpus est distribué comme suit : 119 phrases pour le développement, 400 phrases pour le test et 4 000 phrases pour l'apprentissage.

Les résultats obtenus de SVM-POS sont de 95.49% d'étiquettes correctes. En effet, nous avons nous-même analysé les résultats de cet étiqueteur (nous les avons utilisés pour notre application), et nous avons remarqué que la majorité des erreurs sont liées à la mauvaise segmentation de l'article « Al » et à la confusion des noms avec les adjectifs et inversement.

Par la suite, l'équipe de Mona Diab a réalisé une amélioration de cet étiqueteur (Diab, 2010) sous le nom d'AMIRA 2.0[6] (non téléchargeable), qui obtient des résultats plus performants au niveau de la segmentation (99.2%) et une précision de plus de 96% au niveau de l'étiquetage.

3.2.3. L'étiqueteur APT de Khoja

Il s'agit d'une adaptation à l'arabe du système anglais BNC qui combine des données statistiques et des règles pour déterminer tous les traits morphologiques d'une unité lexicale. Cet analyseur est entrainé à partir d'un corpus contenant 50 000 mots en utilisant 131 étiquettes. D'après (Khoja, 2001) le nombre élevé de tags permettrait de donner des résultats plus efficaces au niveau de l'étiquetage.

3.2.4. MorphArabe

C'est un analyseur développé en langage orienté objet par l'équipe lyonnaise SILAT en 2002. C'est un programme informatique qui présente une collection de classes et d'objets. Il est entrainé sur le corpus Dinar.1. Il identifie les traits morphosyntaxiques des

[6] http://nlp.ldeo.columbia.edu/amira/

mots. D'après (Abbes, 2004), le moins ambigu des marqueurs est la racine. L'ajout de nouveaux traits augmente la discrimination dans l'analyse et offre plus de solutions.

3.2.5. Analyseur Sakhr

C'est un analyseur développé par (Chalabi, 2004) qui permet de traiter aussi bien l'arabe moderne[7] que l'arabe classique[8]. Il permet de définir la forme de base par suppression de tous les préfixes et les suffixes du mot à traiter et donne les traits morphologiques de ce dernier. D'après le site web de la société sakhr[9], cet analyseur donne des résultats qui atteignent une précision de plus de 90%.

3.2.6. Etiqueteur de XEROX :

La phase de segmentation pour cet analyseur est faite par un transducteur à états finis (Farghaly & Dichy, 2003) en découpant la chaine d'entrée en unités lexicales qui correspondent à une forme fléchie ou une ponctuation, en donnant à chaque segment des étiquettes qui représentent le comportement morphologique de chaque unité lexicale et sa catégorie. Cet étiqueteur regroupe 4 930 racines et 400 modèles qui permettent de produire 90 000 lexèmes. Cet analyseur utilise des règles à large couverture, par contre il génère un taux assez élevé d'ambiguïtés lexicales. Il ne traite pas bien la phase de désambiguïsation.

3.2.7. TAGGAR :

C'est un analyseur morphosyntaxique spécialement développé pour la synthèse vocale arabe des textes voyellés. Il prend en considération l'ordre de traitement des mots pour minimiser les erreurs d'étiquetage. Le traitement se fait dans l'ordre suivant : analyse des mots outils et des mots spécifiques, analyses des formes verbales et enfin, analyse des formes nominales. Cet analyseur utilise 35 étiquettes grammaticales qui se répartissent en trois grandes familles de catégories : 4 étiquettes pour les particules, 16 étiquettes pour les verbes, et 15 étiquettes pour les noms.

L'évaluation a été faite sur un corpus de 5 563 mots ; TAGGAR a engendré un taux d'erreur de 2% sur les étiquettes ce qui a entrainé seulement 1% d'erreurs sur les frontières

---

7 Le langage de la presse, de la littérature et de la correspondance formelle.

8 La langue du Coran, parlée au VIIe siècle.

9 http://www.sakhr.com

de groupes syntaxiques. Près de 98% des pauses insérées automatiquement sont correctement placées (Zemirli & Khabet, 2004).

4. Conclusion

Les différentes recherches effectuées sur l'analyse morphosyntaxique de la langue arabe montrent que c'est une langue très difficile à traiter à cause de l'agglutination et des ambiguïtés graphiques. Dans ce chapitre, nous avons présenté les différents problèmes de l'étiquetage de la langue arabe et nous avons donné des évaluations pour quelques étiqueteurs existants. Il nous apparait que les erreurs d'étiquetage sont souvent liées à l'étape de tokenisation. La mise en œuvre d'une meilleure segmentation lexicale des textes arabes constitue donc une piste intéressante pour l'amélioration de l'étiquetage et guidera les développements que nous allons mener par la suite.

Partie2 :

Développement d'un nouvel outil pour l'étiquetage de l'arabe

Introduction :

Le traitement automatique de la langue arabe s'est rapidement développé au cours de ces dernières années. L'étiquetage morphosyntaxique pour cette langue reste toujours un sujet d'intérêt pour de nombreux chercheurs du fait de son rôle de brique de base dans de nombreuses applications du TAL. Bien que de nombreux systèmes aient été réalisés selon des méthodes différentes, les pistes d'amélioration sont encore très ouvertes.

En vue de fournir un outil gratuit, efficace, générique, et d'une utilisation facile par la communauté des chercheurs, TreeTagger parait être un choix adapté : il est très répandu, tout en produisant des résultats de qualité.

Le fonctionnement de cet outil est basé sur un lexique, ainsi qu'un corpus d'apprentissage pour l'entraîner. Parmi ses avantages, il est relativement facile à adapter sur n'importe quelle langue. C'est pourquoi, nous avons choisi de l'étudier au cours de notre recherche.

Pour mettre en œuvre TreeTagger, il faut préalablement effectuer un prétraitement du texte en entrée : le texte doit être tokenisé, c'est-à-dire segmenté au niveau lexical.

Chapitre1 : segmentation des unités lexicales

1. Introduction

La segmentation est un processus nécessaire dans le traitement morphologique de la langue. Le but de la segmentation est de diviser un texte en une suite de tokens[10] afin de préparer le traitement morphosyntaxique (étiquetage ou POS tagging en anglais).

La phase de segmentation consiste donc à identifier les frontières des mots, c'est-à-dire des unités lexicales autonomes que l'on cherche à étiqueter au niveau des principales parties du discours (verbe, nom, adjectif, adverbe, déterminant, etc.). La segmentation vise aussi à détacher des unités morphémiques qui n'ont pas de forme libre, comme les clitiques, et qui apparaissent attachés à d'autres mots (par exemple le déterminant pour le nom et la préposition de conjugaison pour le verbe). Ces unités sont difficiles à identifier en arabe parce qu'elles présentent des ambiguïtés sur le plan graphique (de par leur ressemblance avec d'autres éléments constitutifs des mots). La segmentation a été décrite dans plusieurs recherches comme une phase cruciale avant le traitement syntaxique (Habash & Rambow, 2005), car la qualité des résultats finaux en dépend.

Ce chapitre est consacré à la description de la méthode mise en œuvre pour la segmentation, ainsi que des ressources utilisées.

2. Principe de segmentation

Le problème de la segmentation ne se pose pas de la même manière pour toutes les langues. Pour les langues comme l'anglais ou le français, les unités lexicales (tokens) sont dans la plupart des cas reconnaissables par une simple analyse graphique en s'appuyant sur les caractères séparateurs (espaces, ponctuations, apostrophe, etc.) présents dans les textes.

L'arabe, quant à lui, est en principe monosyllabique, ce qui signifie que chaque syllabe peut être une unité lexicale. Il est possible de former des mots complexes à

[10] Ensembles des unités morphosyntaxiques minimales (mot, une partie de mot, clitique..…).

partir de plusieurs syllabes, ce qui rend difficile le problème de segmentation. Pour la langue arabe, la majorité des travaux de segmentation se basent sur des règles qui s'appuient sur des listes de clitiques, préfixes, suffixes et racines (Mars & al, 2008). Ces règles s'appuient sur les principes de constitution d'un mot complexe et son contexte dans la phrase.

C'est pourquoi, il est difficile d'identifier la racine (unité lexicale) pour les mots qui contiennent des flexions (exemple : les terminaisons des verbes conjugués).
Au cours de notre recherche, nous avons essayé d'élaborer un algorithme de segmentation en nous basant sur des règles qui traitent dans la majorité des cas la forme correcte d'un mot en arabe. Le succès de notre méthode repose essentiellement sur une grande liste de mots déjà bien segmentés.

Notre algorithme de segmentation est composé de trois modules organisés de manière séquentielle.

- D'abord, on effectue une segmentation grossière au niveau des espaces et des signes de ponctuations.
- Ensuite, on examine les tokens ainsi obtenus, et on les compare avec les formes déjà segmentées d'un corpus traité de façon semi-automatique (cf. section suivante pour une description du corpus). Si le token est trouvé dans ce corpus, sa segmentation est validée.
- Si le token est absent du corpus, on recherche grâce à une expression régulière qui représente la forme complète d'un mot arabe (pré-bases racine post-bases), les éventuelles pré-bases et post-bases attachés à la racine. Cette expression est construite à partir de listes définies à l'avance. Pour chaque pré-base ou post-base identifiée, nous vérifions le statut de la partie restante du mot découpé.

Avec cette méthode, nous avons remarqué qu'il reste des ambiguïtés de découpage pour certains mots qui peuvent se découper de plusieurs façons différentes. Par exemple le mot suivant en arabe peut être découpé de trois manières différentes en fonction de son contexte dans la phrase (voir tableau suivant):

Découpage possible	Traduction en français
أ + لَم + هُم	les a-t-il ramassés ?
أَلم + هُم	leurs douleurs
أَل + مُهم	l'important

Tableau 12 : Les différents découpages du mot أَلَمهم (>lmhom)

Ce problème reste toujours difficile à résoudre puisque le découpage de ces types de mots dépend obligatoirement du contexte et de sa position dans la phrase. Pour ce cas, notre algorithme de segmentation, d'abord, prend la totalité du mot et le découpe en utilisant une expression régulière qui représente une règle appliquable. Ensuite, il le compare aux mots existants dans le corpus en retenant le découpage valide. C'est pourquoi, la qualité de la segmentation dépend de la taille du corpus, censé contenir les mots les plus fréquents en arabe avec leur segmentation correcte. Cependant, pour résoudre le cas d'ambiguïté au niveau de découpage reste une tâche non triviale.

Le schéma suivant illustre le principe de notre algorithme de segmentation :

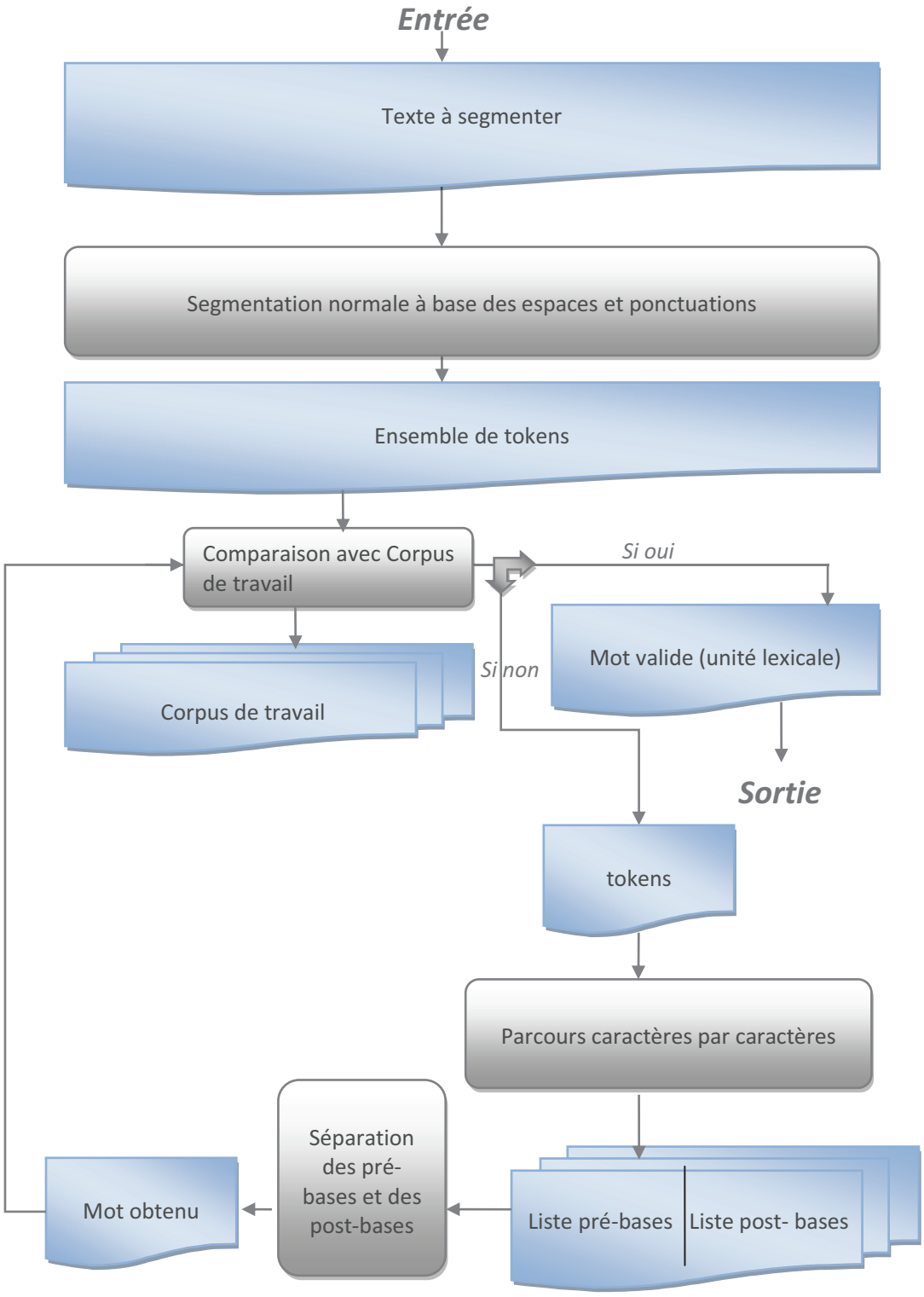

Figure 2 : Principe de la segmentation

3. Données utilisés

Dans l'idéal, pour une large couverture de la langue arabe, il faudrait que le corpus pré-segmenté soit très vaste, afin d'avoir un vocabulaire étendu. N'ayant pas ces ressources à notre disposition, et pour gagner du temps, nous avons utilisé le même corpus que pour l'entrainement de l'étiquetage, ce que nous appelons notre corpus du travail (voir section prochaine).

Par ailleurs, nous avons développé deux listes complémentaires de pré-bases et post-bases fréquentes en arabe (plus des détails dans la suite).

3.1. Constitution du corpus de travail

Malgré les différentes recherches effectuées sur le traitement automatique de la langue arabe, il nous a été difficile de trouver des ressources toutes faites. C'est pourquoi, nous avons décidé de constituer notre propre corpus. Pour avoir un vocabulaire suffisamment étendu, nous avons pris le corpus EASC proposé par (EL-Haj, kruschwitz & Fox, 2010) qui comporte 153 articles répartis sur une dizaine de domaines différents. Pour obtenir ce corpus nous avons envoyé un mail à Mahmoud EL-Haj qui nous a donné les différents articles. Le tableau suivant donne des statistiques par domaine sur la constitution de notre corpus (pour un extrait de notre corpus du travail, voir annexe 8, p 80)

Domaine	Nombre de phrases	Nombre d'occurrences / nombre de mots différents	Nombre de caractères
Art	149	3848 / 2223	40 450
Religion	125	3248 / 1845	34 452
Education	85	2524 / 1474	26 994
Science et technologie	200	6371 / 3012	68 186
Environnement	537	13494 / 5790	141 300
Sport	161	3327 / 1755	34 554
Finance	**212**	4718 / 2545	51 817
Tourisme	336	6730 / 3262	70 693
Santé	262	5733 / 2781	61 609
Politique	271	8240 / 4072	90 032
Total	2338	58 233 / 21 238	619 988

Tableau 13 : Constitution du corpus de travail

Comme on l'a dit précédemment, le principe de notre segmentation dépend essentiellement de la comparaison de chaque mot avec les mots pré-segmentés de notre corpus de travail. Pour réaliser cette tâche de pré-segmentation on a suivi les étapes suivantes :

- D'abord, nous avons regroupé tous les articles choisis en un seul fichier.

- Ensuite, pour éviter tous les problèmes de codage de la langue arabe, et pour faciliter le traitement automatique, nous avons écrit un script perl qui permet de translittérer ce fichier selon la table de Buckwalter (voir 73 p. 72). Cette translittération, souvent appliquée en TAL, présente l'intérêt de n'utiliser que des caractères ASCII et d'être totalement réversible (c'est-à-dire qu'il est aisé de retrouver le texte original en appliquant la translittération inverse).

- Puis, nous avons appliqué l'étiqueteur ASVM 1.0 réalisé par (Diab, Hacioglu & Jurafsky, 2004) sur notre fichier afin d'obtenir notre propre fichier segmenté selon cet étiqueteur. Comme on l'a déjà noté, cet étiqueteur fait beaucoup d'erreurs au niveau de la segmentation, et il a été nécessaire de corriger manuellement le fichier résultat pour éliminer la plupart de ces erreurs.

Nous avons effectué la phase de correction en deux étapes :

- Dans un premier temps, nous avons implémenté un script perl qui permet de corriger la majorité des erreurs au niveau de l'article « Al ». Ce script cherche tout d'abord tous les mots qui commencent par l'article « Al ». Ensuite, à l'aide d'une expression régulière, il identifie les suites des caractères qui suivent l'article « Al » et les compare aux autres formes de notre corpus. Si la séquence existe, il découpe le mot au niveau de l'article (séparation entre l'article et le reste du mot). Sinon, il passe au mot suivant.

- Puis, nous avons commencé à corriger notre corpus manuellement en utilisant des expressions régulières pour étendre les corrections effectuées à l'ensemble du corpus. Ne disposant que d'un temps limité pour cette recherche, il ne nous a pas été possible de corriger manuellement la totalité de notre corpus - il faut compter environ 30 secondes pour corriger un mot mal segmenté et généraliser sa correction à l'ensemble du corpus, grâce à un

rechercher/remplacer. Nous avons corrigé les 13 000 premières lignes (soit environ 400 phrases) et avons abouti à un script contenant 1 184 expressions régulières (extrait voir annexe 12, p 90). Ces expressions régulières peuvent s'appliquer sur n'importe quel corpus. Elles sont définies manuellement pour corriger les mots qui existent dans notre corpus (c'est-à-dire pour un mot, à partir d'une expression régulière en utilisant la technique « rechercher/ remplacer », et corriger ce mot dans la totalité du corpus). Notons cependant qu'à l'issue de ces corrections, la présence de mots mal segmentés devenait de plus en plus rare (pour donner une estimation, on trouve des erreurs environ tous les 200-300 mots).

3.2. Listes des pré-bases et post-bases

La séparation des pré-bases et des post-bases est une phase intéressante dans la segmentation de texte arabe. Comme pour la plupart des langues, il est possible d'énumérer les pré-bases et les post-bases les plus fréquentes, car elles forment une liste fermée. Dans notre travail on a défini une liste des pré-bases dont la longueur varie entre 1 et 4 caractères (b, w,k,Al, fbAl, etc.) et une liste de post-bases dont la longueur varie entre 1 et 6 caractères (h,hA,hom,homA, etc.). Pour gagner du temps, nous avons pris les listes réalisées par (Abbes, 2004). En outre, pour faciliter le travail, nous nous sommes intéressés seulement aux pré-bases et post-bases les plus utilisées pour construire nos propres listes. Celles-ci contiennent 77 pré-bases et 110 post-bases (extrait : voir annexe 3 et annexe 4, p. 74-75).

4. Exemples de phrases segmentées par notre script et évaluation

Voici quelques phrases segmentées par notre script :

- Phrase 1 : **wkAn qd sbq *lk ftrp qSYrp nmt fYhA HrkAt kAnt ttrqb mjY' AlmwEwd Al*Y b$rt bh Alktb AlsmAwYp**.

  وكانَ قد سَبق ذلك فترَة قصيرة نَمت فيهَا حَركات كَانت تَترقب مَجيء المَوعود الذِي بَشرت به الكُتب السَموية

⇨ W / kAn **/** qd/ sbq/ *lk / ftrp / qSYrp / nmt / fY/ hA / HrkAt / kAnt / ***ttrqb*** / mjY' / Al / mwEwd / Al*Y / b$rt / b / h / Al / ktb / Al / smAwYp .

Dans cette phrase, notre script a généré une seule erreur au niveau du verbe qui reste attaché au caractère qui désigne le genre du sujet qui précède le verbe et qui le détermine. Cette erreur provient de la mauvaise segmentation du mot dans notre corpus.

- Phrase 2 : **>mA Al$rYEp fhY Asm lAl>HkAm AlEmlYp AltY txtlf bAxtlAf Alrsl.**

أمّا الشَريعة فهي إَسم للأحكَام العَملية التِي تَختلف باختِلاف الرُسُل

⇨ >mA / Al /$rYEp / f / hY / Asm / l / Al / >HkAm / Al / EmlYp / AltY / txtlf / b /AxtlAf / Al / rsl.

Dans cette phrase, notre script fait une segmentation parfaite (aucune erreur).

- Phrase 3 : **wqAl Almjls <n AlSlAp bAllgtYn tnHrf En AltEAlYm Al<slAmYp .**

وقال المَجلس إنَّ الصَلاة باللُّغتين تنْحَرف عَن التَّعَاليم الإسلَامية

⇨ W / qAl / Al / mjls / <n / Al /SlAp / **b / Al / lgt / Yn** / tnHrf / En / Al /tEAlYm /Al / <slAmYp.

- Phrase 4 **: wAktsbt AvnAn mn AlsYmfwnYAt AltY ktbhA fY Smmh >kbr $EbYp,whmA AlsYmfwnYp AlxAmsp wAltAsEp.**

واكتَسَبت إثنَان مِن السَمفونيات الَتِي كَتبَها فِي صمِّه أكبر شَعبية، السَمفونِية الخَامِسة والتَّاسِعة

⇨ W / ***Aktsbt*** / AvnAn / mn / Al / sYmfwnYAt / AltY / ktb / hA / fY / Smm /h / >kbr / $EbYp/ w / hmA / Al / sYmfwnYp / Al / xAmsp / w / Al / tAsEp.

Dans cette phrase, on trouve une erreur au niveau de la terminaison du verbe. Malgré le fait que l'on a bien défini la post-base dans notre liste, notre script ne traite pas ce mot à cause de sa présence telle quelle dans notre corpus de travail. Il le prend donc comme un mot valide.

Sur le plan quantitatif, on note que sur 51 mots on a obtenu 48 mots bien segmentés (94%)

On peut considérer que les mots qui se terminent par « At » sont des mots valides (ces types de mots sont considérés en arabe comme des noms pluriels). Cette évaluation sommaire, qui porte sur un très petit échantillon du corpus, nous semble malgré tout représentative des phénomènes que nous avons rencontrés sur de plus grandes quantités de textes. Ces résultats nous semblant satisfaisants, nous avons décidé de conserver en l'état ce script de segmentation pour préparer l'étape de l'étiquetage.

5. Conclusion

Dans ce chapitre, nous avons présenté le principe de notre segmentation de la langue arabe ainsi que les différentes ressources utilisées pour la réalisation de cette tâche. L'évaluation sommaire de notre script a montré des premiers résultats encourageants. Comme le principe de notre segmentation repose sur la comparaison de chaque mot segmenté avec les mots pré-segmentés de notre corpus de travail, celle-ci pourrait toujours être améliorée en complétant ce corpus afin d'obtenir une plus grande couverture.

Chapitre 2 : L'étiquetage avec TreeTagger

1. Introduction

En général, l'étiquetage morphosyntaxique d'une langue est un processus qui consiste à ajouter aux mots des informations morphologiques concernant leurs catégories morphosyntaxiques ou parties du discours. Pour la langue arabe, l'étiquetage reste toujours une étape complexe à aborder à cause des ambiguïtés lexicales des unités. Selon (Laporte, 2000) : « *l'analyse morphosyntaxique est l'ensemble des techniques qui concourent à passer d'un texte brut, exempt d'informations linguistique, à une séquence des mots étiquetés par des informations linguistiques* ». Comme on a noté dans l'état de l'art, l'étiquetage de la langue arabe est principalement basé sur deux types d'approche : étiquetage à base de règles et étiquetage avec apprentissage. L'outil TreeTagger que nous avons utilisé dans notre application concerne cette dernière catégorie de système, et fait recours à des modèles probabilistes (modèles de chaine de Markov cachés HMM et arbres de décision).

Ce chapitre, est consacré à la description de l'outil en question ainsi les différents ressources utilisés pour l'étiquetage afin de réaliser notre étiqueteur en basant sur cet outil.

2. Définition de TreeTagger

2.1. Principe

Le TreeTagger est un outil permettant l'étiquetage morphosyntaxique et la lemmatisation. Il a été développé par Helmut Schmid[11] dans le cadre de projet TC[12] dans l'ICLUS (Institut for Computational Linguistics of the University of Stuttgart). Cet outil est gratuit, disponible en ligne, et facile à installer sur les systèmes d'exploitations Linux ou Windows. Il a été utilisé avec succès pour étiqueter des nombreuses langues (anglais, français, allemand, italien, néerlandais, espagnol, bulgare, russe, grec, portugais, chinois, swahili …) . En théorie, il est adaptable sur toutes les langues, si un lexique et un corpus d'apprentissage manuellement étiquetés sont disponibles. Cet outil a été évalué sur des

[11] http://www.ims.uni-stuttgart.de/~schmid/
[12] http://www.ims.uni-stuttgart.de/projekte/tc/

nombreuses langues. Pour l'ancien français, et d'après (Stein, 2007), il a été évalué sur 500 000 mots avec un taux de précision 92,7%. Parmi les avantages de TreeTagger par rapport aux autres outils est la lemmatisation des mots. C'est pourquoi nous avons décidé de l'adapter sur la langue arabe.

Le principe de cet outil se rapproche de taggers n-grammes traditionnelle. Ce dernier est un outil qui permet à partir d'une séquence de n mots de calculer la probabilité de l'apparition d'une étiquette donnée sachant les étiquettes de n-1 mots qui le précède en se basant sur le modèle de Markov caché(HMM). L'entraînement de cet outil nécessite un corpus d'apprentissage bien étiquette. Le principe de « tagger n-gramme » est basé sur deux modèles (modèle d'étiquetage et modèle de langage). Par contre, TreeTagger utilise un arbre de décision binaire pour calculer la taille de contexte à utiliser afin d'estimer les probabilités de transition. Le TreeTagger fonctionne avec deux programmes :

- *train-tree-tagger :* comme son nom l'indique c'est programme pour l'apprentissage qui génère un fichier de paramètre (.par) à partir d'un corpus d'apprentissage et un lexique.
- *tree-tagger :* c'est l'étiqueteur proprement dit, qui prend en entrée le fichier de paramètre généré par le programme d'apprentissage et un fichier texte en argument et qui produit en sortie le texte étiqueté.

2.2. Construction d'arbre de décision :

Les arbres de décision sont l'une des nombreuses méthodes de classification. Les méthodes de classification ont pour but d'identifier les classes auxquelles appartiennent des objets à partir de certains traits descriptifs. Les algorithmes d'apprentissage basés sur les arbres de décision sont efficaces et disponibles dans la plupart des environnements de fouille des données (datamining). L'algorithme ID3 implémenté par (Quinlan, 1986) fait partie de ces types des algorithmes. La construction de l'arbre de décision se fait d'une manière récursive à partir d'un ensemble des séquences de mots (trigrammes) en utilisant une version modifié de l'algorithme ID-3. Dans chaque étape de récursivité, un test est créé qui divise l'ensemble des échantillons trigrammes en deux sous-ensembles avec une distinction maximale concernant la distribution de probabilité de la troisième étiquette. Le test examine l'un de deux étiquettes précédentes et vérifie si elle est identique à une étiquette t. un test a la forme suivante :

$$tag - i = t \; ; i \in \{1,2\} \; ; t \in T$$

Avec T : l'ensemble des étiquettes.

Pour chaque étape de récursivité, tous les tests possibles sont comparés. Le meilleur c'est celui qui donne la plus forte probabilité conditionnelle pour le nœud courant de l'arbre. Ensuite, ce nœud est développé d'une manière récursive sur chacune de deux sous-ensembles trigrammes qui sont définis par le test. Les deux sous-ensembles sont attachés au nœud courant et correspondent aux deux sous-arbres V et F du test. Le schéma suivant en donne une illustration :

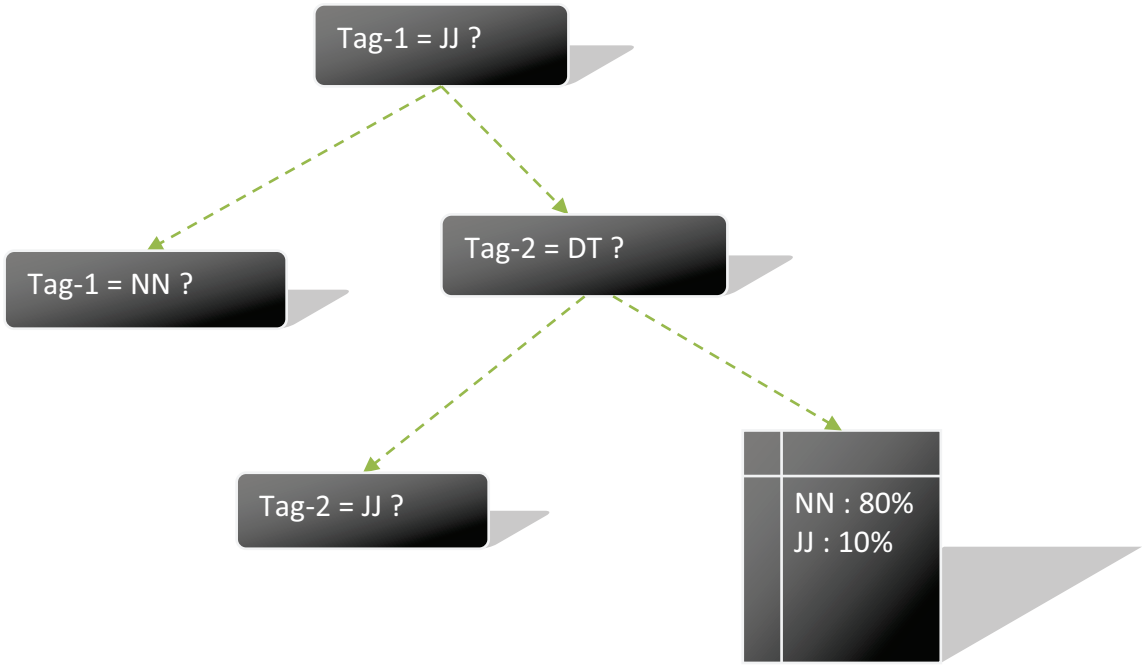

Figure 3 : Un simple arbre de décision

Le critère utilisé pour comparer tous les tests possibles Q est la quantité d'information qui est apporté par la troisième étiquette en faisant chaque test. En effet, maximiser le gain d'information est équivalent à minimiser la quantité moyenne d'information Iq qui est nécessaire pour identifier la troisième étiquette. Pour le calculer, on utilise cette équation :

$$Iq = -p(C+|C)\sum_{t\in T} p(t|C+)\log_2 p(t|C+) - p(C-|C)\sum_{t\in T} p(t|C-)\log_2 p(t|C-)$$

Avec :

- C : le contexte qui correspond au nœud courant.
- C+ : le contexte C sachant que la condition de test q est valide.
- C- : le contexte C sachant que la condition de test q est non valide.
- P(C+|C) : la probabilité sachant que le test q est valide.

- P(C-|C) : la probabilité sachant que le test q est non valide.
- P (t|C+) : la probabilité de la troisième étiquette si le test est valide.
- P (t|C-) : la probabilité de la troisième étiquette si le test est non valide.

Ces probabilités sont estimées à partir des fréquences avec la méthode du *maximun likelihood estimation* (MLE) (l'estimation du maximum de vraisemblance)[13] en appliquant ces formules :

$$p(C+|C) = \frac{f(C+)}{f(C)}$$

$$p(C-|C) = \frac{f(C-)}{f(C)}$$

$$p(t|C+) = \frac{f(t,C+)}{f(C+)}$$

$$p(t|C-) = \frac{f(t,C-)}{f(C-)}$$

Avec :

- f© : le nombre des trigrammes dans le corpus d'apprentissage actuel
- f(C+) : le nombre des trigrammes validés
- f(C-) : le nombre des trigrammes non validés
- f(t,C+) : le nombre des trigrammes validés et qui possède un troisième tag est égale à t
- f(t,C-) : le nombre des trigrammes non validés et qui possède un troisième tag est égale à t

Le développement récursif de l'arbre de décision s'arrête si le prochain test généré correspond à un sous-ensemble dont la taille est inférieure à un seuil prédéfini. Si on prend par exemple un seuil est égal à 2, il faut alors que f(C+) <2 ou f(C-) <2. Les probabilités p(t|C) de la troisième étiquette sont estimées à partir de tous les trigrammes qui correspondent à cette étape de récursivité, et ils sont stockés au niveau de nœud courant en se basant sur cette équation :

$$p(t|C) = \frac{f(t,C)}{f(C)}$$

[13] http://en.wikipedia.org/wiki/Maximum_likelihood

Après la version initiale de l'arbre de décision qui a été construit, ce dernier est optimisé, grâce à certaines simplifications. Si les deux sous-nœuds d'un nœud sont des feuilles, et si le gain d'information pondéré au niveau du nœud est inférieur à certain seuil, les sous-nœuds sont supprimés et le nœud devient une feuille elle-même. Le gain d'information pondéré est définit par :

$$G = f(C)(I0 - Iq)$$

$$I0 = \sum_{t \in T} p(t|C) \log_2 p(t|C)$$

Où I0 est la quantité d'information qui est nécessaire pour enlever l'ambiguïté au niveau du nœud courant, et Iq est la quantité d'information qui encore nécessaire après que le résultat du test q est connu. Comme la plupart des étiqueteurs statistiques, treetagger détermine la meilleure séquence d'étiquettes pour une séquence donnée de mots avec un algorithme de propagation dynamique (en l'occurrence l'algorithme de (Viterbi, 1967)).

3. Construction de données de TreeTagger pour l'arabe

La préparation des données pour un outil donné est une phase importante pour atteindre le but en considération. C'est pour cela, nous avons passé pas mal de temps sur la construction de ces données et surtout sur le lexique qui prend une forme spécifique (cf. plus de détails dans la section prochaine).

3.1. Choix de jeux d'étiquettes

L'objectif de notre travail est de réaliser un étiqueteur générique capable d'identifier les parties du discours (nom, verbe, adjectif, adverbe,…). Les fichiers de paramètres de TreeTagger proposent souvent de choisir entre des jeux d'étiquette restreints et des jeux enrichis (avec plus d'informations sur les traits). C'est pourquoi, et pour gagner de temps, nous avons décidé de prendre le jeu des étiquètes proposés par (Diab, Hacioglu & Jurafsky, 2004) car nous avons utilisé au début ASVM 1.0 pour la préparation de notre corpus de travail, malgré que dont très réduits au niveau de leur nombre en ajoutant une étiquette qui désigne la fin de phrase (étiquette spécifique pour TreeTagger). Ces jeux des étiquettes contiennent 21 étiquettes qui permettent d'identifier les principaux tokens en arabes. Le choix de ces étiquettes était obtenu à partir d'une adaptation de jeu d'étiquettes anglais en arabe lors de la création du corpus Treebank Arabic par Mona Diab. C'est pour cela l'identification des verbes (verbe passif, verbe parfait, verbe imparfait) dans ce jeu

n'est pas une spécificité de l'arabe, mais c'était identifié encore pour le jeu d'étiquette anglais dans le corpus Penn TreeBank de l'anglais. Le tableau suivant donne une idée précise sur ces étiquettes :

Etiquette	Explication
JJ	Adjective
RB	Adverbe
CC	Conjonction de coordination
DT	Déterminant
FW	Mot étranger
NN	Nom commun au singulier
NNS	Nom commun au pluriel
NNP	Nom propre au singulier
NNPS	Nom propre au pluriel
VBP	Verbe à l'imparfait
VBN	Verbe passive
VBD	Verbe parfait
RP	particule
PRP	Pronom personnel
PRP$	Pronom personnel possessive
CD	Cardinal (nombre)
IN	Conjonction de subordination
UH	Interjection
PREP	Préposition
WP	Pronom relatif
WRB	Wh-adverbe
PUNC	Ponctuation
SENT	Le point (fin de phrase)

Tableau 14 : Listes des étiquettes

3.2. Lexique

Toute analyse morphosyntaxique a besoin d'un lexique qui donne des informations sur l'usage grammatical de chaque unité lexicale. Ces informations varient d'un lexique à un autre. Comme nous avons noté au début de ce chapitre pour entrainer TreeTagger, nous avons besoin d'un lexique de la langue en question. Pour Treetagger, le fichier qui contient

les lexiques doit être représenté d'une manière spécifique : un format CSV avec trois colonnes séparés par des tabulations, contenant sur chaque ligne un mot, les catégories et le lemme associé. Notons que, une forme des mots peut avoir plusieurs catégories ainsi des lemmes. C'est pourquoi, il faut bien éliminer la redondance des mots dans notre lexique sur des lignes différents pour qu'il soit adaptable avec l'outil en question.

Dans notre étiqueteur le lexique joue un rôle important pour l'identification des catégories et du lemme de chaque mot en entrée. C'est pourquoi, nous avons passé un peu de temps au cours de notre recherche construire un lexique assez vaste. Pour ce faire, nous avons pris la liste de mots[14] proposés par (Buckwalter, 2002) utilisés pour la réalisation de l'étiqueteur « AraMorph ». Cette liste contient 82 158 racines qui représentent 38 600 lemmes. En arabe, la racine est une suite des lettres formant le radical du mot, elle forme la base du mot, et elle s'identifie à partir d'un schème donnée qui nous avons déjà décrit dans le premier chapitre de ce mémoire. Par contre, le lemme est la forme classique et entièrement vocalisée à la quelle se rattache la forme du texte (Tuerlinckx, 2004) :

- Les *verbes* sont ramenés à la 3^{e} pers. Sg. De l'accompli actif, sauf dans le cas de certains verbes figés n'ayant qu'une conjugaison partielle.
- Les *noms variables* sont ramenés à la forme du nominatif sg. (masc. Pour les noms qualificatifs), les *noms invariables* à leur forme classique vocalisée (masc. Sg. Pour les pronoms).
- les *particules* sont ramenées à leur forme classique vocalisée.

Le fichier qu'on a récupéré contient des informations additionnelles pour chaque mot comme la forme voyellée et la catégorie. Pour réaliser notre propre lexique, nous avons suivi la démarche suivante :

- Nous avons pris seulement les colonnes qui correspondent à la racine et son forme voyellé.
- Au début, nous avons sélectionné la colonne des racines dans un fichier. Ensuite nous avons étiqueté ce fichier avec l'étiqueteur « ASVM1.0 », afin d'obtenir la liste des catégories de chaque mot en entrée. Enfin, et à l'aide

[14] http://download.savannah.gnu.org/releases/aramorph/

des commandes Unix, nous avons corrigé la sortie de l'étiquetage en le mettant sous la forme de deux colonnes séparés par tabulation.

- Nous avons sélectionné la colonne qui contient les mots voyellés, et nous avons collé dans la troisième colonne de notre fichier en utilisant de la commande « awk » en Unix.
- Pour que le lexique soit adaptable à TreeTagger et pour gagner le temps, nous avons réalisé un script perl (voir annexe 11, p 89) qui permet d'éliminer la redondance de chaque mot et de le mettre dans une seule ligne sous la forme suivante : mot Cat1 lemme1 Cat2 lemme2……..CatN lemmeN en le séparant par tabulation. Ce script est basé sur une expression régulière qui fait une recherche sur tous les lignes qui contient le même mot et le remplace avec une seule ligne sous la forme correcte.
- Nous avons ajouté les listes de pré-bases et post-bases à l'entête de notre lexique pour le compléter.

Notons bien que le lexique ne contient pas des chiffres. Pour la phase de la lemmatisation dans notre lexique, ce n'est pas évident de générer tous les lemmes de mots en question. C'est pourquoi nous avons gardé pour la plupart des cas la forme voyellés du mot et nous avons ajoutés le caractère spéciale «-» comme lemme pour les pré-bases et les post-bases ajoutés à la main.

Voici un extrait de notre lexique :

```
 A      PRP$    -
Al      DT      -
b       PREP    -
f       IN      -
fA      IN      -
l       IN      -
w       CC      w
>w      IN      >w
Yn      PRP$    Yn
Anjbt   VBP     Anjb
>Y      IN      >Y
mwADYE  NNS     mwDwE
AvntY   CD      AvnAn
kY      IN      kY
gYr     RB      gYr
zYAd    NNP     zYAd
msrHYAtNNS      msrH
>sAsYp  JJ      >sAs
kAfp    RB      kAfp
<Dafp   RB      <Dafp
End     RP      End
fYrwz   NNP     fYrwz
```

```
>gAnY   NNS     gn~
gn~t    VBD     gn~
bYd     RP      bYd
jrAHp   NN      jrH
rHbAnY  NNP     rHbAnY
fwrYp   JJ      fwr
vwrp    NN      vAr
Ah      PRP$    -
AhA     PRP$    -
Ahm     PRP$    -
AT      VBN     >aT~
YstmE   VBN     smE
tElYqAtNNS      Elq
Al>rbEA'        NNP     Al>rbEA'
>rbEYnYAt       NNS     >rbE
Ydrs    VBD     drs
Yqr&    VBP     qr>
tzAwr   VBN     tazAwar
51awny  VBN     tawAnaY
AnhzAm  NN      {inohizAm
qtSd    VBP     qotaSid
Emlt    VBD     Eml
mtbAh   NN      mutabAh
|sn     JJ      |sin
```

Figure 4 : Extrait du lexique arabe

3.3. Corpus d'apprentissage

Comme son nom l'indique, un corpus d'apprentissage est une collection de données qui possède une forme bien définie à l'avance qui permet d'entrainer un outil en vue d'une tâche donnée. Pour créer notre corpus d'apprentissage nous avons pris le corpus de travail déjà utilisé pour la segmentation et nous l'avons divisé sur deux sous-corpus comme suit : un corpus de test qui contient les 234 premières phrases et le reste pour le corpus d'apprentissage. Notre corpus d'apprentissage contient 52 171 mots dont 19 086 mots différents regroupés dans 2 096 phrases. Notons que l'ensemble des mots de notre corpus est non voyellés. Le tableau suivant représente la distribution de différentes étiquettes utilisées dans notre corpus avec le pourcentage par rapport à l'ensemble des étiquettes :

Etiquette	**Nombre d'occurrences**	**%d'utilisation**
CC	5099	6,40
CD	1222	1,50
DT	11264	14,32
FW	192	0,20
IN	8 103	10,30
JJ	5 603	7,10
NN	25 920	32,95
NNP	2 487	3,10
NNS	2 870	3,60

NOFUNC	8	0,01
PREP	1 781	2,20
PRP	3 235	4,10
PRP$	185	0,20
PUNC	2 752	3,40
RB	1 018	1,30
RP	6 520	8,20
SENT	2 179	2,77
VBD	2 859	3,60
VBN	378	0,50
VBP	2 036	2,60
WP	1 716	2,20
WRB	2	0,0025
Total	78 650	100

Tableau 15 : Distribution des étiquettes dans le corpus d'entraînement

La figure suivante présente un extrait de notre corpus d'apprentissage après la correction du sorite d'ASVM1.0 :

```
w       CC
ntYjp   NN
l       IN
*lk     WP
f       IN
<n      RP
hnAk    RB
ArtbAT  NN
tArYxY  JJ
bYn     RB
Al      DT
bhA}Yp  NN
w       CC
Al      DT
bAbYp   NN
.       SENT
w       CC
bEd     RP
>n      RP
$AE     VBD
>mr     NN
Al      DT
bAbYp   NN
qAmt    VBD
Al      DT
slTAt   NNS
Al      DT
<YrAnYpJJ
,       PUNC
b       PREP
<YEAz   NN
mn      IN
```

```
rjAl    NNS
Al      DT
dYn     NN
,       PUNC
b       PREP
tE*Yb   NN
AlbAb   NN
Yn      PRP
w       CC
Al      DT
qbD     NN
Ely     RP
AlbAb   NNP
snp     NN
1847m   CD
w       CC
<YdAE   VBP
h       PRP$
Al      DT
sjn     NN
.       SENT
```

Figure 5 : Extrait du corpus d'apprentissage

3.4. Classes ouvertes

C'est un fichier qui contient les étiquettes les plus utilisées dans le corpus d'apprentissage. Au niveau de l'apprentissage, l'outil que nous avons appliqué utilise ce fichier pour identifier l'étiquette d'un mot qui n'existe pas dans le corpus d'apprentissage. Voici le fichier définit au cours de notre travail :

```
FW JJ NN NNP NNS RB VBD VBN VBP
```

Figure 6 : Classes ouvertes

4. Adaptation de TreeTagger sur l'arabe

4.1. Apprentissage

Pour l'arabe, l'entrainement du TreeTagger a nécessité une procédure en deux étapes :

- Nous avons d'abord défini toutes les données nécessaires pour l'apprentissage (corpus d'apprentissage, classes ouvertes et lexique) en appliquant notre jeu d'étiquettes.
- Ensuite, nous avons exécuté le programme d'apprentissage afin de générer le fichier de paramètres.

Notons bien qu'avant de générer ce fichier résultat, ce programme vérifie la compatibilité de toutes les données en question. Comme nous avons noté au début de ce chapitre, le lexique et le corpus d'apprentissage ont une forme spéciale pour qu'ils soient adaptables sur ce type de programme. Par exemple, la redondance d'un mot et la présence des chiffres dans le lexique donnent lieu à des erreurs. La mise en forme correcte des données nous a pris un certain temps, car la tâche n'était pas des plus aisées. L'application de ce programme d'apprentissage nous a permis de donner le fichier des paramètres qui servent ensuite à l'étiquetage.

4.2. Étiquetage

Après la phase d'apprentissage, TreeTagger peut procéder à l'étiquetage en appelant le deuxième programme « tree-tagger » que nous avons décrit au début de ce chapitre. Préalablement, le texte à analyser doit être translittéré avec le codage de *Buckwalter,* et tokenizé avec notre script de segmentation. Comme nous l'avons noté, TreeTagger a beaucoup de points communs avec les étiqueteurs « n-gramme ». Dans ce type d'approche, on modélise la probabilité d'une séquence de mots d'une manière récursive en se basant sur l'équation suivante :

$$P(w_1 w_2 \dots w_n, t_1 t_2 \dots t_n) = p(t_n | t_{n-2} t_{n-1}) p(w_n | t_n) p(w_1 w_2 \dots w_{n-1}, t_1 t_2 \dots t_{n-1})$$

Avec $w_i : mot\ et\ t_i : tag$

Au contraire d'un étiqueteur « n-gramme », Treetagger fait l'estimation de la probabilité de la transition à partir d'un arbre de décision binaire qui prend comme valeur les étiquettes dans ses nœuds internes et l'étiquette estimé avec leur pourcentage dans les feuilles ainsi l'estimation « oui » ou « non » comme transition. La probabilité d'un trigramme est déterminée par le chemin correspondant à travers l'arbre, depuis la racine jusqu'à la feuille en suivant la transition qui a la valeur « oui ». Si on prend par exemple la séquence trigramme des mots qui ont les étiquettes suivantes « DT JJ NN ». Pour estimer la probabilité d'un nom qui est précédé par un déterminant et un adjectif P(NN / DT,JJ) on suive le chemin valide en commençant de la racine qui contient l'étiquette JJ jusqu'à la feuille qui contient l'étiquette NN avec leur probabilité sachant que l'étiquette $t_{-2} = DT$ (voir l'exemple : figure 3 p 46).

Par défaut, pour un texte en arabe segmenté, TreeTagger donne une liste de tous les mots avec leurs catégories et leur lemme s'il existe. Un paramétrage de TreeTagger permet soit

d'attribuer le lemme « unknown » à toutes les formes inconnus, soit de donner la forme elle-même par défaut dépendamment de lexique.

Par contre, la phase de segmentation est effectuée par notre propre script et n'est pas par les tokenizeurs génériques de TreeTagger. L'exemple suivant représente une phrase étiqueté selon TreeTagger :

```
lwdfYj  NN      lwdfyj
fAn     NNP     fAn
bYthwfnNNP      bythwfn
m&lf    NN      m&lf
mwsYqY  JJ      mwsYqY
>lmAnY  JJ      >lmAnY
wld     VBN     wld
Eam     NN      Eam
1770    CD      Unknow
m       FW      m
fY      IN      fY
mdYnp   NN      mdYnp
bwn     NNP     bwn
.       SENT    .
```

Figure 7 : Exemple de sortie de TreeTagger

En résumé, ce processus d'étiquetage nécessite un enchaînement de plusieurs phases. La figure suivante présente les différents processus (phases) de notre travail :

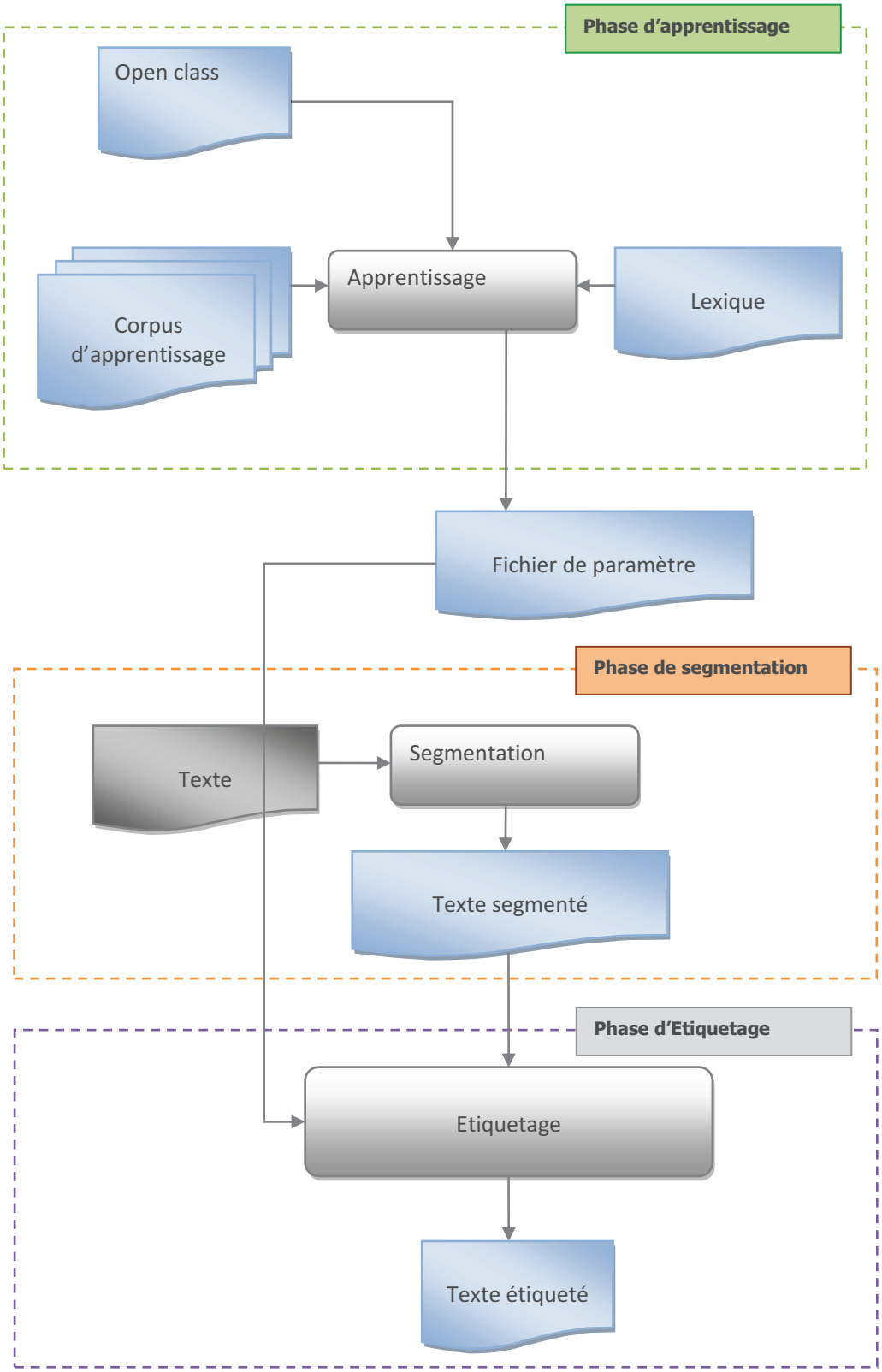

Figure 8 : Les différents processus de l'étiquetage

5. Conclusion

Dans ce chapitre, nous avons présenté le nouvel outil de l'étiquetage pour la langue arabe, en commençant tout d'abord par la présentation de toutes les données nécessaires pour la phase d'entrainement. Ensuite, nous avons présenté le principe d'étiquetage de cet outil en décrivant la méthode utilisée pour l'estimation des étiquettes (arbre de décision) qui nous permettons par la suite de connaître les avantages et les inconvénients de notre systèmes ainsi l'amélioration attendus.

Chapitre 3 : Evaluation et perspectives

L'évaluation d'un système est une étape cruciale de son développement, car elle permet de mettre en évidence les points forts et les limites de ce système, afin de dégager des pistes pour l'améliorer. Elle permet également de comparer ce système avec des approches concurrentes, afin de situer ses performances par rapport à l'état de l'art. En général, l'évaluation d'un système d'étiquetage consiste à comparer la sortie de ce dernier avec un fichier de référence contient les même données de test étiqueté et validé manuellement.

Dans ce chapitre, nous commençons tout d'abord par une brève présentation concernant notre corpus de test. Ce dernier est un fichier contient une centaine des phrases qui servent à évaluer notre système (voir la section prochaine). Pour que les résultats obtenus lors des tests soient valables, il est essentiel que les données du corpus de test est globalement différent de celle de l'apprentissage Ensuite, nous présentons l'outil que nous avons utilisé pour évaluer notre travail ainsi que les différents résultats obtenus.

1. Corpus de test

Comme nous l'avons déjà évoqué précédemment dans cette étude, notre corpus de travail global est divisé en deux sous-corpus (corpus d'apprentissage et corpus de test). Notre corpus de test comporte les 234 premières phrases du corpus du travail, pour un total de 6 029 mots (qui représentent 3 407mots distincts). Par ailleurs, il est constitué d'articles concernant la thématique de l'art. Pour mettre en œuvre l'évaluation, nous avons besoin de réaliser un étiquetage de référence qui contient les phrases de test bien segmentées et avec des étiquettes vérifiées manuellement (voir annexe 7, p 79).

2. Evaluation de notre travail

Un étiqueteur morphosyntaxique est un système complexe constitué de plusieurs modules (segmentation, étiqueteur non contextuel, désambiguïseur) utilisant des ressources variées (lexique, jeu d'étiquettes, corpus, etc). L'évaluation d'un tel système peut donc concerner toute une gamme d'aspects différents.

Comme notre système donne toujours des résultats (il n'est jamais silencieux), nous nous sommes limités au problème de l'évaluation de la précision (complémentaire du « bruit ») de l'étiquetage réalisé par TreeTagger, c'est-à-dire le taux d'étiquetage correct. Cependant, il faut être conscient que ce seul taux ne signifie que peu de chose dans la comparaison entre les systèmes, car la précision de chaque système dépend du mode de segmentation et du jeu d'étiquettes utilisés, ainsi que des données de test utilisées.

Une démarche d'évaluation simple consiste à comparer le fichier résultat de notre étiqueteur appliqué à notre corpus de référence. Pour faire cette comparaison, nous avons essayé d'implémenter un script mais nous avons trouvé une difficulté au niveau de la cohérence entre les deux fichiers. Cette difficulté est située au niveau des différences de segmentation entre le fichier de référence et le fichier de résultat, ce qui nécessite de faire correspondance entre deux flux des caractères parallèle mais segmentés différemment. C'est pourquoi nous avons décidé d'utiliser un outil générique pour l'évaluation des étiqueteurs morphosyntaxique : cet outil très répandu est nommé *Sclite*[15] (plus des détails dans la section prochaine). En effet, pour évaluer notre système d'étiquetage, il faut obligatoirement faire l'alignement entre les deux fichiers en question. L'outil utilisé résoudre bien ce problème.

Pour l'évaluation relative d'un système d'étiquetage, on peut considérer les éléments suivants :

- Une évaluation des types d'ambiguïté pour apprécier la difficulté de l'étiquetage : le nombre moyenne d'étiquettes possibles à assigner à chaque mot, et les types (ou classes) d'ambiguïté, en précisant notamment la fréquence relative dans le corpus test de chacune de ces classes.
- Une évaluation des types d'erreurs : les types d'ambiguïté conduisant le plus fréquemment à des erreurs d'étiquetage, ainsi que le mauvais découpage des mots.
- Une évaluation quantitative de la précision de l'étiquetage.

[15] http://www.itl.nist.gov/iad/mig/tools/

Notons que, Sclite, nous permettons de faire l'évaluation quantitative de notre système en calculant le pourcentage des étiquettes correctes produites grâce à l'alignement entre ses deux fichiers d'entrées.

Dans la section prochaine nous allons présenter une approche pour l'évaluation de notre étiqueteur ainsi l'outil utilisé.

2.1. Présentation de Sclite :

Le programme *Sclite* est un outil dédié à l'évaluation de la production des systèmes de reconnaissance vocale. Il fait partie du *NIST SCTK*[16] qui constitue une boite à outils pour l'évaluation, utilisé lors des compagnes du NIST. Cet outil a déjà été utilisé pour évaluer différents étiqueteurs morphosyntaxiques en adaptant ses entrées/sorties. Les entrées de cet outil sont deux fichiers référence et hypothèse (résultat) numéroté d'une manière identique selon le nombre des phrases sous la forme « mot/ catégorie » Le programme de comparer le fichier résultat de l'étiquetage avec le fichier de référence en mettant en œuvre un processus d'alignement entre ces deux fichiers.
Après que les deux étiquetages résultats et références ont été alignées, ce programme calcule les métriques pour l'évaluation du système. Cet outil présente l'avantage de fournir des options assez riches pour l'affichage des sorties. Par exemple nous avons utilisé l'option « -C » qui permet d'afficher les résultats sous forme de graphe, et l'option « -c »qui permet d'analyser les résultats caractère par caractère.

2.2. Evaluation quantitative :

Sur le plan quantitatif, pour effectuer une évaluation relative de notre étiqueteur, nous avons comparé les résultats obtenus avec ceux d'ASVM1.0 (également appelé AMRA1.0). L'utilisation de Sclite pour évaluer notre système a nécessité une transformation du format de ces sorties. La préparation des deux fichiers (résultat et référence) a été effectué par le script « préparerdonnées » (voir annexe 10, p 89). Tout d'abord, ce script donne aux fichiers le format correcte pour qu'ils soient compatibles avec l'outil en question. Ensuite, il attribue à chaque phrase un numéro qui permet d'aligner les deux fichiers (Exemple : voir annexe 5, p76). Les deux mesures communément utilisées

[16] http://www.icsi.berkeley.edu/Speech/docs/sctk-1.2/sctk.htm

pour évaluer un système d'étiquetage sont le taux de précision P et celui du rappel R. Comme notre système est robuste (il donne des résultats dans tous les cas de figure), nous avons limité l'évaluation au calcule de la précision. Pour ce faire, et après que les textes de référence et les textes d'hypothèse ont été alignés, le système commence son analyse phrase par phrase. Pour chaque phrase de fichier hypothèse compare le couple mot/catégorie à celle des phrases de fichier référence en obtenant à la fin le taux de précision de chaque phrase grâce à cette équation :

$$P = \frac{nombr\ e\ \ des\ couples\ \ corrects\ \ donn\acute{e}\ dans\ \ le\ \ fichier\ \ r\acute{e}sultat}{nombre\ \ total\ \ des\ mots\ \ du\ \ fichier\ \ de\ \ r\acute{e}ference} * 100$$

Afin, de calculer la précision des toutes les phrases, cet outil nous a donné la précision moyenne sur l'ensemble de test noté Pmoy qui est définit par :

$$Pmoy = \frac{\sum_{i=1}^{n} Pi}{n}$$

Avec n, le nombre des phrases de notre corpus (n=234), et Pi : la précision de la phrase i.

Ce mode de calcule a pour effet de minimiser l'impact des erreurs qui apparaissent dans les phrases très longues, car il donne une pondération plus importante aux étiquettes des phrases courtes.

Pour obtenir une évaluation comparative précise, nous avons évalué les deux systèmes sur le même corpus de test. Avec, notre étiqueteur nous avons obtenu un taux de précision moyen égal à 86.5%. Le tableau suivant représente les résultats obtenus sur quelques phrases (voir annexe 9, p 81) :

Phrase	Nombre de mots	nombre de mots bien étiquetés	Nombre de mots mal étiquetés	Précision
1	14	100	0	100,00%
2	17	16	1	94,10%
3	13	13	0	100,00%
4	10	10	0	90,00%
5	21	19	2	90,50
6	16	13	3	81,30%
7	31	28	3	90,30%
8	15	15	0	100,00%
9	9	9	0	100,00%

10	28	28	0	100,00%

Tableau 16 : Résultats pour un échantillon des phrases étiquetées par notre étiqueteur

Pour l'étiqueteur « ASVM1.0 », nous avons obtenu un taux de précision égal à 60% sur les mêmes données. Les résultats sur le même échantillon sont présentés dans le tableau suivant :

Phrase	**Nombre des mots**	**nombre de mots bien étiquetés**	**Nombre de mots mal étiquetés**	**Précision**
1	14	10	4	71,40%
2	17	10	7	59,00%
3	13	5	8	38,50%
4	10	9	1	90 ,00%
5	21	12	9	57,10%
6	16	6	10	37,50%
7	31	15	16	48,40%
8	15	4	11	26,70%
9	9	9	0	100,00%
10	28	17	11	60,70%

Tableau 17 : Résultats pour un échantillon des phrases étiquetés par ASVM1.0

En résumé, pour bien voir la différence entre notre étiqueteur et celle d'ASVM1.0, nous avons mis les résultats de la précision des phrases présentés dans les tableaux précédents dans l'histogramme suivant :

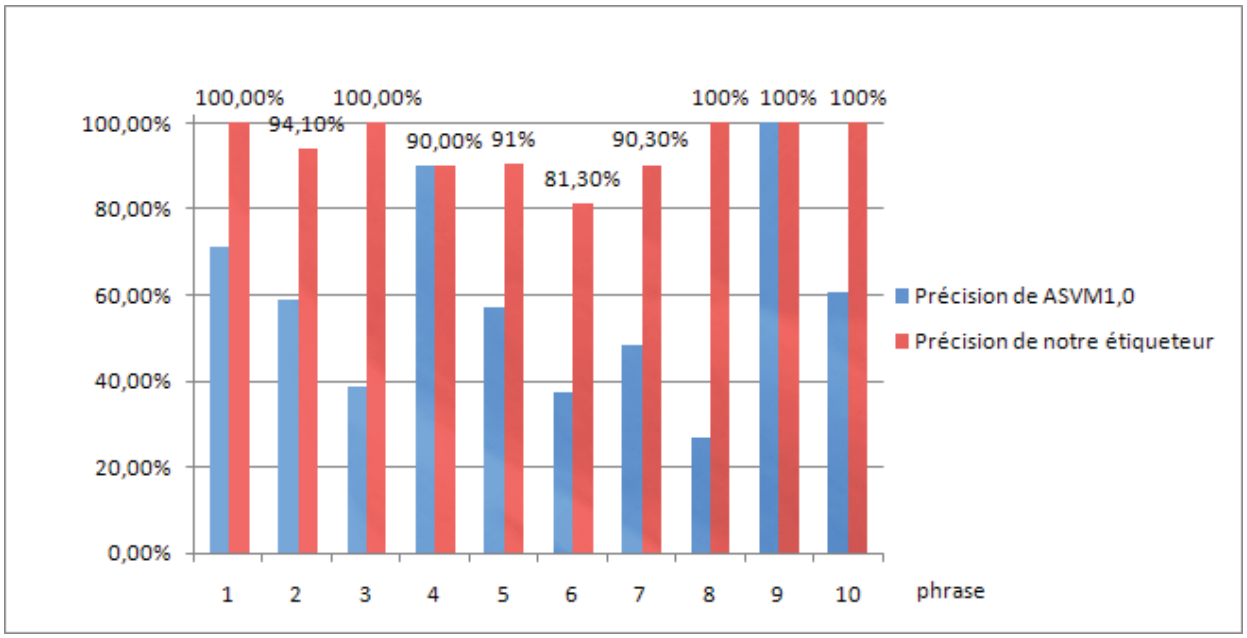

Figure 9 : Histogramme comparatif de notre étiqueteur avec ASVM1.0

2.3. Evaluation qualitative :

Au contraire d'une évaluation quantitative, l'évaluation qualitative se fait manuellement. Par manque de temps, nous n'avons pas pu évaluer manuellement toutes les phrases de notre corpus de test. C'est pourquoi nous avons examiné seulement les 50 premières phrases. Ces phrases contiennent 869 mots étiquetés parmi lesquels nous avons trouvé 24 étiquettes fausses c'est-à-dire 2,75 % d'erreur.

Après avoir analysé manuellement les 50 phrases, nous avons remarqué que les erreurs sont dues, en général à la mauvaise segmentation des mots ou à l'absence des mots dans le lexique ainsi le cas d'ambiguïté. Le tableau suivant illustre les différents cas de figure de ces 24 étiquettes erronés :

Nombre des phrases	Nombre des mots	Nombre d'étiquettes erronés	Nombre des mots mal segmentés	Nombre des mots absent dans le lexique	Nombre des mots ambigus
50	869	24	11	9	4
Pourcentage		2,75	1,26	1.03	0,46

Tableau 18 : Les différents cas de figure des étiquettes erronées sur 50 phrases examinées manuellement

Voici quelques phrases étiquetés par notre étiqueteur :

Phrase1 :

لودفيج فان بيتهوفن مؤلف موسيقي ألماني ولد عام 1770 م في مدينة بون .

lwdfYj / NNP fAn / NNP bYthwfn / NNP m&lf / NN mwsYqY / JJ >lmAnY / JJ wld / VBN Eam / NN 1770 / CD m / NN fY / IN mdYnp / NN bwn / NNP . / SENT

Phrase2 :

يعتبر من أ برز عباقرة الموسيقى في جميع ا لعصور وأبدع، أعمالاً موسيقية خالدة .

Yetbr / VBN mn / IN >brz / JJ EbAqrp / NN Al / DT mwsYqy / NN fY / IN jmYE / JJ Al / DT Eswr / NNS , / PUNC w / CC >bdE / VBN >EmAlAF / NN mwsYqYp / JJ xAldp /JJ . / SENT

Notons bien que, dans ces deux phrases notre étiqueteur donne des résultats sans erreurs (100% des mots correctement étiquetés).

Phrase3 :

لذلك ينصح عادة بأن يتمرن ممن يريد التعلم بالتدريب على إخراج الصوت أولا ومن ثم عندما يستطيع ذلك يبدأ بالتعلم على إخراج الدرجات الصوتية (تمرين الأصابع).

l/IN *lk/WP **YnSH/VBD** EAdp/ b/PREP >n/RP **Ytmrn/VBN** mn/IN **YrYd** /VBN Al/DT **tElm/VBN** b/PREP Al/DT tdrYb/NN Ely/RP <xrAj /NN Al/DT Swt/NN >wlA /JJ w/CC mn/IN vm/RB EndmA/IN YstTYE/VBD *lk/WP **Ybd>/VBP** b/PREP Al/DT **tElm/VBN** Ely/RP <xrAj /NN Al/DT drjAt/NNS Al/DT SwtYp/NN (/PUNC tmrYn /NN Al/DT >SAbE /NNS)/PUNC ./SENT

Dans cette phrase, notre étiqueteur génère deux types d'erreurs à cause de :

- ✓ Mauvaise segmentation qui produit un mauvis étiquetage, les erreurs se présente dans les mots suivantes (YnSH, Ytmrn, YrYd, Ybd>). Notre système attache la préposition « Y » au verbe qui le détermine.
- ✓ Confusion entre le nom et le verbe (tElm). Chaque mot précède un déterminant, doit prendre l'étiquette « NN », mais notre étiqueteur, il le génère comme un verbe.

Phrase4 :

قدم أول عمل موسيقي وعمره 8 سنوات .

qdm / NN >wl / JJ Eml / NN mwsYqY / JJ w / CC Emr / NN h / PRP 8 / CD snwAt / NNS . / SENT

Dans cette phrase, notre étiqueteur génère une erreur au niveau de premier mot à cause de l'ambiguïté (il a donne l'étiquette NN au mot « qdm » au lieu de VBD).

Enfin, pour avoir l'efficacité de notre système nous avons sélectionné une dizaine des phrases plus loin de notre corpus d'apprentissage. Les phrases ont été récupéré du livre « ALMuquadima » (المقدمة) d'Ibn Khaldoun[17]. Ces phrases comportent 488 mots segmentés. Après avoir analysé ces phrases par notre étiqueteur, nous avons obtenu un taux d'erreur de 6,55%.

3. Conclusion et Perspectives

L'évaluation sommaire que nous avons menée indique que notre étiqueteur arabe donne dans la plupart des phrases un étiquetage correct.

Les résultats de l'évaluation quantitative montrent un gain de notre méthode par rapport à l'étiqueteur ASVM1.0. Par contre, nous n'avons pas fait une évaluation qualitative d'ASVM2 sur la totalité de notre corpus de test, puisque ce dernier n'est pas gratuit. Mais, nous avons examiné quelques phrases. Nous avons conclu que l'étiqueteur ASVM2 donne plus de performance par rapport à ASVM1.0 et particulièrement au niveau de segmentation. Néanmoins, nous avons bénéfice des quelques avantage : l'analyse correcte de la plupart des mots qui débute par un article. Cependant, notre étiqueteur génère encore quelques erreurs au niveau de l'étiquetage ou de la segmentation.

Pour améliorer les résultats, l'évaluation qualitative suggère les pistes suivantes :

- ✓ Amélioration de la segmentation : l'étiquetage de la langue arabe reste une tâche difficile à traiter à cause de l'étape de segmentation. En effet, si on arrive à segmenter correctement les textes, l'affectation des étiquettes aux mots devient beaucoup plus facile. C'est pourquoi, nous proposons comme solution les méthodes statistiques basés sur un modèle du langage pour résoudre le cas d'ambiguïté

[17] C'est un historien et philosophe arabe qui a vécu entre 1332 et 1406. C'est lui le fondateur de la sociologie.

- ✓ Amélioration du corpus d'apprentissage : l'étiquetage supervisé doit être essentiellement accompagné par un corpus d'apprentissage bien traité manuellement. C'est pourquoi, pour obtenir des bons résultats sur n'importe quel texte du test, il faut que notre corpus d'apprentissage contienne le maximum des paroles arabes mais plus au moins différents (diminution de la redondance des mots dans le corpus d'apprentissage). Ce pendant, notre corpus doit être très varié : accueillir des phrases de tous les domaines (religion, art, éducative, littéraire….).
- ✓ Amélioration de la couverture du lexique : tout d'abord, le lexique joue un rôle très important dans notre étude pour générer notre fichier de paramètre. Pour cela, notre lexique doit être contenir tous les mots en arabe.
- ✓ Amélioration de lemmatisation : Comme nous avons indiqué précédemment, notre lexique ne contient pas les vrais lemmes des mots. C'est pourquoi notre système ne fait pas correctement cette phase. En effet, et à mon avis il est obligatoirement améliorer ce point. Cependant, il faut ajouter dans notre lexique le vrai lemme de chaque mot. Pour le faire, doit réaliser un lemmatiseur pour la langue arabe, ou d'introduire à la main ces lemmes qui reste une tâche importante concernant le traitement automatique de la langue arabe.

Conclusion générale

Dans cette étude, nous avons adapté l'outil « TreeTagger » pour la langue arabe. Pour ce faire, nous avons organisé notre travail selon trois étapes principales. D'abord, nous avons récolté et préparé toutes les données linguistiques nécessaires : corpus de travail, lexique, jeux d'étiquettes et corpus d'apprentissage. Ensuite, nous avons développé une méthode de segmentation des textes arabes, basée sur l'utilisation de notre corpus. Enfin, nous avons terminé par une évaluation quantitative et qualitative de notre système.

Il nous est apparu que la segmentation correcte de l'arabe, au niveau lexical, est une étape cruciale pour obtenir par la suite un étiquetage de bonne qualité. Nous avons proposé une technique simple de segmentation basée sur un corpus pré-segmenté, en introduisant des listes de pré-bases et post-bases complémentaires.

Au cours de notre recherche, nous avons fait face à certaines difficultés, et accompli un certain nombre de développement, que l'on peut résumer ainsi :

- La tâche principale a été la constitution des ressources arabes. C'est pour cette raison qu'il nous a fallu constituer notre propre corpus de travail en récupérant, d'une part, une collection de textes bruts (sans étiquette ni segmentation) et en appliquant d'autre part, un prétraitement avec ASVM 1.0.

- Il nous a fallu ensuite effectuer le choix de la segmentation convenable pour les mots ambigus, c'est-à-dire les mots qui admettent plusieurs découpages différents. La correction de la segmentation a été effectuée de façon manuelle et semi-automatique, au moyen d'expressions régulières.

- Le corpus ainsi obtenu a ensuite fait l'objet d'une correction manuelle de l'étiquetage relativement chronophage. Ne disposant que d'un temps limité, nous avons dû nous contenter d'un corpus de taille modeste (52 171 mots environ). Cette phase de correction était obligatoire pour un étiquetage supervisé.

- L'identification des lemmes associés au token obtenu reste une tâche difficile, et requiert des ressources lexicales dont nous ne disposions pas. Cette étape n'a donc pas pu être accomplie.

- Enfin, nous n'avons pas pu réaliser un script ad hoc pour faire l'évaluation de notre travail : établir les liens entre deux fichiers parallèles segmentés d'une manière différente nécessite de mettre en œuvre des algorithmes assez complexes. Nous avons cependant pu utiliser des outils génériques pour réaliser cette évaluation.

✔ *Tâche accomplie avec des améliorations significatives*

✘ *Tâche mise de côté lors de cette étude*

Au final, nous avons obtenu de bons résultats au niveau de l'étiquetage, comme l'a montré l'évaluation qualitative et quantitative effectuée sur notre système. C'est pourquoi, on peut conclure que nous avons atteint à peu de frais le niveau des étiqueteurs présentés dans l'état de l'art, et que nous sommes assez proche de fournir à la communauté scientifique un système générique et simple à mettre en œuvre. Il reste cependant à traiter le niveau de la lemmatisation, notre système ne produisant pas les lemmes dans la majorité des cas, du fait de l'absence de ressource lexicale. Pour publier nos paramètres sur le site de TreeTagger, il faudrait essentiellement améliorer le lexique en y ajoutant tous les lemmes.

D'un point de vue général, le traitement automatique de la langue arabe - et en particulier l'étiquetage morphosyntaxique - reste un domaine très ouvert et présente des marges de progression importantes, du fait de la richesse morphologique de cette langue. Comme nous l'avons montré, la segmentation lexicale, une des opérations de base souvent considérée comme triviale dans des langues comme l'anglais ou le français, reste un des problèmes clé de l'arabe, où de grandes améliorations peuvent encore être apportées.

Bibliographie

Abbes, Ramzi. (décembre 2004). *la conception et la réalistion d'un concordancier électronique pour l'arabe.* Thèse de doctorat en sciences de l'information, Lyon,ENSSIB/INSA.

Aljlayl.M and Frieder.O. (2002). On Arabic Search: Improving the Retrieval Effectiveness via a Light Stemmng Approach. *In 11th International Conference on Information and Knowledge Management (CIKM)* , 340-347.

Attia.M. (2006). *An Ambihuity controlled Morphological Analyser for Modern Standard Arabic Modelling Finite State networks.* London: Acte de la conférence internationale 'the challenge of arabic for NLP/MT, the Britsh computer society.

Baloul.S,Alissali.M, Baudry.M and Boula de Mareuil.P. (2002). Interface syntaxe-prosodie dans un système de synthèse de la parole à partir d'une texte en arabe. *24es Journnées d'étude sur la parole* , 329-332.

Brill.Eric. (1993). Tagging an unfamiliar text with minimal human supervision. *in proceedings of the Fall Dymposium on Probabilistic Approch to Natural Language*.

Buckwalter.Tim. (2002). Arabic Morphological Analyser version 1.0 Linguistic Data Consortium. *Catalogue numéro LDC 2002 L49* .

Chalabi.A. (2004). Sakhrt Arabic Lexicon. *Actes de la conférence internationnale NEMLAR, Arabic Language Ressources and Tools le Caire Egypte* .

Dichy, J. (1997). Pour une lexicomatique de l'arabe :l'unité lexicale simple de l'inventaire du mot. *META journal de traduction* , 291-306.

El Jihad A., Yousfi A., Rencontre des Etudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL'2005), « étiquetage automatique des textes arabes par modèle de Markov caché » Durbain (France), 6-10 juin 2005.

Farghaly.A and Dichy.J. (2003). Roots & Patterns VS Stems plus Grammair-Lexis specification :On what basis should a multilingual lexical database centred on arabic be built? *Acte de la 9ème MT conference, Workshop on Machine translation for semitic language :issues and approaches ;New Orleans, Louisiana, USA* .

Habash.N and Rambow.O. (2005). Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambguation in One Fell Swoop. In Proceedings of the Conference of the Association for Computational Linguistics (ACL'05), Ann Arbor, MI.

Haddad. Ahmed, Ben Ghezala. Henda and Ghenima. Malek (2007). Conception d'un catégoriseur morphologique fondé sur le principe d'Eric Brill dans un contexte Multi-Agents. 26 th conference on Lexis and Grammar, Bonifacio.

Jinxi Xu, Alexander Fraser and Ralph.M Weischedel. (August 2002). Emprirical Studies in Strategies for arabic Retrieval. (269-274).

Kadri.Y and Benyamina.A. (1992). Système d'analyse syntaxicosémantique du langage arabe. *mémoire d'ingénieur univercité d'Oran* .

Khoja.S. (2001). APT: Arabic Part-of-speech Tagger. *Workshop NAACL* .

Kiraz.G.A. (1996). Analysis of the Arabic Borken Plural and Diminutive. *In Proceedings of the 5th international conference and Exhibition on Multi-Lingual Computing* .

Laporte.E. (2000). Mot et niveau lexical. *Jean-marie pierre : Ingenierie des langues* , 25-46.

M. El-Haj, U. Kruschwitz, and C. Fox. Using Mechanical Turk to Create a Corpus of Arabic Summaries" in the Language Resources (LRs) and Human Language Technologies (HLT) for Semitic Languages workshop held in conjunction with the 7th International Language Resources and Evaluation Conference (LREC 2010), pages 36-39, Valletta, Malta.

Mars M, Zrigui M, Belgacem M,Zouaghi A, Antoniadis G. "A Semantic Analyzer for the Comprehension of the Spontaneous Arabic Speech", 9th
International Conference on Computing CORE08, Journal Research in Computing Science (Journal RCS), ISSN: 1870-4069, Vol 34, pp 129-140, CORE0, Mexico. 2008.

Mona Diab, Kadri Hacioglu and Daniel Jurafsky. (2004). Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks. *HLT-NAACL* , 149-152.

Mona T.Diab. (2010). Second Generation AMIRA Tools for Arabic processing: Fast and Robust Tokenization, pas Tagging and Base phrase chunking. *Center for Computational Learning Systems Columbia University* , 285-288.

Quinlan, J. (1986). Algorithme de base des arbres des décision: ID3 :" induction of Decision Trees" Machine Learning . *vol.1* , pp 81-106.

Thi Minh. Hu, Laurent. R and Xuan.L. (2003). Une étude de cas pour l'étiquetage morpho-syntaxique de textes vietnamiens. *Batz-sur Mer.*

Thibeault, M. (2004). La catégorisation grammaticale automatique : adaptation du catégoriseur de Brill au Français et modification de l'approche . *Université Laval Québec, Canada* , 11.

Vasilakopoulos, A. (2003). Improved Unknown Word Guessing by Decision Tree Induction. *Dans Proceedings of CLUK. Edinburgh* .

Viterbi. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm . *IEEE Transavtions Information an theory* , pp260-269.

Zemirli.Z and Khabet.S. (Avril 2004). *Un analyseur morphosyntaxique destiné à la synthèse vocale de textes arabes voyellés.* Fès: JEP-TALN 2004.

Zouaghi.A, Zrigui.M and Ben Ahmed.M. (2005). A statistical model for semantic decoding of arabic language.

Zrigui.M, Mili.A and Jemni .M. (1991). vers un système automatique de synthèse de la parole arabe. *Maghrebien symposium on progrramming and system Alger* , 180-197.

Annexe

Annexe1: Alphabets arabes

أ
ب
ت
ث
ج
ح
خ
د
ذ
ر
ز
س
ش
ص
ض
ط
ظ
ع
غ
ف
ق
ك
ل
م
ن
ه
و
ي

Annexe2 : Table de translitération de Buckwalter :

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

Annexe3 : Listes des pré-bases (Extrait) :

Pré-bases	Translitération en Buckwalter	Dimension
أ	>	1
ت	T	1
ي	Y	1
أن	>n	2
أت	>t	2
أي	>Y	2
و	w	1
وأ	w>	2
ون	wn	2
وت	wt	2
ف	f	1
فأ	f>	2
فت	Ft	2
فن	Fn	2
في	fY	2
ل	l	1
لأ	lA	2
س	s	1
أست	>st	3
أسي	>sY	3
ب	B	1
أل	Al	2
لأل	lAl	3
أبأل	>bAl	4

Annexe4 : Listes des post-bases (Extrait) :

Post-base	Transliteration Buckwalter	Dimension
ني	nY	2
نا	nA	2
ك	k	1
كما	kmA	3
كم	km	2
كن	kn	2
ه	h	1
ها	hA	2
هما	hmA	3
هم	hm	2
هن	hn	2
انهما	AnhmA	5
انهم	Anhm	4
ون	wn	2
ننا	nnA	3
وا	wA	2
واهما	wAhmA	5
ونهما	wnhmA	5
ات	At	2
ونن	Wnn	3
يه	Yh	2
ينا	YnA	3
يها	YhA	3
تموهما	tmwhmA	6
يننا	YnnA	4
تهما	thmA	4
اه	Ah	2
اها	AhA	3
انه	Anh	3
نك	nk	2

Annexe5 : Alignement du corpus de référence selon « Sclite » (extrait) :

```
lwdfYj/NNP fAn/NNP bYthwfn/NNP m&lf/NN mwsYqY/JJ >lmAnY/JJ wld/VBN EAm/NN
1770/CD m/NN fY/IN mdYnp/NN bwn/NNP ./SENT (1)
 YEtbr/VBN mn/IN >brz/JJ EbAqrp/NN Al/DT mwsYqy/NN fY/IN jmYE/JJ Al/DT
ESwr/NNS ,/PUNC w/CC >bdE/VBD >EmAlAF/NN mwsYqYp/JJ xAldp/JJ ./SENT (2)
 l/IN h/PRP Al/DT fDl/NN Al/DT >EZm/JJ fY/IN tTwYr/NN Al/DT mwsYqy/NN
Al/DT klAsYkYp/JJ ./SENT (3)
 qdm/VBD >wl/JJ Eml/NN mwsYqY/JJ w/CC Emr/NN h/PRP 8/CD snwAt/NNS ./SENT
(4)
 t$ml/VBP m&lfAt/NNS h/PRP l/IN Al/DT >wrkstrA/NN tsEp/CD sYmfwnYAt/NNS
w/CC xms/CD mqTwEAt/NNS mwsYqYp/JJ Ely/IN Al/DT bYAnw/NN w/CC mqTwEp/NN
Ely/IN Al/DT kmAn/NNP ./SENT (5)
 k/IN mA/WP >l~f/VBD Al/DT EdYd/RP mn/IN Al/DT mqTwEAt/NNS Al/DT
mwsYqYp/JJ k/IN mqdmAt/NNS l/IN Al/DT >wbrA/NN ./SENT (6)
 bd>/VBD bYthwfn/NNP Yfqd/VBP smE/NN h/PRP fY/IN Al/DT vlAvYnYAt/JJ mn/IN
Emr/NN h/PRP <lA/RP >n/RP *lk/WP lm/RP Y&vr/VBP Ely/IN <ntAj/NN h/PRP
Al*Y/WP AzdAd/VBD fY/IN tlk/RB Al/DT ftrp/NN w/CC tmYz/VBP b/PREP Al/DT
>bdAE/NN ./SENT (7)
 mn/IN >jml/JJ >EmAl/NNS h/PRP Al/DT smfwnYp/JJ Al/DT xAmsp/JJ w/CC Al/DT
sAdsp/JJ w/CC Al/DT tAsEp/CD ./SENT (8)
 w/CC qd/RP twfY/VBN fY/IN fYYnA/NNP EAm/NN 1827/CD m/NN ./SENT (9)
 $hdt/VBD mdYnp/NN bwn/NNP Al/DT >lmAnYp/NNP mYlAd/NN Al/DT fnAn/NN Al/DT
EbqrY/JJ lwdfj/NNP fAn/NNP bYthwfn/NNP fY/IN 16/CD dYsmbr/NNP EAm/NN
1770/CD ,/PUNC w/CC tm/VBD tEmYd/NN h/PRP fY/IN 17/CD dYsmbr/NNP 1770/CD
./SENT (10)
 Zhr/VBN tmYz/VBP h/PRP Al/DT mwsYqY/JJ mn*/RP Sgr/JJ h/PRP ,/PUNC f/CC
n$rt/VBD >wly/JJ >EmAl/NNS h/PRP w/CC hw/PRP fY/IN Al/DT vAnYp/CD E$r/CD
mn/IN Emr/NN h/PRP EAm/NN 1783/CD m/NN ./SENT (11)
 AtsEt/VBP $hrt/NN h/PRP k/IN EAzf/NN bYAnw/NN fY/IN sn/NN mbkrp/NN
,/PUNC vm/RB zAd/VBP <ntAj/NN h/PRP w/CC *AE/VBN SYt/NN h/PRP k/IN
m&lf/NN mwsYqy/NN ./SENT (12)
 EAny/VBP bYthwfn/NNP kvYrAF/JJ fY/IN HYAt/NN h/PRP ,/PUNC EA}lYAF/JJ
w/CC kvYrAF/JJ ,/PUNC f/CC b/PREP Al/DT rgm/RB mn/IN >n/RP >bA/NN h/PRP
hw/PRP mElm/NN h/PRP Al/DT >wl/JJ Al*Y/WP wjh/VBD AhtmAm/NN h/PRP l/IN
Al/DT mwsYqy/NN w/CC lqn/VBN h/PRP Al/DT Ezf/NN Ely/IN Al/DT bYAnw/NN
w/CC Al/DT kmAn/NNP ,/PUNC <lA/RP >n/RP h/PRP lm/RP Ykn/VBP Al/DT >b/NN
Al/DT mYvAlY/JJ ,/PUNC f/CC qd/RP kAn/VBD mdmnAF/JJ l/IN Al/DT kHwl/NN
,/PUNC k/IN mA/WP >n/RP wAldt/NN h/PRP twfYt/VBN w/CC hw/PRP fY/IN Al/DT
sAbEp/JJ E$r/CD mn/IN Emr/NN h/PRP bEd/RP SrAE/NN TwYl/JJ mE/IN Al/DT
mrD/NN ,/PUNC tArkp/NN l/IN h/PRP ms&wlYp/NN Al/DT EA}lp/NN ./SENT (13)
 mmA/IN mnE/VBD h/PRP mn/IN <tmAm/NN xTt/NN h/PRP w/CC Al/DT sfr/NN
<ly/IN fYYnA/NNP ,/PUNC EASmp/NN Al/DT mwsYqy/NN fY/IN *lk/WP Al/DT
ESr/NN ./SENT (14)
 f/CC hl/RP kAn/VBD Al/DT t>lYf/NN Al/DT mwsYqY/JJ hw/PRP nwE/NN mn/IN
>nwAE/NNS Al/DT ElAj/NN w/CC Al/DT tglb/NN Ely/IN Al/DT m$Akl/NNS b/PREP
Al/DT nsbp/NN l/IN bYthwfn/NNP ./SENT (15)
 fY/IN 1789/CD m/NN tHqq/VBD Hlm/NN h/PRP >xYrAF/JJ ,/PUNC f/CC qd/RP
>rsl/VBD h/PRP HAkm/NN bwn/NNP <ly/IN fYYnA/NNP ,/PUNC w/CC hnAk/RB
ttlm*/VBP Ely/IN Yd/NN hAYdn/NNP ./SENT (16)
 w/CC lkn/RP bYthwfn/NNP ,/PUNC SAHb/NN Al/DT >lHAn/NNS wAjh/VBD bED/RP
Al/DT xlAfAt/NNS mE/IN mElm/NN h/PRP ,/PUNC w/CC EndmA/IN sAfr/VBD
hAYdn/NNP <ly/IN lndn/NNP ,/PUNC tHwl/VBD bYthwfn/NNP <ly/IN mElmYn/NNS
|xrYn/JJ mvl/RP sAlYrY/NNP w/CC $Ynk/NNP w/CC >lbrY$tbYrjr/NNP ./SENT
(17)
 w/CC qd/RP >shmt/VBD kl/RP h*h/WP Al/DT drws/NNS w/CC Al/DT AHtkAkAt/NNS
fY/IN tkwYn/NN $xSYp/NN bYthwfn/NNP Al/DT fnYp/JJ ./SENT (18)
```

```
 w/CC HAwl/VBD >n/RP Y$q/VBN l/IN nfs/NN h/PRP TrYq/NN k/IN EAzf/NN fY/IN
EASmp/NN Al/DT mwsYqy/NN ,/PUNC w/CC srEAn/JJ mA/WP lAqy/VBD mkAnt/NN
kbry/JJ xASp/RB fY/IN Al/DT >wsAT/NNS Al/DT >rstqrATYP/JJ ./SENT (19)
 f/CC qd/RP HAz/VBD b/PREP <EjAb/NN Al/DT >srp/NN Al/DT mlkYp/JJ w/CC
Ewml/VBP k/IN SdYq/NN >kvr/JJ mn/IN h/PRP m&lfAF/JJ ./SENT (20)
 b/PREP Al/DT rgm/RB mn/IN *lk/WP f/CC qd/RP EA$/VBD w/CC mAt/VBD
fqYrAF/JJ ,/PUNC gnA/NN h/PRP hw/PRP >EmAl/NNS h/PRP Al/DT fnYp/JJ Al/DT
mtmYzp/JJ ./SENT (21)
 f/CC qd/RP jA'/VBD <ntAj/NN h/PRP Al/DT fnY/JJ gzYrAF/JJ Hty/RP bEd/RP
<SAbt/NN h/PRP b/PREP Al/DT Sm/NN ./SENT (22)
 bd>t/VBD <SAbp/NN bYthwfn/NNP b/PREP Sm/NN bsYT/JJ EAm/NN 1802/CD ,/PUNC
f/CC bd>/VBD fY/IN Al/DT AnsHAb/NN mn/IN Al/DT >wsAT/NNS Al/DT fnYp/JJ
tdrYjYAF/JJ ,/PUNC w/CC >mDy/VBN HYAt/NN h/PRP b/PREP lA/RP zwAj/NN
YrtbT/VBP b/PREP ElAqAt/NNS Edp/RB mE/IN sYdAt/NNS SgYrAt/JJ ./SENT (23)
 <lA/RP >n/RP h/PRP lm/RP Ytwqf/VBP En/IN Al/DT <ntAj/NN Al/DT fnY/JJ
,/PUNC w/CC lkn/RP >EmAl/NNS h/PRP Atx*t/VBD AtjAh/NN jdYd/JJ ./SENT (24)
 w/CC mE/IN AzdYAd/NN HAlp/NN Al/DT Sm/NN AltY/WP >SAbt/VBD h/PRP ,/PUNC
AmtnE/VBP En/IN Al/DT Ezf/NN fY/IN Al/DT HflAt/NNS Al/DT EAmp/JJ ,/PUNC
w/CC AbtEd/VBP En/IN Al/DT HYAp/NN Al/DT AjtmAEYp/JJ w/CC Atjh/VBD l/IN
Al/DT wHdp/NN ,/PUNC w/CC qlt/VBD m&lfAt/NNS h/PRP ,/PUNC w/CC >SbHt/VBD
>kvr/JJ tEqYdAF/JJ ./SENT (25)
 Hty/RP >n/RP h/PRP rd/VBD Ely/IN AntqAdAt/NNS nqAd/NNS h/PRP b/PREP
>n/RP h/PRP YEzf/VBD l/IN Al/DT >jYAl/NNS Al/DT qAdmp/JJ ./SENT (26)
 w/CC b/PREP Al/DT fEl/NN mAzAlt/RP >EmAl/NNS h/PRP Hty/RP Al/DT Ywm/NN
mn/IN >hm/JJ mA/WP >ntjt/VBD h/PRP Al/DT mwsYqy/NN Al/DT klAsYkYp/JJ
Al/DT EAlmYp/JJ ./SENT (27)
 w/CC Aktsbt/VBD AvnAn/CD mn/IN Al/DT sYmfwnYAt/NNS AltY/WP ktb/VBD
hA/PRP fY/IN Sm/NN h/PRP >kbr/JJ $EbYp/NN ,/PUNC w/CC hmA/PRP$ Al/DT
sYmfwnYp/NN Al/DT xAmsp/JJ w/CC Al/DT tAsEp/CD ./SENT (28)
 k/IN mA/WP >n/RP h/PRP >Hdv/VBD Al/DT kvYr/JJ mn/IN Al/DT tgYYrAt/NNS
fY/IN Al/DT mwsYqy/NN ,/PUNC w/CC >dxl/VBD Al/DT gnA'/NN w/CC Al/DT
klmAt/NNS fY/IN sYmfwnYt/NN h/PRP Al/DT tAsEp/CD ./SENT (29)
 f/CC jA't/VBD rsAlt/NN h/PRP <ly/IN Al/DT EAlm/NN kl/RP Al/DT b$r/NN
s/PREP YSbH/VBN wn/PRP$ <xwp/NN ./SENT (30)
 Al/DT nAY/NN |lp/NN nfxYp/NN tEd/VBP b/PREP Hq/NN >qdm/JJ |lp/NN
mwsYqYp/JJ fY/IN Al/DT tArYx/NN (/PUNC <*A/RP AstvnY/VBD nA/PRP Al/DT
|lAt/NNS Al/DT <YqAEYp/JJ (/PUNC w/CC l/IN Al/DT nAY/NN Edp/RB >smA'/NNS
tErf/VBP b/PREP hA/PRP mn/IN hA/PRP Al/DT nAY/NN Al/DT qSbp/NN Al/DT
$bAbp/NN Al/DT mnjYrp/JJ ./SENT (31)
 w/CC Al/DT nAY/NN klmp/NN fArsYp/JJ tEnY/VBP Al/DT mzmAr/NN ./SENT (32)
```

Annexe6 : Corpus d'apprentissage (extrait):

```
w       CC
ntYjp   NN
l       IN
*lk     WP
f       IN
<n      RP
hnAk    RB
ArtbAT  NN
tArYxY  JJ
bYn     RB
Al      DT
bhA}Yp  NN
w       CC
Al      DT
bAbYp   NN
.       SENT
w       CC
bEd     RP
>n      RP
$AE     VBD
>mr     NN
Al      DT
bAbYp   NN
qAmt    VBD
Al      DT
slTAt   NNS
Al      DT
<YrAnYpJJ
,       PUNC
b       PREP
<YEAz   NN
mn      IN
rjAl    NNS
Al      DT
dYn     NN
,       PUNC
b       PREP
tE*Yb   NN
AlbAb   NN
Yn      PRP
w       CC
Al      DT
qbD     NN
Ely     RP
AlbAb   NNP
snp     NN
1847    CD
m       NN
w       CC
<YdAE   VBP
h       PRP$
Al      DT
sjn     NN
.       SENT
w       CC
kAnt    VBD
<YrAn   NNP
mHkwmp  NN
```

```
An*Ak   RB
mn      IN
qbl     RP
Asrp    NN
AlqAjArNNP
AltrkmAnYp      NNP
.       SENT
```

Annexe7 : Corpus de référence (extrait):

```
lwdfYj  NNP
fAn     NNP
bYthwfnNNP
m&lf    NN
mwsYqY  JJ
>lmAnY  JJ
wld     VBN
EAm     NN
1770    CD
m       NN
fY      IN
mdYnp   NN
bwn     NNP
.       SENT
YEtbr   VBN
mn      IN
>brz    JJ
EbAqrp  NN
Al      DT
mwsYqy  NN
fY      IN
jmYE    JJ
Al      DT
ESwr    NNS
,       PUNC
w       CC
>bdE    VBD
>EmAlAFNN
mwsYqYpJJ
xAldp   JJ
.       SENT
l       IN
h       PRP
Al      DT
fDl     NN
Al      DT
>EZm    JJ
fY      IN
tTwYr   NN
Al      DT
mwsYqy  NN
Al      DT
klAsYkYp        JJ
.       SENT
qdm     VBD
>wl     JJ
Eml     NN
mwsYqY  JJ
w       CC
Emr     NN
```

```
h       PRP
8       CD
snwAt   NNS
.       SENT
t$ml    VBP
m&lfAt  NNS
h       PRP
l       IN
Al      DT
>wrkstrA        NN
tsEp    CD
sYmfwnYAt       NNS
w       CC
xms     CD
mqTwEAtNNS
mwsYqYpJJ
Ely     IN
Al      DT
bYAnw   NN
w       CC
mqTwEp  NN
Ely     IN
Al      DT
kmAn    NNP
.       SENT
```

Annexe8 : Corpus du travail (extrait):

```
t$ml    VBP
m&lfAt  NNS
h       PRP$
l       IN
Al      DT
>wrkstrA        NN
Ts_Ep   CD
sYmfwnYAt       NNS
w       CC
xms     CD
mqTwEAtNNS
mwsYqYpJJ
Ely     RP
Al      DT
bYAnw   NN
w       CC
mqTwEp  NN
Ely     RP
Al      DT
kmAn    NNP
.       SENT
kmA     IN
>l~f    VBD
Al      DT
EdYd    RP
mn      IN
Al      DT
mqTwEAtNNS
Al      DT
mwsYqYpJJ
k       PRP$
mqdmAt  NNS
```

```
l       IN
Al      DT
>wbrA   NN
.       SENT
bd>     VBD
bYthwfnNNP
Yfqd    VBN
smE     NN
h       PRP$
fY      IN
Al      DT
vlAvYnYAt       JJ
mn      IN
Emr     NN
h       PRP$
<lA     RP
>n      RP
*lk     WP
lm      RP
Y&vr    VBP
Ely     RP
<ntAj   NN
h       PRP$
Al*Y    RP
AzdAd   VBD
fY      IN
tlk     RB
Al      DT
ftrp    NN
w       CC
tmYz    VBP
b       PREP
Al      DT
>bdAE   NN
.       SENT
```

Annexe9 : résultat des échantillons de notre étiqueteur selon Sclite :

resultat.txt_4.txt								
SPKR	# Snt	# Wrd	Corr	Sub	Del	Ins	Err	S.Err
1)	1	14	100.0	0.0	0.0	0.0	0.0	0.0
2)	1	17	94.1	5.9	0.0	0.0	5.9	100.0
3)	1	13	100.0	0.0	0.0	0.0	0.0	0.0
4)	1	10	90.0	10.0	0.0	0.0	10.0	100.0
5)	1	21	90.5	9.5	0.0	0.0	9.5	100.0
6)	1	16	81.3	18.8	0.0	0.0	18.8	100.0
7)	1	31	90.3	9.7	0.0	0.0	9.7	100.0

8)	1	15	100.0	0.0	0.0	0.0	0.0	0.0
9)	1	9	100.0	0.0	0.0	0.0	0.0	0.0
10)	1	28	100.0	0.0	0.0	0.0	0.0	0.0
11)	1	27	100.0	0.0	0.0	0.0	0.0	0.0
12)	1	22	90.9	9.1	0.0	0.0	9.1	100.0
13)	1	90	94.4	5.6	0.0	0.0	5.6	100.0
14)	1	21	100.0	0.0	0.0	0.0	0.0	0.0
15)	1	25	92.0	8.0	0.0	0.0	8.0	100.0
16)	1	24	95.8	4.2	0.0	0.0	4.2	100.0
17)	1	34	97.1	2.9	0.0	0.0	2.9	100.0
18)	1	17	100.0	0.0	0.0	0.0	0.0	0.0
19)	1	28	92.9	7.1	0.0	0.0	7.1	100.0
20)	1	18	94.4	5.6	0.0	0.0	5.6	100.0
21)	1	22	90.9	9.1	0.0	0.0	9.1	100.0
22)	1	16	100.0	0.0	0.0	0.0	0.0	0.0
23)	1	36	100.0	0.0	0.0	0.0	0.0	0.0
24)	1	19	100.0	0.0	0.0	0.0	0.0	0.0
25)	1	43	95.3	4.7	0.0	0.0	4.7	100.0
26)	1	18	94.4	5.6	0.0	0.0	5.6	100.0
27)	1	22	90.9	9.1	0.0	0.0	9.1	100.0
28)	1	25	92.0	8.0	0.0	0.0	8.0	100.0
29)	1	27	92.6	7.4	0.0	0.0	7.4	100.0
30)	1	15	100.0	0.0	0.0	0.0	0.0	0.0
31)	1	42	97.6	2.4	0.0	0.0	2.4	100.0
32)	1	9	88.9	11.1	0.0	0.0	11.1	100.0
33)	1	53	96.2	3.8	0.0	0.0	3.8	100.0
34)	1	21	100.0	0.0	0.0	0.0	0.0	0.0
35)	1	35	100.0	0.0	0.0	0.0	0.0	0.0
36)	1	23	95.7	4.3	0.0	0.0	4.3	100.0
37)	1	14	100.0	0.0	0.0	0.0	0.0	0.0
38)	1	34	94.1	5.9	0.0	0.0	5.9	100.0

39)	1	41	95.1	4.9	0.0	0.0	4.9	100.0
40)	1	67	98.5	1.5	0.0	0.0	1.5	100.0
41)	1	21	100.0	0.0	0.0	0.0	0.0	0.0
42)	1	36	100.0	0.0	0.0	0.0	0.0	0.0
43)	1	25	100.0	0.0	0.0	0.0	0.0	0.0
44)	1	41	97.6	2.4	0.0	0.0	2.4	100.0
45)	1	37	91.9	8.1	0.0	0.0	8.1	100.0
46)	1	58	91.4	8.6	0.0	0.0	8.6	100.0
47)	1	20	95.0	5.0	0.0	0.0	5.0	100.0
48)	1	13	92.3	7.7	0.0	0.0	7.7	100.0
49)	1	39	94.9	5.1	0.0	0.0	5.1	100.0
50)	1	19	94.7	5.3	0.0	0.0	5.3	100.0
51)	1	158	98.1	1.9	0.0	0.0	1.9	100.0
52)	1	31	96.8	3.2	0.0	0.0	3.2	100.0
53)	1	26	100.0	0.0	0.0	0.0	0.0	0.0
54)	1	46	89.1	10.9	0.0	0.0	10.9	100.0
55)	1	71	91.5	8.5	0.0	0.0	8.5	100.0
56)	1	34	97.1	2.9	0.0	0.0	2.9	100.0
57)	1	24	100.0	0.0	0.0	0.0	0.0	0.0
58)	1	37	97.3	2.7	0.0	0.0	2.7	100.0
59)	1	179	98.3	1.7	0.0	0.0	1.7	100.0
60)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
61)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
62)	1	39	97.4	2.6	0.0	0.0	2.6	100.0
63)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
64)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
65)	1	146	95.2	4.1	0.7	0.0	4.8	100.0
66)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
67)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
68)	1	99	96.0	4.0	0.0	0.0	4.0	100.0

69)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
70)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
71)	1	22	95.5	4.5	0.0	0.0	4.5	100.0
72)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
73)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
74)	1	69	97.1	2.9	0.0	0.0	2.9	100.0
75)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
76)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
77)	1	48	91.7	8.3	0.0	0.0	8.3	100.0
78)	1	64	96.9	3.1	0.0	0.0	3.1	100.0
79)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
80)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
81)	1	25	96.0	4.0	0.0	0.0	4.0	100.0
82)	1	69	92.8	7.2	0.0	0.0	7.2	100.0
83)	1	62	93.5	6.5	0.0	0.0	6.5	100.0
84)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
85)	1	1	100.0	0.0	0.0	0.0	0.0	0.0
86)	1	86	95.3	4.7	0.0	0.0	4.7	100.0
87)	1	113	93.8	6.2	0.0	0.0	6.2	100.0
88)	1	86	90.7	9.3	0.0	0.0	9.3	100.0
89)	1	86	91.9	8.1	0.0	0.0	8.1	100.0
90)	1	146	94.5	5.5	0.0	0.0	5.5	100.0
91)	1	141	94.3	5.7	0.0	0.0	5.7	100.0
92)	1	89	97.8	2.2	0.0	0.0	2.2	100.0
93)	1	21	90.5	9.5	0.0	0.0	9.5	100.0
94)	1	140	91.4	8.6	0.0	0.0	8.6	100.0
95)	1	97	92.8	7.2	0.0	0.0	7.2	100.0
96)	1	111	91.9	8.1	0.0	0.0	8.1	100.0
97)	1	71	95.8	4.2	0.0	0.0	4.2	100.0
98)	1	104	95.2	4.8	0.0	0.0	4.8	100.0
99)	1	105	98.1	1.9	0.0	0.0	1.9	100.0

100	1	51	90.2	9.8	0.0	0.0	9.8	100.0
101	1	53	94.3	5.7	0.0	0.0	5.7	100.0
102	1	46	93.5	6.5	0.0	0.0	6.5	100.0
103	1	1	100.0	0.0	0.0	0.0	0.0	0.0
104	1	27	88.9	11.1	0.0	0.0	11.1	100.0
105	1	34	94.1	5.9	0.0	0.0	5.9	100.0
106	1	35	97.1	2.9	0.0	0.0	2.9	100.0
107	1	2	100.0	0.0	0.0	0.0	0.0	0.0
108	1	8	100.0	0.0	0.0	0.0	0.0	0.0
109	1	2	100.0	0.0	0.0	0.0	0.0	0.0
110	1	11	81.8	18.2	0.0	0.0	18.2	100.0
111	1	20	100.0	0.0	0.0	0.0	0.0	0.0
112	1	32	93.8	6.3	0.0	0.0	6.3	100.0
113	1	16	100.0	0.0	0.0	0.0	0.0	0.0
114	1	30	96.7	3.3	0.0	0.0	3.3	100.0
115	1	31	96.8	3.2	0.0	0.0	3.2	100.0
116	1	32	100.0	0.0	0.0	0.0	0.0	0.0
117	1	28	96.4	3.6	0.0	0.0	3.6	100.0
118	1	14	92.9	7.1	0.0	0.0	7.1	100.0
119	1	25	84.0	16.0	0.0	0.0	16.0	100.0
120	1	13	92.3	7.7	0.0	0.0	7.7	100.0
121	1	39	94.9	5.1	0.0	0.0	5.1	100.0
122	1	60	90.0	10.0	0.0	0.0	10.0	100.0
123	1	9	100.0	0.0	0.0	0.0	0.0	0.0
124	1	15	86.7	13.3	0.0	0.0	13.3	100.0
125	1	8	87.5	12.5	0.0	0.0	12.5	100.0
126	1	26	96.2	3.8	0.0	0.0	3.8	100.0
127	1	14	92.9	7.1	0.0	0.0	7.1	100.0
128	1	18	94.4	5.6	0.0	0.0	5.6	100.0
129	1	68	82.4	17.6	0.0	0.0	17.6	100.0

130	1	29	100.0	0.0	0.0	0.0	0.0	0.0
131	1	20	85.0	15.0	0.0	0.0	15.0	100.0
132	1	67	94.0	6.0	0.0	0.0	6.0	100.0
133	1	1	100.0	0.0	0.0	0.0	0.0	0.0
134	1	52	94.2	5.8	0.0	0.0	5.8	100.0
135	1	8	75.0	25.0	0.0	0.0	25.0	100.0
136	1	19	89.5	10.5	0.0	0.0	10.5	100.0
137	1	29	89.7	10.3	0.0	0.0	10.3	100.0
138	1	37	94.6	5.4	0.0	0.0	5.4	100.0
139	1	53	79.2	20.8	0.0	0.0	20.8	100.0
140	1	34	79.4	20.6	0.0	0.0	20.6	100.0
141	1	45	84.4	15.6	0.0	0.0	15.6	100.0
142	1	23	69.6	30.4	0.0	0.0	30.4	100.0
143	1	32	93.8	6.3	0.0	0.0	6.3	100.0
144	1	49	91.8	8.2	0.0	0.0	8.2	100.0
145	1	25	84.0	16.0	0.0	0.0	16.0	100.0
146	1	37	89.2	10.8	0.0	0.0	10.8	100.0
147	1	86	80.2	19.8	0.0	0.0	19.8	100.0
148	1	27	96.3	3.7	0.0	0.0	3.7	100.0
149	1	22	90.9	9.1	0.0	0.0	9.1	100.0
150	1	35	94.3	5.7	0.0	0.0	5.7	100.0
151	1	24	91.7	8.3	0.0	0.0	8.3	100.0
152	1	87	88.5	11.5	0.0	0.0	11.5	100.0
153	1	137	76.6	23.4	0.0	0.0	23.4	100.0
154	1	55	85.5	14.5	0.0	0.0	14.5	100.0
155	1	68	88.2	11.8	0.0	0.0	11.8	100.0
156	1	22	95.5	4.5	0.0	0.0	4.5	100.0
157	1	47	83.0	17.0	0.0	0.0	17.0	100.0
158	1	26	84.6	15.4	0.0	0.0	15.4	100.0
159	1	32	81.3	18.8	0.0	0.0	18.8	100.0
160	1	77	76.6	23.4	0.0	0.0	23.4	100.0

161	1	40	75.0	25.0	0.0	0.0	25.0	100.0
162	1	46	73.9	26.1	0.0	0.0	26.1	100.0
163	1	25	88.0	12.0	0.0	0.0	12.0	100.0
164	1	18	72.2	27.8	0.0	0.0	27.8	100.0
165	1	4	100.0	0.0	0.0	0.0	0.0	0.0
166	1	36	69.4	30.6	0.0	0.0	30.6	100.0
167	1	40	70.0	30.0	0.0	0.0	30.0	100.0
168	1	58	77.6	22.4	0.0	0.0	22.4	100.0
169	1	406	74.4	25.6	0.0	0.0	25.6	100.0
170	1	106	83.0	17.0	0.0	0.0	17.0	100.0
171	1	30	76.7	23.3	0.0	0.0	23.3	100.0
172	1	44	72.7	27.3	0.0	0.0	27.3	100.0
173	1	84	67.9	32.1	0.0	0.0	32.1	100.0
174	1	49	73.5	26.5	0.0	0.0	26.5	100.0
175	1	62	82.3	17.7	0.0	0.0	17.7	100.0
176	1	32	78.1	21.9	0.0	0.0	21.9	100.0
177	1	54	88.9	11.1	0.0	0.0	11.1	100.0
178	1	70	70.0	30.0	0.0	0.0	30.0	100.0
179	1	88	67.0	33.0	0.0	0.0	33.0	100.0
180	1	206	62.1	37.9	0.0	0.0	37.9	100.0
181	1	19	63.2	36.8	0.0	0.0	36.8	100.0
182	1	105	74.3	25.7	0.0	0.0	25.7	100.0
183	1	27	70.4	29.6	0.0	0.0	29.6	100.0
184	1	93	71.0	29.0	0.0	0.0	29.0	100.0
185	1	42	85.7	14.3	0.0	0.0	14.3	100.0
186	1	40	77.5	22.5	0.0	0.0	22.5	100.0
187	1	38	86.8	13.2	0.0	0.0	13.2	100.0
188	1	16	75.0	25.0	0.0	0.0	25.0	100.0
189	1	84	60.7	39.3	0.0	0.0	39.3	100.0
190	1	32	71.9	28.1	0.0	0.0	28.1	100.0

191	1	30	86.7	13.3	0.0	0.0	13.3	100.0
192	1	34	85.3	14.7	0.0	0.0	14.7	100.0
193	1	21	90.5	9.5	0.0	0.0	9.5	100.0
194	1	26	88.5	11.5	0.0	0.0	11.5	100.0
195	1	42	81.0	19.0	0.0	0.0	19.0	100.0
196	1	44	79.5	20.5	0.0	0.0	20.5	100.0
197	1	31	80.6	19.4	0.0	0.0	19.4	100.0
198	1	29	82.8	17.2	0.0	0.0	17.2	100.0
199	1	53	86.8	13.2	0.0	0.0	13.2	100.0
200	1	35	77.1	22.9	0.0	0.0	22.9	100.0
201	1	90	73.3	26.7	0.0	0.0	26.7	100.0
202	1	24	41.7	58.3	0.0	0.0	58.3	100.0
203	1	44	65.9	34.1	0.0	0.0	34.1	100.0
204	1	19	73.7	26.3	0.0	0.0	26.3	100.0
205	1	32	84.4	15.6	0.0	0.0	15.6	100.0
206	1	14	92.9	7.1	0.0	0.0	7.1	100.0
207	1	27	81.5	18.5	0.0	0.0	18.5	100.0
208	1	26	76.9	23.1	0.0	0.0	23.1	100.0
209	1	26	84.6	15.4	0.0	0.0	15.4	100.0
210	1	67	82.1	17.9	0.0	0.0	17.9	100.0
211	1	15	60.0	40.0	0.0	0.0	40.0	100.0
212	1	23	82.6	17.4	0.0	0.0	17.4	100.0
213	1	31	67.7	32.3	0.0	0.0	32.3	100.0
214	1	47	70.2	29.8	0.0	0.0	29.8	100.0
215	1	52	76.9	23.1	0.0	0.0	23.1	100.0
216	1	51	84.3	15.7	0.0	0.0	15.7	100.0
217	1	71	80.3	19.7	0.0	0.0	19.7	100.0
218	1	57	80.7	19.3	0.0	0.0	19.3	100.0
219	1	52	65.4	34.6	0.0	0.0	34.6	100.0
220	1	1	100.0	0.0	0.0	0.0	0.0	0.0
221	1	20	70.0	30.0	0.0	0.0	30.0	100.0

```
|--------+-------------+-----------------------------------------|
| 222    |    1     38 | 81.6   18.4    0.0    0.0   18.4  100.0 |
|--------+-------------+-----------------------------------------|
| 223    |    1     34 | 79.4   20.6    0.0    0.0   20.6  100.0 |
|--------+-------------+-----------------------------------------|
| 224    |    1     19 | 89.5   10.5    0.0    0.0   10.5  100.0 |
|--------+-------------+-----------------------------------------|
| 225    |    1     50 | 94.0    6.0    0.0    0.0    6.0  100.0 |
|--------+-------------+-----------------------------------------|
| 226    |    1     21 | 95.2    4.8    0.0    0.0    4.8  100.0 |
|--------+-------------+-----------------------------------------|
| 227    |    1     59 | 89.8   10.2    0.0    0.0   10.2  100.0 |
|--------+-------------+-----------------------------------------|
| 228    |    1     23 | 82.6   17.4    0.0    0.0   17.4  100.0 |
|--------+-------------+-----------------------------------------|
| 229    |    1     33 | 75.8   24.2    0.0    0.0   24.2  100.0 |
|--------+-------------+-----------------------------------------|
| 230    |    1     32 | 65.6   34.4    0.0    0.0   34.4  100.0 |
|--------+-------------+-----------------------------------------|
| 231    |    1     57 | 75.4   24.6    0.0    0.0   24.6  100.0 |
|--------+-------------+-----------------------------------------|
| 232    |    1     27 | 77.8   22.2    0.0    0.0   22.2  100.0 |
|--------+-------------+-----------------------------------------|
| 233    |    1     12 | 75.0   25.0    0.0    0.0   25.0  100.0 |
|--------+-------------+-----------------------------------------|
| 234    |    1     40 | 72.5   27.5    0.0    0.0   27.5  100.0 |
|========+=============+=========================================|
| Sum/Avg|  234   9678 | 86.5   13.4    0.0    0.0   13.5   79.5 |
|========+=============+=========================================|
|  Mean  |  1.0   41.4 | 88.8   11.2    0.0    0.0   11.2   79.5 |
|  S.D.  |  0.0   41.2 | 10.7   10.7    0.0    0.0   10.7   40.5 |
| Median |  1.0   31.0 | 91.9    8.1    0.0    0.0    8.1  100.0 |
`--------+-------------+-----------------------------------------'
```

Annexe10 : Script d'alignement :

```
#!/bin/bash
# supprimer les tabulations
sed 's/\t/\//g' $1 > $1\_1
# supprimer "\n"
tr '\n' ' ' < $1\_1 > $1\_2
# mettre "\n" à la fin de chaque phrase
sed 's/\.\/SENT/\.\/SENT\n/g' $1\_2 > $1\_3
#for num in $(seq 1 1 234 ) ; do echo -n "(${num})" >>
numid.txt ; echo >> numid.txt; done ;
# l'alignement du texte
paste -d ' ' $1\_3 numid.txt > $1\_4.txt
```

Annexe11 : Script de nettoyage de lexique :

```
my
$file="/home/dhaou/install/TreeTaggerarabe/data/lexicon_arabe
.txt";
my %hash=();
open (IN,"<",$file);
my @lines=<IN>;

foreach my $line (@lines){
      if($line=~/^(.*\$*.+)\t(.+)\t(.*\$*.+)(\t.*\$*.+)*[\n\r
]$/){
            my $mot=$1;
            my $cat=$2;
            my $lemm=$3;

            if(!exists($hash{$mot})){
                  $hash{$mot}=$cat."\t".$lemm;
            } else {
                  if
($line=~/^(.*\$*.*)\t(.*)\t(.*\$*.*)?$/&& $cat ne $2 || $lemm
ne $3){
                        $hash{$mot}.="\t".$2."\t".$3;
                  }
            }

      }

}
close(IN);


open(OUT,">","tmp2.txt");
foreach my $key (keys(%hash)){
      print OUT $key."\t".$hash{$key}."\n";
}

close(OUT);
```

Annexe12 : Script de correction du corpus de travail (Extrait) :

```
#!/bin/sh
sed         -e         's/lwdfYj\tNN/lwdfYj\tNNP/g'         -e
's/AlmwsYqy\tNN/Al\tDT\nmwsYqy\tNN/g'<
corpus_train.buck.TOK.POS.txt >corpus_train.new.txt
sed -i 's/Al>EZm\tJJ/Al\tDT\n>EZm\tJJ/g' corpus_train.new.txt
 sed -i 's/AlfDl\tNN/Al\tDT\nfDl\tNN/g' corpus_train.new.txt
```

```
sed       -i       's/AlklAsYkYp\tJJ/Al\tDT\nklAsYkYp\tJJ/g'
corpus_train.new.txt
sed          -i          's/AlESwrØŒ\tNN/Al\tDT\nESwr\tNN/g'
corpus_train.new.txt
sed       -i       's/Al>wrkstrA\tNN/Al\tDT\n>wrkstrA\tNN/g'
corpus_train.new.txt
sed -i 's/ts_Ep\tNN/Ts_Ep\tCD/g' corpus_train.new.txt
sed -i 's/AlEdYd\tNN/Al\tDT\nEdYd\tNN/g' corpus_train.new.txt
sed       -i       's/AlmqTwEAt\tNNS/Al\tDT\nmqTwEAt\tNNS/g'
corpus_train.new.txt
sed          -i          's/Al>wbrA\tNN/Al\tDT\n>wbrA\tNN/g'
corpus_train.new.txt
sed        -i         's/AlmwsYqYp\tJJ/Al\tDT\nmwsYqYp\tJJ/g'
corpus_train.new.txt
sed-i's/b\tIN\nYthwfn\tNN/bYthwfn\tNNP/g'
corpus_train.new.txt
sed -i 's/Yfqd\tJJ/Yfqd\tVBN/g' corpus_train.new.txt
sed      -i      's/AlvlAvYnYAt\tNNS/Al\tDT\nvlAvYnYAt\tCD/g'
corpus_train.new.txt
sed -i 's/Al\*Y\tNN/Al\*Y\tWP/g' corpus_train.new.txt
sed -i 's/Alftrp\tNN/Al\tDT\nftrp\tNN/g' corpus_train.new.txt
sed          -i          's/Al<bdAE\tNN/Al\tDT\n>bdAE\tNN/g'
corpus_train.new.txt
sed        -i         's/AlsmfwnYp\tJJ/Al\tDT\nsmfwnYp\tJJ/g'
corpus_train.new.txt
sed          -i          's/AlxAmsp\tJJ/Al\tDT\nxAmsp\tJJ/g'
corpus_train.new.txt
sed          -i          's/AlsAdsp\tJJ/Al\tDT\nsAdsp\tJJ/g'
corpus_train.new.txt
sed          -i          's/AltAsEp\tJJ/Al\tDT\ntAsEp\tJJ/g'
corpus_train.new.txt
sed        -i        's/Al>lmAnYp\tNN/Al\tDT\n>lmAnYp\tNNP/g'
corpus_train.new.txt
sed -i 's/AlfnAn\tNN/Al\tDT\nfnAn\tNN/g' corpus_train.new.txt
sed -i 's/E$rmn\tNN/E$r\tCD\nmn\tIN/g' corpus_train.new.txt
sed -i 's/1783\tCD\nm\tNN/1783m\tCD/g' corpus_train.new.txt
sed -i 's/AtsEt\tNN/AtsEt\tVBP/g' corpus_train.new.txt
sed -i 's/mbkrpØŒ\tNN/mbkrpØŒ\tJJ/g' corpus_train.new.txt
sed -i 's/\*AE\tNN/\*AE\tVBN/g' corpus_train.new.txt
 sed -i 's/EAny\tNN/EAny\tVBP/g' corpus_train.new.txt
sed -i 's/kvYrAF\tNNP/kvYrAF\tRB/g' corpus_train.new.txt
 sed -i 's/Alrgm\tNN/Al\tDT\nrgm\tRb/g' corpus_train.new.txt
sed -i 's/Al>wl\tJJ/Al\tDT\n>wl\tJJ/g' corpus_train.new.txt
sed -i 's/lqn\tNN/lqn\tVBN/g' corpus_train.new.txt
sed -i 's/AlEzf\tJJ/Al\tDT\nEzf\tNN/g' corpus_train.new.txt
sed -i 's/Ely\tNNP/Ely\tIN/g' corpus_train.new.txt
sed -i 's/Al>b\tNN/Al\tDT\n>b\tNN/g' corpus_train.new.txt
sed-i's/AlmvAlY\tJJ/Al\tDT\nmvAlY\tJJ/g' corpus_train.new.txt
sed -i 's/kAn\tVBD/kAn\tIN/g' corpus_train.new.txt
sed -i 's/AlkHwl\tNN/Al\tDT\nkHwl\tNN/g' corpus_train.new.txt
sed -i 's/twfYt\tNNP/twfYt\tVBN/g' corpus_train.new.txt
```

```
sed-i's/AlsAbEp\tJJ/Al\tDT\nsAbEp\tJJ/g' corpus_train.new.txt
sed -i 's/E$r\tNN/E$r\tCD/g' corpus_train.new.txt
sed -i 's/AlmrD\tNN/Al\tDT\nmrD\tNN/g' corpus_train.new.txt
sed-i's/AlEA}lp\tNN/Al\tDT\nEA}lp\tNN/g' corpus_train.new.txt
sed -i 's/Alsfr\tNN/Al\tDT\nsfr\tNN/g' corpus_train.new.txt
sed -i 's/AlEsr\tNN/Al\tDT\nEsr\tNN/g' corpus_train.new.txt
sed-i's/Alt>lYf\tNN/Al\tDT\nt>lYf\tNN/g' corpus_train.new.txt
sed-i's/AlmwsYqY\tJJ/Al\tDT\nmwsYqY\tJJ/g'
corpus_train.new.txt
sed -i 's/AlElAj\tNN/Al\tDT\nElAj\tNN/g' corpus_train.new.txt
```

Annexe13: Code d'apprentissage:

```
#!/bin/sh
/home/dhaou/install/TreeTagger/bin/train-tree-tagger
data/lexique_arabe.txt                    data/openclassTags.txt
data/corpus_app_0.txt lib/arabe.par
```

Annexe14: Code d'étiquettage:

```
 #!/bin/sh
/home/dhaou/install/TreeTagger/bin/tree-tagger          -token
lib/arabe.par data/test.txt data/resultat.txt
```

Annexe15 : Définitions des quelques mots utilisés:

Racine : la racine d'un verbe est une chaîne qui est à la base de tout verbe dérivé selon trois lettres.

Suffixe : c'est une chaine de caractères qu'ont ajouté à la fin du verbe pour constituer la forme conjugué de ce verbe.

Préfixe : c'est une chaîne de caractères qu'ont ajouté au début du verbe pour constituer la forme conjugué de ce verbe.

UNIVERSITE Stendhal GRENOBLE 3

Déclaration anti-plagiat

Document à scanner après signature
et à joindre au mémoire électronique

DECLARATION

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

NOM : ...G. Haula..... PRENOM :.....D. H Aou.....

DATE : 27/09/2011....