Université paris 8

UNIVERSITÉ PARIS 8 DES CRÉATIONS

LABORATOIRE PARAGRAPHE

École doctorale : Cognition, Langage, Interaction

THÈSE

pour obtenir le grade de docteur délivré par

Université Paris 8 Vincennes Saint-Denis

Spécialité doctorale *Informatique*

présentée et soutenue publiquement par

Hussein Awdeh

le 17 décembre 2025

Induction de segmentation morphologique grammaticale en arabe standard moderne

Directeur de thèse : Gilles Bernard, Paragraphe, U. Paris 8
Co-directeur de thèse : Mohammad Hajjar, GRIT, U. Libanaise

Jury

M. Kurosh Madani,	Professeur	Paris-Est Créteil	LISSI	Président
M. Hichem Maaref,	Professeur	Evry Paris-Saclay	IBISC	Rapporteur
M. Faouzi Boufares,	MCF HDR	Sorbonne Paris-Nord	LIPN	Rapporteur
M. Gilles Bernard,	Professeur	Paris 8	Paragraphe	Directeur
M. Mohammad Hajjar,	Professeur	U. Libanaise	GRIT	Co-directeur

Université Paris 8
Paragraphe
EA n° 349 Saint Denis, France

THÈSE

Remerciements

Mes sincères respects vont à toutes les personnes qui ont contribué à la bonne orientation de mon travail.

Je dois tout d'abord beaucoup à mon directeur de thèse, le professeur Gilles Bernard, qui a guidé ma thèse durant mes recherches. Merci du fond du cœur pour la qualité de vos attentions, votre force, vos compétences, votre esprit vif, votre créativité et votre patience. Sa confiance en mes recherches et mes capacités, qui m'a permis de faire partie de son équipe de recherche, m'a poussé de plus en plus à approfondir les pratiques de traitement du langage et le domaine de l'intelligence artificielle.

De même, je tiens à exprimer mes remerciements à mon co-directeur de ma thèse de doctorat, M. le professeur Mohammed Hajjar, Doyen de l'Institut Universitaire de Technologie (IUT Saida) à l'Université Libanaise, qui a été mon professeur et mon superviseur durant mes études supérieures à l'école doctorale spécialité Système d'information, et qui m'a formé et m'a encouragé à continuer mes recherches avec lui en signant une convention de co-direction de thèse entre l'université Paris 8 et l'université libanaise. Je dois beaucoup à ses aides, ses exhortations, ses conseils, ses observations et ses multiples corrections, ainsi qu'à son soutien inépuisable au bon moment.

Tout mon respect et mes remerciements vont également aux membres du jury, qui me font l'honneur de juger mon travail : aux rapporteurs de mon mémoire, MM. Faouzi Boufares et Hichem Maaref.

Mes remerciements les plus chaleureux vont aussi à M. Khaldoun Zreik directeur du Laboratoire Paragraphe, à M. Arab AliCherif en charge de l'équipe d'informatique de ce laboratoire. Merci à mes collègues Adelle Abdallah et Youssef Zaki qui m'ont aidé à surmonter de nombreux obstacles.

Je souhaite également adresser mes remerciements de manière générale à l'ensemble des professeurs des deux universités, Université libanaise et Université Paris 8, et à l'équipe de recherche, pour leurs efforts, leur aide précieuse, leurs conseils et leur bonne humeur.

Ma sincère gratitude va à mes parents pour leur intérêt pour moi, leur soutien,

leurs prières durant mes études.

Un grand merci du fond du cœur à ma chère femme pour son encouragement, son soutien tout au long de mes études, surtout en m'aidant dans la correction de la rédaction de mes articles.

Je remercie beaucoup mes filles Aya, Yara et Sara qui ont enduré mes occupations et mes fréquentes absences.

Enfin, je remercie sincèrement tous mes collègues et mes amis pour leurs soutiens, et leurs encouragements au cours de mes recherches.

Sommaire

Remerciements 3

Résumé 7

Abstract 9

Introduction 11

I Problématique 15

1 La segmentation en arabe standard moderne 17

2 Corpus en langue arabe 43

II État de l'art 63

Introduction à l'état de l'art 65

3 La segmentation par règles 71

4 La segmentation complexe 117

5 Étiquetage des parties du discours 141

III Système et réalisations **155**

6 Corpus de référence et évaluation des segmenteurs **157**

7 Éléments pour un segmenteur **195**

Conclusion et perspectives **205**

A Mots invariables **207**

B Segmentations multiples du gold corpus **211**

Mes publications **217**

Bibliographie **219**

Table des figures **245**

Liste des tableaux **247**

Table des matières **256**

Résumé

Cette thèse étudie la question de la segmentation morphologique grammaticale en arabe standard moderne, sous l'angle de l'induction de connaissances. La segmentation est l'opération par laquelle les mots écrits sont découpés en mots sémantiques et en morphèmes. Cette opération est relativement simple dans la plupart des écritures européennes, mais en arabe, elle est beaucoup plus complexe, pour trois raisons : (a) le mot écrit en arabe contient souvent plusieurs mots sémantiques, dont un contient plusieurs morphèmes ; il peut même contenir une phrase entière ; (b) l'écriture arabe omet dans sa version courante les voyelles brèves et un certain nombre d'autres éléments phonétiques, et certaines lettres peuvent être lues comme des voyelles longues ou des consonnes, d'où une grande quantité d'ambiguïtés et de découpages possibles ; (c) la morphologie de l'arabe est riche et complexe. D'une opération triviale, la segmentation devient une opération qui implique la grammaire, la morphologie, la syntaxe et la sémantique. Pour induire la segmentation à partir de données observées, objectif ultime, il faut également des données annotées disponibles, et leur très faible nombre est un aspect du problème. Nous nous sommes concentrés, dans ces travaux de recherche, à constituer l'ensemble des éléments nécessaires à un tel projet, centré sur la morphologie grammaticale plutôt que lexicale.

À partir d'une étude approfondie du problème et des solutions qui y ont été apportées par l'état de l'art, nous proposons une série d'expériences et de réalisations qui culminent autour de la production d'un corpus de référence annoté manuellement (dit *gold corpus*) et d'une méthodologie pour étendre ce corpus automatiquement à une grande taille tout en préservant au maximum la fiabilité des annotations (dit *silver corpus*). Ces corpus nous ont servi à évaluer un ensemble conséquent d'algorithmes de segmentation, pour en sélectionner quelques-uns dont nous analysons les défauts en vue de préparer la réalisation d'un segmenteur à base d'intelligence artificielle, qui corrigerait ces défauts en apprenant sur notre corpus étendu. Notre méthodologie est appliquée à des cas concrets et peut être étendue. Nous mettons également à la disposition de la communauté scientifique, en plus de ces corpus, des outils techniques, des données prétraitées et tout ce qui est nécessaire à la mise en place d'un apprentissage séquentiel.

Mots clés : Segmentation, Arabe standard moderne, Corpus de référence, Annotation, Informatique linguistique, Morphologie grammaticale, Racinisation, Lemmatisation, Induction

Abstract

This research explores the issue of grammatical morphological segmentation in Modern Standard Arabic from the perspective of knowledge induction. Segmentation refers to the process by which written words are broken down into semantic units and morphemes. While this operation is relatively straightforward in most European scripts, it is significantly more complex in Arabic for three main reasons : (a) A written Arabic word may contain several semantic words, with one being composed of multiple morphemes ; it can even represent an entire sentence. (b) Standard Arabic orthography omits short vowels and several other phonetic elements, and some letters can be interpreted as either long vowels or consonants, resulting in a high degree of ambiguity and multiple possible segmentations. (c) Arabic morphology is rich and intricate. What is a trivial operation in other languages becomes a process that involves grammar, morphology, syntax, and semantics. To induce segmentation from observed data — the ultimate goal — annotated resources are required. However, their scarcity is a major challenge. This research focuses on assembling the necessary components for such a project, with an emphasis on grammatical rather than lexical morphology.

Building on a thorough analysis of the problem and existing solutions, we present a series of experiments and outcomes centered on the creation of a manually annotated gold corpus and a methodology for automatically expanding it into a larger, reliable silver corpus. These resources enabled us to evaluate a wide range of segmentation algorithms and identify key limitations, paving the way for the development of an AI-based segmenter trained on our extended corpus. Our approach is applied to concrete cases and remains scalable. In addition to the corpora, we provide the scientific community with technical tools, preprocessed data, and all necessary components for sequential learning.

Keywords : Segmentation, Modern Standard Arabic, Gold Corpus, Silver Corpus, Tagging, Computational linguistics, Grammatical morphology, Stemming, Lemmatisation, Induction

Introduction

La présente recherche porte sur la segmentation automatique des mots de l'arabe standard moderne en morphèmes, en utilisant le moins de connaissances possibles sur la langue considérée, d'où le terme d'induction. Elle fait partie d'un projet plus général porté par le laboratoire GRIT de l'université Libanaise et par le laboratoire Paragraphe de l'université Paris 8, qui consiste à élaborer des outils d'induction pour l'arabe avec une utilisation minimale de connaissances préalables.

Ce projet repose sur l'hypothèse que le sens est au moins partiellement reconstructible au travers des propriétés formelles et de la distribution des formes, en cherchant à montrer ce qu'on peut construire à partir simplement d'un grand corpus de documents. On citera, par exemple, la thèse de G. Lebbos sur l'apport de la distribution des mots dans un grand corpus à leur analyse sémantique (Lebboss, 2016), la thèse d'A. Abdallah sur la possibilité d'induire des éléments de sémantique grammaticale à partir de la distribution des structures syntaxiques de surface (Abdallah, 2021), la thèse de Y. Zaki sur l'induction de grammaire de phrase à partir d'un grand corpus (Zaki, 2022).

Notre thèse s'inscrit dans ce contexte, insistant sur la morphologie grammaticale, que les travaux précédents de notre équipe n'ont pas exploitée. Elle se déroule en parallèle avec le travail de nos collègues doctorants O. Saadiyeh, poursuivant le travail de Y. Zaki vers la réalisation d'un analyseur syntaxique inductif, et M. Khalil, poursuivant le travail d'A. Abdallah et de G. Lebboss vers une description sémantique inductive, avec lesquels nous avons beaucoup échangé.

L'arabe standard moderne désigne la variété d'arabe utilisée dans les journaux, la plupart des livres, les informations télévisées, et, d'une manière générale, dans toutes les circonstances où l'on doit communiquer avec des personnes sans tenir compte de la région d'où elles sont originaires. Il s'oppose à l'arabe classique, la langue du Coran et de la période classique, à l'arabe ancien, de la période préislamique, et aux dialectes parlés aujourd'hui dans les différentes régions du monde arabe (arabe tunisien, marocain, libanais, égyptien, etc.; qui en réalité diffèrent même à l'intérieur d'un même pays). C'est une sorte de *koinê* (langue commune) qui a commencé à s'élaborer à l'époque ottomane (roman de Baibars, par exemple),

mais qui s'est véritablement instaurée avec l'apparition de la presse imprimée puis de la télévision. Il n'y a pas de locuteurs natifs de cette langue, d'où certaines difficultés qui lui sont propres.

La segmentation morphologique de l'arabe est difficile en raison à la fois de la complexité de la morphologie de la langue mais aussi en raison des conventions régissant l'écriture, s'agissant essentiellement d'une langue écrite. C'est pourtant une étape fondamentale du traitement automatique des langues, nécessitée par exemple pour l'extraction d'information, la traduction automatique, l'analyse syntaxique. Un élément tout simple permet de se rendre compte de la difficulté : il y a un écart de près de quarante ans entre la définition de la méthodologie de construction d'analyseurs morphologiques par Koskenniemi (1983), qui a donné lieu à la plateforme morphologique PC Kimmo utilisée pour de nombreuses langues, et l'équivalent pour l'arabe par Habash, Marzouk et al. (2022), avec la plateforme CamelMorph. Au moment où cette thèse a débuté, cette méthodologie n'existait pas encore. Un deuxième aspect du problème est le faible nombre de corpus libres annotés disponibles pour l'arabe (et même de corpus annotés tout court). Les chercheurs sont souvent amenés à réaliser leurs propres corpus de test, ce qui restreint la portabilité et la reproductibilité des expériences.

Parmi les différents types que recouvre le concept de segmentation et les mots qui lui sont souvent associés (par exemple racinisateur, lemmatiseur, extracteur de racine) et dont la définition n'est pas toujours claire, nous nous intéressons à la segmentation avec pour objectif l'input dans un système d'induction de motifs grammaticaux auquel nous avons aussi participé (Abdallah et al., 2021), et qui à son tour doit servir à la construction d'un analyseur syntaxique. D'où notre focalisation sur la segmentation de la morphologie grammaticale, qui diffère du rôle de la segmentation dans l'extraction d'information par exemple, plutôt orienté sur le côté lexical de la segmentation.

Notre mémoire se répartit en trois parties : problématique, état de l'art et réalisations.

La première partie présente la problématique sous deux angles :

Le premier chapitre détaille la problématique de la segmentation, concernant la langue et l'écriture, ainsi que les propriétés fondamentales sur lesquelles on peut s'appuyer à la recherche de solutions. Cette problématique permet de définir différents niveaux de segmentation, ainsi qu'une liaison intime entre la segmentation et la détection des catégories morpho-syntaxiques de l'arabe (parties du discours). Les données fournies sont le plus exhaustives possibles.

Le deuxième chapitre étudie la problématique des corpus, avec un intérêt particulier pour les corpus pré-segmentés en open source. En effet, pour pouvoir appliquer des techniques modernes d'induction, par exemple neuronales, il faut pouvoir

apprendre par l'exemple ; d'autre part, le caractère open-source permet que chaque chercheur puisse ajouter sa propre contribution au corpus. On y remarquera qu'il y a en fait très peu de corpus qui répondent à cette exigence, et aucun qui ait été contrôlé manuellement.

La partie suivante présente l'état de l'art de notre domaine, en trois chapitres :

Les troisième et quatrième chapitres passent en revue l'ensemble des segmenteurs existants, avec un effort d'exhaustivité, et leurs algorithmes respectifs, en mettant l'accent sur une comparaison approfondie et une évaluation critique de ces segmenteurs. Nous les répartissons en différentes catégories en fonction du type d'information utilisée, du type de retour, et du type d'algorithme mis en oeuvre. On y remarque que peu de segmenteurs ont tenté une approche inductive jusqu'à aujourd'hui, et aucun sur l'arabe. La matière étant très abondante, nous avons divisé cette présentation en deux chapitres en fonction de la complexité des algorithmes utilisés. Le troisième chapitre aborde les segmenteurs à base de règles ; le quatrième aborde les segmenteurs intégrant plusieurs techniques, probabilistes et neuronales.

Le cinquième chapitre passe en revue l'ensemble des étiqueteurs de parties du discours existants, car, comme l'a montré le premier chapitre, segmentation et étiquetage sont intimement liés. Nous nous intéressons particulièrement aux étiqueteurs apprenant par l'exemple.

La troisième partie présente ce que nous avons réalisé, en deux chapitres :

Le sixième chapitre explique comment nous avons constitué notre propre corpus segmenté et open-data, pour pallier à l'absence de tels corpus montrée au chapitre deux ; ce corpus est extrait du seul corpus annoté morphologiquement et open-data pour l'arabe standard moderne, Kalimat, dont nous analysons les annotations et les différentes caractéristiques. Nous détaillons les caractéristiques du corpus que nous avons manuellement annoté morphologiquement et segmenté, et nous démontrons qu'il a, malgré sa relative petite taille, les propriétés d'un corpus standard. Nous utilisons ensuite notre corpus pour évaluer les segmenteurs présentés dans l'état de l'art, sous plusieurs angles, avec une analyse approfondie de leurs résultats, qui nous amènent à en élminer la plupart ; et c'est Farasa qui s'avère le plus performant, avec malheureusement des défauts très problématiques, en particulier l'échec à segmenter plus d'un verbe sur deux et l'incapacité à résoudre les cas ambigus.

Le septième chapitre présente nos éléments pour servir à la réalisation d'un segmenteur capable d'être utilisé dans un pipe Ce corpus n'étant pas assez grand pour servir de base à l'apprentissage, nous avons réfléchi sur des méthodes de fiabilisation des annotations automatiques produites sur le corpus initial (Kalimat), contenant une dizaine de millions de mots. Faute de temps et de moyens, nous

n'avons pas pu effectuer la totalité des expériences que nous avions prévu de faire pour générer un méta-segmenteur à partir des meilleurs segmenteurs de l'état de l'art.

Le septième chapitre propose un segmenteur sans connaissance préalable, basé sur l'architecture GRU, fonctionnant à partir des annotations produites sur le corpus complet avec une ingénierie des caractéristiques mêlant le *one-hot encoding* et les méthodes d'*embedding*, pour obtenir un corpus annoté conséquent. Malheureusement le résultat n'a pu être contrôlé que sur l'ensemble d'apprentissage, car notre corpus de référence n'était pas assez grand pour être divisé en corpus d'apprentissage et corpus de test. La petite taille du corpus est malgré tout partiellement compensée par la forte redondance des annotations.

Puis la conclusion récapitule les éléments essentiels de notre travail et propose des pistes pour la suite, en particulier les possibilités qu'ouvre le *few-shot learning*, qui n'existait pas au début de cette thèse mais dont l'intérêt n'a fait que croître.

Les principales contributions de cette thèse sont :

- une revue relativement complète des corpus en arabe standard moderne avec une attention particulière sur le statut du corpus et l'existence d'annotations (Awdeh, Abdallah et al., 2021c) ;
- un état de l'art quasi-exhaustif des segmenteurs et de leurs algorithmes, ainsi qu'un état de l'art des étiqueteurs de parties du discours ;
- la production[1] d'un corpus open-data de taille modeste (près de 29 000 mots environ) manuellement segmenté (le seul existant à notre connaissance) (Awdeh, Abdallah et al., 2021a,b ; Awdeh, 2025) ;
- une évaluation des principaux segmenteurs sur ce corpus (Awdeh, Saadiyeh et al., 2025) ;
- la production d'un grand corpus (10 millions de mots environ) segmenté avec les meilleurs segmenteurs et annoté par une procédure automatique fiabilisée pour les catégories grammaticales de l'arabe : noms, verbes, particules (Awdeh, 2025 ; Awdeh, Khalil et al., 2025) ;
- les caractéristiques des vecteurs d'entrée d'un segmenteur à apprentissage séquentiel (Awdeh, 2025).

Il nous semblé crucial que l'ensemble des éléments produits pour ce travail soient disponibles en open-source (licence CC by NC).

1. Mon livret "A Silver Standard Arabic Corpus for Segmentation And Validation" est disponible sur https://www.amazon.com/dp/1952751365, https://www.elivapress.com/en/book/book-8238862797. dans Amazon et Eliva Press en 2020.

Première partie

Problématique

Chapitre 1

La segmentation en arabe standard moderne

Sommaire

1.1 Introduction 18
1.2 Bref aperçu phonologique 19
1.3 La graphie arabe 21
1.3.1 Les unités de la graphie 24
1.3.2 Variantes graphiques 24
1.4 Structure du mot 26
1.4.1 Clitiques et catégories 30
1.5 Morphologie du mot 32
1.5.1 Noms et adjectifs 33
1.5.2 Verbes 34
1.5.3 Verbes faibles 36
1.5.4 Affixes et catégories morphosyntaxiques 37
1.6 Types de segmentation 39
1.7 Conclusion 41

1.1 Introduction

Ce chapitre présente l'essentiel de la problématique concernant la segmentation des mots en arabe standard moderne, du point de vue de la graphie et de la langue, ainsi que les types d'approches possibles pour la résoudre. Pour plus de détails, voyez Habash (2005), thèse qui explore la morphologie arabe et ses interactions avec la phonologie et le lexique, ainsi que les implications de la structure morphologique sur le traitement automatique de l'arabe.

La segmentation des mots écrits en unités signifiantes appelées morphèmes (Creutz et al., 2007) constitue une étape fondamentale dans les applications de traitement automatique des langues (Yemane Tedl, 2018). C'est aussi bien vrai pour la recherche d'information et la fouille de documents (Green et DeNero, 2012) que pour l'analyse syntaxique (S. Chen, Y. Xu et Chang, 2011) ou sémantique, et pour la compréhension automatique. Sans elle, pas d'étiquetage des parties du discours, pas d'analyse syntaxique ou sémantique.

La segmentation permet également de limiter le nombre de mots écrits à traiter. Par exemple, en anglais, des formes telles que *subreddits*, *clickbait* et *nanodiamonds*, pourront être rares et ne pas se retrouver dans nos ressources externes, le résultat de la segmentation *sub-reddits*, *click-bait* et *nano-diamonds* pourra plus facilement y être retrouvé (Melo, 2014).

Cette décomposition revêt une importance et des difficultés particulières dans le contexte des langues sémitiques, notamment de l'arabe, dont la structure morphologique est complexe. En effet, dans les langues à morphologie fusionnelle comme l'arabe, nous pouvons trouver des milliers de formes différentes du même mot, créant des défis de rareté difficiles à surmonter pour les modèles qui reposent sur l'identification des mots. Une segmentation morphologique précise permet non seulement de relier de nouvelles formes à des formes de mots précédemment connues, mais diminue également considérablement la rareté des données et le nombre d'hapax.

Le problème, comme nous le verrons, est accentué par les caractéristiques propres à l'écriture arabe, où le mot écrit peut correspondre à ce qui, dans d'autres systèmes d'écriture, serait une phrase entière. Par conséquent, non seulement on a affaire à de nombreuses formes flexionnelles, ainsi qu'à des mots-valises comme dans l'exemple anglais ci-dessus, mais les pronoms, les articles, et les prépositions, sont intégrés dans les mots. Comparé par exemple au cas de l'anglais, où la segmentation est principalement affaire de mots composés, ici la segmentation implique aussi bien la syntaxe que la sémantique.

Un troisième aspect du problème est la rareté des ressources permettant l'élaboration de méthodes automatiques, et le fait que les rares ressources existantes ne

sont pas libres, et souvent trop chères pour les chercheurs des laboratoires des pays concernés. C'est en particulier le cas des corpus étiquetés, qui sont une nécessité pour appliquer des méthodes d'intelligence artificielle.

C'est pourquoi l'objectif de cette thèse est de faire un état des lieux de la segmentation des mots écrits en arabe, de proposer une méthodologie de travail avec des ressources *open-data*, et de montrer l'application de cette méthodologie pour produire notre propre modèle de segmenteur basé sur l'apprentissage automatique par des réseaux récurrents.

Dans la suite de ce chapitre, nous allons présenter en détail les trois types de difficultés auxquelles ceux qui souhaitent élaborer un segmenteur de l'arabe sont confrontés : celles liées à l'écriture, à la langue, et aux corpus étiquetés. Les descriptions que nous ferons de la graphie, de la langue et des corpus se concentreront uniquement sur les aspects qui impactent notre problématique.

1.2 Bref aperçu phonologique

Avant d'aborder les problèmes de la graphie, nous décrivons brièvement la phonologie de l'arabe standard moderne (ASM). L'arabe standard moderne est une langue commune à l'ensemble des locuteurs des variétés d'arabe parlées dans les différentes régions, une *koinê*. Ce n'est la langue native de personne ; par conséquent, sa phonologie varie quelque peu en fonction de la variété native du locuteur, mais obéit à des principes communs. C'est une langue sémitique, et comme les autres langues de cette famille, elle possède beaucoup de consonnes et peu de voyelles. La réalisation des voyelles est l'endroit où il y a le plus de différences d'une variété à l'autre, et les voyelles ont essentiellement une fonction grammaticale.

L'ASM possède un système phonologique de 6 voyelles :

- 3 brèves : {**a, i, u**}.
- 3 longues : {**ā, ī, ū**}, transcrites aussi comme {**a:, i:, u:**} et translittérées comme {**aa, ij ou iy, uw**}.

Il possède 28 consonnes, présentées dans le tableau 1.1, qui est tiré de la représentation du système des consonnes par Roman (1974) (reprenant Cantineau). Les phonèmes sont transcrits en API (alphabet phonétique international), mais pour un certain nombre de phonèmes, on utilise traditionnellement la variante en gras à droite plutôt que les transcriptions API qui figurent sur leur gauche. Dans la partie droite du tableau figure la description de chaque consonne en termes de traits (sourd / sonore, point d'articulation labial / dental / interdental / alvéolaire / vélaire / uvulaire / glottal, etc.).

Tab. 1.1 : Table des consonnes

Lettre	**Description**
m	complexe vocalique interrompu continu nasal labial
n	complexe vocalique interrompu continu nasal dental
l	complexe vocalique interrompu continu alvéolaire
r	complexe vocalique interrompu alvéolaire
w	complexe vocalique labial
j y	complexe vocalique palatal
tˤ ṭ	complexe interrompu sourd emphatique dental
t	complexe interrompu sourd dental
k	complexe interrompu sourd vélaire
q	complexe interrompu sourd uvulaire
dˤ ḍ	complexe interrompu emphatique dental
d	complexe interrompu dental
b	complexe interrompu labial
ṣ	complexe sourd emphatique alvéolaire
f	complexe sourd labial
s	complexe sourd alvéolaire
θ	complexe sourd interdental
ʃ ṡ	complexe sourd palatal
x χ	complexe sourd uvulaire
ħ ḥ	complexe sourd pharyngal
ðˤ ẓ	complexe emphatique interdental
ð	complexe interdental
z	complexe dental
ʒ ʤ ǧ	complexe palatal
ɣ	complexe uvulaire
ʕ	complexe pharyngal
ʔ	interrompu glottal
h	glottal

Les transcriptions de l'arabe que nous donnons dans la suite du mémoire sont celles de l'Alphabet Phonétique International (API), à l'exception des symboles indiqués en gras dans le tableau 1.1.

1.3 La graphie arabe

L'alphabet arabe est composé de vingt-huit lettres, de deux lettres spéciales qui ne s'utilisent qu'à la fin des mots, de huit signes diacritiques[1] et de quelques diacritiques supplémentaires comme ~ (madda), ء (hamza). Il faut ajouter à cette quarantaine de symboles différents le *tatweel*, un symbole qui sert à allonger plus ou moins la liaison entre deux lettres, et qui joue un rôle comparable à la césure (pour faciliter la justification du texte ; au lieu de couper un mot, on l'allonge). Bien entendu, ce symbole n'a aucune utilité dans la segmentation, et toutes les méthodes de segmentation l'éliminent.

La graphie est continue (la plupart des lettres sont liées entre elles) et la forme des lettres change en fonction du contexte. Mais comme nous sommes concernés ici par des textes électroniques, ceci n'impacte pas l'analyse, chaque lettre, quelle que soit la forme qu'elle prend, est représentée par un unique symbole unicode (contrairement aux majuscules et minuscules dans les alphabets qui connaissent cette différence). Il y a des ligatures (deux lettres réunies en une seule), mais celles-ci ne sont pas considérées comme des symboles distincts, donc elles ne nous concernent pas ici.

Le tableau 1.2 représente les lettres de l'alphabet arabe avec leur translittération. Le *alif* ayant trop de transcriptions possibles, on les a indiquées à part. La translittération est une exacte correspondance de la graphie arabe, alors que la transcription ne l'est pas forcément, en raison des caractères spéciaux utilisés. Les vingt-huit lettres de base correspondent aux vingt-huit consonnes citées à la section précédente, et les deux lettres spéciales ont un usage morphologique. Par exemple, le mot طاولة (*table*) est transcrit /**ṭāwla**/. La dernière lettre de ce mot, ة, compromis entre les lettres désignant /**h**/ et /**t**/, ne note pas la voyelle **a** ni une consonne mais le morphème de féminin *-at* qui se prononcera tantôt **a** tantôt **at** en fonction de la relation syntaxique avec le mot suivant (un peu comme avec les liaisons en français). Donc la translittération de la fin de ce mot donnerait /**l**$^{\textbf{at}}$/.

De la même façon, l'autre lettre spéciale, *alif maqsura*, dérivé de la lettre pour /**y**/, note un /**ā**/ dans une racine où figurait un /**y**/ en finale. Dans la présentation traditionnelle de l'alphabet arabe, *alif maqsura* est confondu avec *ya*, et il n'y a donc officiellement que vingt-neuf lettres.

1. Un diacritique est un signe ajouté au-dessus ou au-dessous d'une lettre pour en préciser la prononciation.

Tab. 1.2 : Alphabet (unicode)

Lettre	Translittération	Nom
ا		alif
ب	b	ba
ت	t	ta
ث	θ	tha
ج	ǧ	jim
ح	ħ	ha
خ	χ	kha
د	d	dal
ذ	ð	dhal
ر	r	ra
ز	z	zaïn
س	s	sin
ش	ʃ	chin
ص	ṣ	sad
ض	ẓ	dad
ط	ṭ	ta
ظ	ḍ	za
ع	ʕ	aïn
غ	ɣ	ghaïn
ف	f	fa
ق	q	qaf
ك	k	kaf
ل	l	lam
م	m	mim
ن	n	nun
ه	h	ha
و	w, ū	waw
ي	y, ī	ya
ة	a, at	ta marbuta
ى	ā	alif maqsura
ـــفـــ	–	taṭwīl

Le tableau 1.3 représente tous les signes diacritiques de l'écriture arabe. Au total il y a dix signes diacritiques (Abdulhay, 2012), même si les deux derniers ne sont en général pas considérés comme des diacritiques ; le *madda* n'est compatible qu'avec la lettre *alif*, et le *hamza* avec les lettres *alif*, *ya* et *waw*.

Tab. 1.3 : Les diacritiques arabes

Signe	Transcription	Nom	Position
◌َ	a	Fatha	au-dessus
◌ِ	i	Kasra	en dessous
◌ُ	u	Dama	au-dessus
◌ْ	pas de voyelle	Sukun	au-dessus
◌ً	an	Tanwin Al Fath	au-dessus
◌ٍ	in	Tanwin Al Kasr	en dessous
◌ٌ	un	Tanwin Al Dam	au-dessus
◌ّ	gémination	Chadda	au-dessus
◌ء	ʔ	Hamza	comme la voyelle
◌ٓ	ā	Madda	sur alif

1. Les voyelles *fatha* (a) et *kasra* (i) sont représentées par un tiret mis respectivement au-dessus et au-dessous de la consonne qu'elles suivent. La voyelle *dama* (u) est représentée par un petit و (*waw*) au-dessus de la consonne qu'elle suit. Le signe *sukun* indique l'absence de voyelle.

2. Le signe *chadda* désigne une gémination (un redoublement) de la consonne sur laquelle il se trouve.

3. Le coup de glotte ʔ est une consonne originellement représentée par *alif* ; mais il disparaît après une voyelle en l'allongeant. Les voyelles longues ī, ū sont alors représentées par les consonnes les plus proches *ya* et *waw* ; la voyelle longue ā n'a pas de consonne proche, et c'est donc *alif* qui la représente. D'où une ambiguïté pour cette lettre, que les diacritiques *hamza* (cas du coup de glotte) et *madda* (cas de ā) désambiguïsent.

4. Le *hamza* est également utilisé pour représenter un coup de glotte avant une voyelle longue en l'absence de *alif*, il se met dans ce cas au dessus de la voyelle qu'il précède.

1.3.1 Les unités de la graphie

Selon Djamel Kouloughli (Kouloughli, 1982), les unités de la graphie arabe présentent deux caractéristiques :

- Elles correspondent à des unités accentuées qui comprennent plusieurs mots sémantiques (appelés mots minimaux par l'auteur),
- Chaque mot minimal est représenté par sa forme isolée.

Autrement dit, là où le français transcrit « je suis déjà venu dans cette ville-là », l'arabe transcrirait « jesuisdéjàvenu danscettevillelà » (pas de séparation dans une unité accentuelle), et là où le français transcrit « j'ai connu l'amour », l'arabe transcrirait « jeaiconnu leamour » (écriture des morphèmes sous leur forme isolée).

Exemple : on écrit /ʔaʃʃams/ (*le soleil*) comme الشمس /ʔalʃams/, parce que la forme isolée de l'article est /ʔal/, comme en français la forme isolée de l'article masculin[1] est « le » (et non pas « l' »).

1.3.2 Variantes graphiques

L'écriture arabe autorise de nombreuses variantes d'un même mot, ce qui conduit à la fois à des ambiguïtés (plusieurs mots différents écrits de la même façon) et en même temps à augmenter le nombre de mots uniques (un même mot écrit de plusieurs manières). La principale catégorie de variantes est due à l'usage des diacritiques, mais il y en a d'autres, que nous passons rapidement en revue ci-dessous.

Usage des diacritiques

De fait, les signes diacritiques (voyelles et gémination) sont rarement utilisés dans les documents arabes, qu'ils soient manuscrits ou électroniques. Par exemple :

- la forme حمل /ħml/ sans diacritiques peut correspondre à trois mots différents selon sa prononciation (حَمَلَ, حِمِلْ, حَمَلْ = ħamal, ħimil, ħamala, *mouton, fardeau, porter*) ;
- la forme رجل /rğl/ sans diacritiques peut correspondre à deux mots différents) selon sa prononciation (رَجُلْ, رِجِلْ = riğil, rağul *pied, homme*)
- la forme كتب /ktb/ sans diacritiques est utilisée pour désigner la racine « écrire » ; elle peut aussi correspondre à plusieurs dizaines de mots différents,

1. La forme qu'on a si on fait une pause avant le nom : "le... amour", d'où l'intitulé de forme pausale.

par exemple كَتَبَ /kataba/ (*il a écrit*), كُتِبَ /kutiba/ (a été écrit) et كُتُبًا /kutuban/ (*des livres*).

La présence des diacritiques éclaircit le contenu d'un texte et facilite sa lecture, donc les textes religieux et juridiques ainsi que les ouvrages pédagogiques modernes sont souvent entièrement voyellés. Mais la plupart des textes sont dépourvus de diacritiques, ou en mettent de temps en temps.

Coup de glotte

Le coup de glotte est un phonème instable qui disparaît dans certains contextes ; on l'a vu après une voyelle, mais même à l'initiale d'un mot il disparaît si le mot précédent se termine par une voyelle. Par exemple, dans إستقبل الضيوف /ʔistqbala (ʔ)alẓjwf/ (*il a accueilli les invités*), le hamza du deuxième mot disparaît ; dans متى استقبل الضيوف /mta (ʔ)astqbala (ʔ)alẓjwf/ (*Quand il a accueilli les invités*), tous les hamzas disparaissent. Par conséquent même dans un texte qui note tous les diacritiques, le hamza peut être présent ou non pour un même mot.

Lettres ى et ة

Les lettres ى (*alif maqsura*) et ة (*ta marbuta*) n'existent qu'à la fin des mots et sont souvent remplacées dans les documents électroniques par des lettres plus courantes. Du coup un même mot peut se retrouver écrit de deux façons différentes, par exemple le mot pour *bibliothèque*, مكتبة /mktba/ ou مكتبه /mktbah/, et deux mots différents peuvent se retrouver écrits de la même façon : ليلة /layla/ (*nuit*) et ليلى /laylā/ (*prénom*).

Combinaison ئ

La combinaison ئ représente à l'origine une séquence de deux phonèmes devenue un phonème simple (yʔ > ʔ) ; dans cette combinaison y est écrit sans ses points et se confond donc avec *alif maqsura*. Elle est souvent remplacée, dans les documents électroniques, par ي (y). Par exemple le mot arabe مهترئ /mhtrʔ/ (*moisi*) s'écrit souvent مهتري /mhtry/.

On peut donc dire que la graphie arabe a tendance à multiplier le nombre de formes d'un même mot, tout en générant des ambiguïtés avec d'autres mots. Le phénomène n'est pas inconnu ailleurs (majuscule / minuscule par exemple), mais ici c'est son ampleur qui est à noter : il peut y avoir pas loin d'une centaine de formes pour un même mot, et des centaines de mots représentés par une même forme. La combinatoire théorique est donc gigantesque, et pour la réduire, les locuteurs utilisent à la fois la syntaxe et la sémantique.

Pour réduire la variabilité introduite par les variantes graphiques, le traitement linguistique de l'arabe utilise extrêmement fréquemment une **normalisation orthographique**, souvent simplement appelée normalisation (arabe تطبيع taṭbiʕa). Comme restituer les diacritiques est une procédure très complexe, nécessitant une

analyse morphologique, syntaxique et sémantique, tâche qui doit être faite manuellement, la normalisation consiste surtout à supprimer les diacritiques ; elle consiste également à remplacer les différentes variantes de *hamza*, de *alif* ou de *ta marbuta* par un symbole unique. Il existe plusieurs systèmes de normalisation, en fonction de l'objectif visé. Mais bien que la normalisation orthographique diminue la variabilité dans l'écriture d'un mot donné, elle augmente du même coup son ambiguïté.

1.4 Structure du mot

Un mot écrit en arabe peut être considéré à trois niveaux ; au premier niveau, il est composé de clitiques (voir le tableau 1.6) et d'un mot minimal, qui peut être trouvé seul. Les clitiques (articles, possessifs, connecteurs, prépositions, etc.) ont les mêmes fonctions syntaxiques que les mots normaux, mais sont phonétiquement subordonnés et seront donc, dans la graphie arabe, fusionnés avec le mot (contrairement au français, où par exemple dans « je le regarde », *je* et *le* sont des clitiques non fusionnés) ; on peut souvent les trouver, parfois sous une forme légèrement différente, comme mots indépendants. Les clitiques sont divisés en proclitiques (s'ils précèdent le mot minimal) et enclitiques (s'ils le suivent) ; il peut y avoir plus d'un proclitique.

À son tour, le mot minimal est composé d'affixes et d'un radical ; les affixes peuvent être des préfixes, des suffixes ou des infixes (s'ils sont à l'intérieur du radical), voire des circonfixes (affixes complexes composés de plusieurs types d'affixes élémentaires). En français « rapidement, refaire » contiennent le suffixe *ment* ou le préfixe *re* ; mais en arabe, langue flexionnelle, les affixes jouent un rôle beaucoup plus important. Le radical n'existe généralement pas sans affixes.

Au troisième niveau, le radical est composé d'une racine et d'un schème vocalique, qui sont intriqués. Ces éléments seront détaillés à la section suivante. Les affixes et les schèmes sont interdépendants et ne peuvent donc pas être combinés aussi librement que les clitiques.

La segmentation va donc se faire en trois parties, correspondant à ces trois états morphologiques.

1. Dans le mot accentué (mot écrit), il faut identifier les clitiques pour les séparer du mot minimal ;
2. il faut décomposer le mot minimal en radical et affixes ;
3. il faut décomposer le radical en racine et schème ;
4. à partir de la racine et de la catégorie, déduite de la combinaison clitiques, affixes et schèmes, on trouve la forme du dictionnaire sous laquelle est enre-

gistré le mot, qu'on appelle **lemme**[1].

Attention : toutes les combinaisons de schèmes avec les racines ne sont pas possibles, car, en fonction de son sens, chaque racine n'est compatible qu'avec certains schèmes. C'est ce qu'on appelle un modèle. Et même parmi les radicaux possibles, selon Moftah et al. (2009), seuls 2% sont utilisés.

La figure 1.1 montre comment le mot est composé ; l'objectif de la segmentation est donc de remonter ce schéma à l'envers.

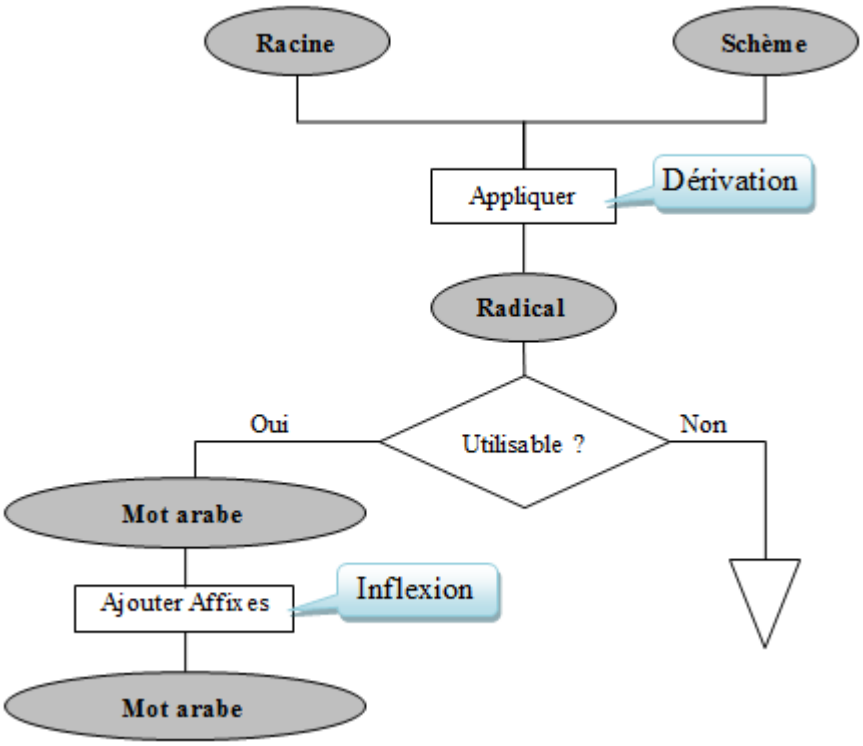


Fig. 1.1 : Génération d'un mot arabe (Al Hajjar, 2010)

Prenons l'exemple du mot أتلعبون /ʔatlʕabūn/ (*Est-ce que vous jouez ?)*, tableau 1.4.

1. On sépare d'abord le proclitique /ʔa/ (*Est-ce que*) du mot minimal /tlʕabūn/ (*vous jouez*) ; il n'y a ici qu'un seul mot, *jouez* ne nécessite pas de pronom ;

1. En général, pour les langues qui connaissent la distinction de nombre pour les noms, c'est la forme du singulier qui est le lemme. L'arabe, comme le français ou l'anglais, est dans ce cas. Pour le verbe, en français le lemme est l'infinitif, en anglais c'est l'infinitif (sans *to*) ; en latin en revanche, c'est la première personne de l'indicatif présent (ex. *amo* est le lemme de *amabat*). En arabe, c'est la troisième personne de l'accompli ; par ex. qatala, *il a tué*, est le lemme de tuqtalūna, *vous avez été tués.*

Tab. 1.4 : Analyse du mot أتلعبون /ʔatlʕabūn/ (Est-ce que vous jouez)

Proclitiques		Mot minimal					Enclitique	Lemme
		Préfixe	Radical		Suffixes			
			Racine	Schème				
ʔa		tlʕabūn						
		t	lʕab		ū	n		
			lʕb	CCaC				laʕiba, jouer

2. Le mot minimal /tlʕabūn/ (*vous jouez*) est séparé en préfixe /t/, indicateur de la deuxième personne, radical /lʕab/ et suffixes /ū/, indicateur du pluriel et /n/ de l'inaccompli.

3. Le radical /lʕab/ est dérivé de la racine /lʕb/ par application du schème CCaC. Ce schème est caractéristique de l'accompli, mais on le trouve aussi ailleurs. Plus de détails sur les schèmes sont donnés à la sect. 1.5.

4. Une fois qu'on a la racine /lʕb/, on peut trouver le lemme لَعِبَ, /laʕiba/ (*il a joué*), si on a trouvé la catégorie (verbe) du mot ; la combinaison d'éléments (proclitique, préfixe, suffixe et schème) indique ici qu'il s'agit d'un verbe, sans ambiguïté possible.

Donc, segmenter c'est construire une structure de la forme (il peut y avoir jusqu'à trois proclitiques, qui se suivent dans un ordre précis) :

procl.debut	procl	procl.fin	pref	radical	suff1	suff2	encl

Le tableau 1.5 donne une idée de la variété des analyses possibles. On notera de possibles confusions, par exemple ʔa peut être un clitique, avec le sens de « est-ce que », ou un préfixe caractérisant l'impératif, ou être confondu avec le début de l'article ʔal. Encore plus complexe, le cas de ʔiktalaʕ (il déracine), dans lequel le préfixe ʔi est présent à cause de l'infixe (non flexionnel) t inclus dans le radical ktalaʕ. Si nous traitions d'affixes dérivationnels, ce t serait à prendre en compte. Mais aucun infixe n'est flexionnel, ce qui n'est sans doute pas un hasard.

L'écriture sans diacritiques rend encore plus complexe l'analyse morphologique, en introduisant un grand nombre d'ambiguïtés supplémentaires.

Prenons l'exemple de la forme سيقبلون /syqblūn/ ; elle peut correspondre à سيقبّلون, /sayqabbalūn/, *il embrassera*, et le lemme est قبّل /qabbala/ ; ou à سيقبَلون,

Tab. 1.5 : Exemples de segmentation

mot	proclitiques		mot minimal			enclitiques
			préfixe	radical	suffixes	
وسينقذونها et ils le sauvegarderont	وَ wa	سَ sa	ي ya	انقذ nqaδ	و ن ū na	ها ha
وشربوا et ils ont bu	وَ wa			شرب ʂarib	وا ū	
ويلعبان ils jouent (duel)	وَ wa		يَ ya	لعب lʕb	ان ān	
نَزحوا ils ont été déplacés				نزح nazħ	وا ū	
اذهب Va !			أ ʔa	ذهب δhaba		
سافرت elle a voyagé				سافر safara	ت t	
اقتلع il déracine			أ ʔi	قلع ktalaʕa		
فيسبقون donc ils battent	ف fa		ي ya	سبق sabak	و ن ū na	
للعبّاد pour les cultes	ل ال li l			عبّاد ʕubbād		
كمدرّستهم comme leur enseignante	ك ka			مدرست mudarsta		هم hum
مدرّسون enseignants				مدرس mudars	ون ūn	
اشاهد le témoin	ال ʔa_			شهد ʂʂahida		

/sayqabilūn/, *il acceptera*, et le lemme est قَبِلَ, /qbila/. Les deux lemmes sont dérivés de la même racine, mais n'ont pas le même sens.

Par conséquent, l'identification d'une racine, d'un schème ou d'un modèle, même quand la graphie est complète avec les diacritiques, est complexe et fait intervenir une analyse syntaxique en même temps qu'une analyse morphologique. En contraste avec des langues comme l'anglais ou le français, dans lesquelles la segmentation intervient en amont pour permettre l'analyse syntaxique et sémantique, en arabe, l'analyse syntaxique dépend de la segmentation et la segmentation dépend de l'analyse syntaxique.

1.4.1 Clitiques et catégories

Le tableau 1.6 présente la liste des clitiques (Al Hajjar, 2010 ; Alotaiby, Foda et Alkharashi, 2010) qui permettent d'effectuer le premier niveau de segmentation, en les séparant du mot minimal. Parmi les proclitiques, on peut distinguer trois types, en fonction de leur place dans la chaîne. On les appellera PC1, PC2 et PC3. Les enclitiques ne constituent qu'un type, que nous résumerons par EC.

Deux éléments sont à noter tout d'abord. Un certain nombre de mots minimaux ne sont compatibles avec aucun clitique. Nous les appellerons **IS** pour mots isolés ; il s'agit essentiellement d'adverbes. D'autre part, il est possible de trouver une séquence de clitiques sans mot minimal. Dans ce cas, la présence d'un EC et d'un PC2 est obligatoire, les PC3 sont impossibles ; seuls certains PC2 sont possibles (la catégorie *avec nom / rien* dans le tableau). Formellement, on ne devrait donc plus parler de clitiques ici, et la question de savoir quelle est la partie principale (le mot minimal) pourrait se poser. Toutefois, pour réaliser une segmentation correcte, cette question n'a pas besoin d'être résolue.

Les PC1 و, ف sont compatibles avec toutes les catégories de mots minimaux autres que IS (y compris les séquences PC2+EC). Leur présence permet donc simplement d'exclure qu'on soit en présence de la catégorie IS.

Concernant le verbe, le tableau peut laisser apparaître qu'il pourrait y avoir trois proclitiques de suite, mais en fait, il ne peut y en avoir que deux, car مَا et أ sont incompatibles avec ل et س. Par conséquent nous pouvons les positionner en PC2 ou en PC3.

Les EC sont essentiellement des marques de personne, qui s'interprètent comme des pronoms compléments d'objets quand ils suivent un verbe (conjugué), comme des pronoms possessifs quand ils suivent un nom, et comme des pronoms quand ils suivent un PC2 ou une particule. La première personne est la seule à avoir deux formes différentes suivant la catégorie.

Par exemple :

Tab. 1.6 : Clitiques

	Proclitiques					Enclitiques		
	Déb.	Mil.	Fin	Phon.	Sens		Phon.	Sens
avec verbe			لَ	la	futur certain	نِي	nī	pron. objet 1 sg
			سَ	sa	futur possible			
		مَا		mā	ce qui			
avec nom			الْ	ʔal	article			
			لْ	l	autre forme de l'article			
		كَ		ka	comme			
avec substantif/rien		بِ		bi	par / avec / dans	يَ	ī, ya	pron. poss. 1 sg
		لِ		li	pour			
avec P		مَا		mā	ce qui + prédicat			
avec tout	وَ			wa	et	نَا	nā	1 pl
	فَ			fa	alors	كَ	ka	2 sg masc
		أَ		ʔa	interrogatif	كِ	ki	2 sg fém
		لا		lā	non, sans	كُمَا	kumā	2 duel
						كُمْ	kum	2 pl masc
						هُ	hu	3 sg masc
						هَا	hā	3 sg fém
						هُمَا	humā	3 duel
						هُمْ	hum	3 pl masc
						هُنَّ	hunna	3 pl fém
						تُم	tum	3 pl
						نَ	na	3 pl fém

- V : أكلمكم : « je vous parle » ;
- PC2: بكُم : « avec vous » ;
- particule (préposition, conjonction, etc.) : فيكُم : « en vous » ;
- nom : كتابكُم : « votre livre ».

La plupart des clitiques gardent la même forme à la pause, ce qui fait qu'ils sont relativement faciles à séparer. Cependant, l'article défini change de forme dans certains cas (après la préposition ل /li/).

Les participes se comportent, relativement aux clitiques, comme des noms.

Par conséquent les clitiques permettent de définir une catégorisation morpho-syntaxique des mots minimaux, qui s'établit comme suit :

- N : la catégorie des noms (y compris adjectifs, infinitifs et participes), qui ont leur propres PC2 et PC3, mais partagent aussi des PC2 avec la catégorie P ;
- V : la catégorie des verbes conjugués, qui ont leurs propres PC2 et PC3, ainsi que leur propre EC (les autres EC sont partagés par toutes les catégories) ;
- P : la catégorie des particules (prépositions et conjonctions pleines) ; elles ont les mêmes EC que les noms, mais un seul PC2 (qui se trouve être identique avec un PC2 propre aux verbes) ;
- IS : la catégorie des mots isolés.

Le PC2 ما est un cas particulier ; ce clitique transforme un prédicat (qui peut être un verbe ou un pronom ou une particule) en un nom substantif ; il pose un problème particulier pour la segmentation, dans la mesure où bien que sa catégorie (finale) soit un nom, le reste peut se segmenter comme un verbe ou ne pas se segmenter (pronom ou particule). Quand c'est un pronom (lui, elle) ou une particule, la traduction fait apparaître un verbe, mais en ASM les prédicats n'ont pas besoin de verbes : « ce qui + lui » signifie *ce qui est*, forcément suivi d'un qualificatif comme *beau*.

1.5 Morphologie du mot

La plupart des mots minimaux en arabe ont une racine trilitère, c'est à dire une racine constituée de trois consonnes. Quelques mots très courants ont des racines

à deux consonnes, et quelques uns, peu courants, ont des racines de quatre ou cinq consonnes. Sur les 10 000 racines que compte environ l'arabe (Al-Kharashi et M. W. Evens, 1994), 85% sont triconsonnantiques (Slim, 2008).

Entre ces consonnes viennent s'insérer des voyelles, parfois des consonnes, et elles peuvent subir des modifications, c'est ce qu'on appelle le schème. Si on représente par C les consonnes de la racine, il y a le schème **CaCiC**, le schème **CiCCa**, le schème **CCaC**[1], etc. Par exemple le mot دَرَسَ /darasa/ *a étudié* a pour schème /**CaCaCa**/, son dérivé دارس /**dāris**/ (une personne qui étudie) a pour schème /**CāCiC**/. Certains schèmes insèrent des consonnes, par exemple **ʔiCtaCaC** (on peut parler de l'infixe **t**) ; avec ce dernier exemple on voit que la relation entre affixe et schème est serrée, il est parfois difficile de faire la distinction. Pour cette raison on peut aussi intégrer les affixes dans un schème étendu, par exemple le schème **maCCaCa** composé du préfixe **ma** et du schème réduit **CCaCa**.

Les consonnes **y, w, ʔ** posent un problème particulier, ce sont des consonnes faibles, qui tombent dans certains cas, ou se convertissent en voyelles longues. Ces consonnes se comportent normalement si elles sont en début de racine, mais pas en deuxième ou troisième position. Les racines qui possèdent une consonne de ce type dans une position autre que la première sont appelées racines faibles et présentent des schèmes spécifiques. Mais il existe aussi un autre cas de racines faibles, celui dont les deuxième et troisième consonnes sont identiques, car dans ce cas certains schèmes sont impossibles.

Il y a environ 900 schèmes (complets, incluant les affixes) qui se combinent avec les racines pour donner des radicaux. Il existe environ 150 affixes (Alsifayani, 2008 ; Slim, 2008).

1.5.1 Noms et adjectifs

La flexion du nom en arabe classique inclut le nombre et le cas. Mais étant donné que dans toutes les variétés d'arabe parlées aujourd'hui la distinction de cas a disparu, la flexion du cas en arabe standard moderne (de surcroît représentée par des diacritiques) est pratiquement inutilisée ou incorrectement utilisée. Par conséquent, le nom (ainsi que les pronoms démonstratifs et relatifs) se fléchit en nombre (singulier, pluriel, duel) et l'adjectif ajoute le genre (masculin, féminin) dans sa flexion, très semblablement au français.

Le pluriel régulier se forme simplement par l'ajout d'un suffixe. Les pluriels

1. Les grammairiens arabes utilisent des formes du verbe فَعَلَ /**faʕala**/ (faire), par exemple فُعِلَ /**fuʕila**/ désigne le schème **CuCiCa**. Pour les verbes ayant deux consonnes, ils utilisent le verbe imaginaire عل /**ʕl**/, pour ceux ayant plus de trois consonnes, ils redoublent la dernière consonne de فعل autant de fois que nécessaire, par exemple فعلل /**fʕlll**/ pour les racines à cinq consonnes.

irréguliers forment 41% des pluriels (voir la figure 3.1), avec des schèmes de pluriel variables (voir la sect. 3.2.11) (39 schèmes reconnus) et souvent très différents du singulier, ce qui fait qu'il est difficile de relier le pluriel au singulier, et donc de trouver le lemme, par des procédures simples. Par exemple, سَرِيْر /**sarir**/ (*lit*), schème **CaCiC**, est le lemme de أَسِرّة /**ʔasirra**/ (*lits*), schème **ʔaCiCCa**. Le français avec *cheval / chevaux* ou l'anglais avec *feet / foot* présentent quelques difficultés, mais en petit nombre, et la procédure consistant à retirer un *s* fonctionne dans la plupart des cas. Mais en arabe la procédure est beaucoup plus complexe dans beaucoup plus de cas.

1.5.2 Verbes

La langue arabe possède un système morphologique des verbes diversifié et complexe, qui inclut des flexions et des dérivations et pose ses difficultés propres en termes de segmentation. Approfondissons chacun de ces aspects pour mieux comprendre cette complexité.

Les verbes se fléchissent en fonction de l'aspect, du temps, de la modalité, de la personne, du genre et du nombre. Voici quelques exemples :

- Aspect : l'arabe distingue l'inaccompli et l'accompli ; Exemple : حمل (il a porté) /**ħamala**/ (accompli), يحمل (il porte) /**yaħmilu**/ (inaccompli)
- Temps : le passé s'exprime avec l'accompli, le présent et le futur s'expriment avec l'inaccompli, moyennant l'ajout d'un préfixe pour le futur.
  Exemple : حمل (il a porté) /**ħamala**/ (passé), يحمل (il porte) /**yaħmilu**/ (présent), سيحمل (il portera) /**sayaħmilu**/ (futur).
- Personne, genre, nombre. La situation est complexe ; tout d'abord, l'expression de la personne est différente à l'accompli et à l'inaccompli. Il y a trois personnes, qui distinguent toutes le singulier et le pluriel ; le duel n'existe qu'à l'inaccompli, et avec le suffixe انِ (**-āni**). Les deuxièmes et troisièmes personnes différencient le masculin et le féminin.

  Exemples (en se limitant aux deuxième et troisième personnes du singulier) :

Accompli	Inaccompli
شربتَ **ʃaribta** *Tu as bu* (m.)	تشرب **taʃrabu** *Tu bois* (m.)
شربتَ **ʃaribti** *Tu as bu* (f.)	تشربينَ **taʃrabīna** *Tu bois* (f.)
شرب **ʃariba** *Il a bu*	يشرب **yaʃrabu** *Il boit*
شربت **ʃaribat** *Elle a bu*	تشرب **taʃrabu** *Elle boit*

- Modalités : le verbe arabe présente un mode indicatif, dont nous venons de voir des exemples ; il y a également un impératif, un subjonctif et un apocopé, tous dérivés de l'indicatif inaccompli. Les grammairiens répertorient un subjonctif accompli, mais il est morphologiquement identique à l'indicatif accompli.

  L'impératif supprime le préfixe de l'inaccompli et ajoute un préfixe **ʔi, ʔa** si le verbe n'est pas faible (en fait, ce n'est pas réellement un préfixe, mais juste un élément qui rend prononçable la séquence **CC** caractéristique du début du schème de l'inaccompli ; il supprime également la voyelle finale. Exemple : اقْرَأْ (**ʔiqraʔ**) « lis ! ».

  Le subjonctif inaccompli remplace le **u** final de l'indicatif par **a**. Exemple : أَتَمَنَّى أَنْ تَفْهَمَ (**ʔatamannā ʔan tafhama**) « j'espère que tu comprends ».

  L'apocopé, utilisé pour exprimer la négation au passé ou l'interdiction, supprime simplement la voyelle finale de l'inaccompli. Exemple : لَمْ يشرب (**lam yaʃrab**) « il n'a pas bu ».

  Ces modifications sont invisibles dans un texte sans diacritiques, sauf pour l'impératif.

Comme en français, on distingue trois groupes de verbes, en fonction de la voyelle qui figure après la deuxième consonne ; ceux qui ont un accompli en **CaCaCa**, en **CaCiCa** et en **CaCuCa**.

Par ailleurs, à partir d'un verbe, on peut dériver d'autres verbes dont le sens est plus ou moins prévisible, ainsi que des noms ; à partir de noms, on peut dériver des verbes. Ces dérivations s'effectuent avec des affixes et des schèmes particuliers, ce qui implique que les formes résultantes ne sont pas très différentes des formes du verbe de base, surtout sans diacritiques. En voici quelques exemples :

- Verbe de base : شَرِبَ /**ʃariba**/, « boire ».
- Réfléchi : انْشَرَبَ /**ʔinʃaraba**/, qui signifie « se boire, boire pour soi ».
- Réciproque : تَشَارَبَ /**taʃaraba**/, qui signifie « boire ensemble ».
- Intensif : اشْتَرَبَ /**ʔiʃtaraba**/, qui signifie « boire beaucoup ».
- Causatif : أَشْرَبَ /**ʔaʃraba**/, qui signifie « faire boire ».

Sans diacritiques, l'accompli du réciproque est identique à la deuxième personne de l'inaccompli du verbe de base, et l'accompli du causatif identique à l'impératif du verbe de base.

Voici quelques exemples de dérivation de noms (souvent des participes) à partir d'un verbe et réciproquement :

- كتب /**kataba**/ (écrire) forme كاتب /**kātib**/ (écrivain) et كتاب **kitāb** (livre).
- شَرِبَ /**ʃariba**/ (boire) forme شراب **ʃrāb** et مشروب **mʃrūb**, *boisson*, ainsi que شارب **ʃārib** *buveur*.
- مَدْرَسَة /**madrasa**/ (école) forme دَرَّسَ /**darrasa** (enseigner).

1.5.3 Verbes faibles

On appelle verbes faibles (en arabe on emploie le terme de verbe malade) les verbes dont la racine est faible ou qui possèdent quatre consonnes. Il y a beaucoup de modèles différents avec ces verbes, ainsi que des verbes irréguliers. Vous trouverez ci-dessous plusieurs exemples spécifiques qui démontrent ces variations de forme. Mais il y en a beaucoup d'autres.

- Verbes à deuxième consonne faible ; par exemple, la racine قول du verbe *dire*, présente les formes :
  - تَقُولُ **taqūlu**, « tu dis », à comparer avec le schème régulier **taCCaCu**
  - قال **qāla**, « il a dit », à comparer avec le schème régulier **CaCaCa**
- Verbes à troisième consonne faible ; par exemple, نَمَا *croître* ou رَمَى *lancer* :
  - نَمَا **namā**, « il a poussé », à comparer avec le schème régulier **CaCaCa**
  - نُرْمِي **ramā**, « il a lancé », à comparer avec le schème régulier **CaCaCa**
  - يَنْمُو **yanmū**, « il croît », à comparer avec le schème régulier **yaCCaCu**
  - تُرْمِي **turmī**, « tu lances », à comparer avec le schème régulier **taCCaCu**
- Verbes à deuxième et troisième consonnes identiques ; par exemple قَلَّ *diminuer*, présente les formes :
  - يَقِلُّ **tuqillu** « tu diminues », à comparer avec le schème régulier **taCCaCu**
  - قَلَّ **qalla** « il a diminué », à comparer avec le schème régulier **CaCaCa**

1.5.4 Affixes et catégories morphosyntaxiques

Le suffixe ⵜ du féminin est exclusivement réservé aux noms. Il n'est à isoler du radical que pour les adjectifs, qui distinguent le masculin du féminin en s'accordant avec le substantif. En revanche, pour les substantifs, il appartient normalement au radical. Par conséquent, la distinction substantif / adjectif est ici pertinente. Cependant, il existe des substantifs qui peuvent s'accorder avec un autre substantif, et se comportent donc comme des adjectifs ; par exemple « la colonelle Miriam ». Dans ce cas, les deux analyses (ⵜ dans le radical ou en suffixe) sont possibles. Il s'agit de substantifs dénotant un grade, titre, etc., qui existent dans les deux genres, et le substantif qui suit est un nom de personne.

On a vu que les formes verbales de l'inaccompli sont caractérisées par des préfixes, alors que celles de l'accompli n'ont que des suffixes. La distinction accompli / inaccompli est donc pertinente pour l'analyse du mot minimal, de même que l'impératif, en raison de son préfixe spécifique. La distinction actif / passif permet de déduire la voyelle qui suit le préfixe ; elle est donc pertinente pour identifier la voyelle, mais celle-ci étant brève, cette distinction n'a aucune pertinence si le mot n'a pas de diacritiques, ce qui est le cas le plus souvent.

Par ailleurs, du point de vue du radical, les participes (et l'infinitif) ont des affixes en commun avec les verbes, ils se distinguent donc des noms ; mais ils partagent aussi le suffixe du féminin avec les adjectifs, ils forment donc une catégorie à part.

Pour les affixes de dérivation, la situation est plus complexe, et nous devrons donc les écarter de notre champ d'étude.

Les mots isolés et particules n'ont pas d'affixes (du moins en synchronie).

On en déduit que les catégories morphosyntaxiques qui sont pertinentes pour décomposer le mot minimal en radical et affixes recoupent et affinent celles définies par la distribution des clitiques (on omet ici l'actif et le passif, étant donné le faible rôle qu'ils peuvent jouer au niveau de la morphologie quand il n'y a pas de diacritiques).

- N-S : noms substantifs (suffixes de nombre)
- N-A : noms adjectifs (comme les substantifs avec le suffixe du féminin en plus)
- N-AV : participes (partagent les suffixes de nombre et genre avec les noms mais aussi les affixes de dérivation des verbes)
- N-SV : infinitif (pas de préfixes ni de suffixes propres, affixes de dérivation)
- V-0 : verbes conjugués à l'impératif (un seul préfixe, suffixes de l'inaccompli)

- V-I : verbes conjugués à l'inaccompli (préfixes spécifiques, suffixes communs avec impératif)
- V-A : verbes conjugués à l'accompli (pas de préfixe, suffixes spécifiques)
- P : particules (pas d'affixes)
- IS : mots isolés (pas d'affixes).

Seuls les verbes possèdent des préfixes flexionnels, qui servent, pour l'inaccompli, à marquer la personne, comme on vient de le voir ; un préfixe apparaît aussi pour l'impératif dans certains cas, comme support pour la voyelle initiale de l'impératif. Les préfixes flexionnels n'ont pas de voyelle prédéfinie, leur voyelle dépend du radical. Le tableau 1.7 en donne la liste exhaustive.

Tab. 1.7 : Préfixes flexionnels

	préf.	**sens**
VI	أ	1è p. sg
	ن	1è p. plur.
	ت	2è p. + 3è p. fém.
	ي	3è p. masc.
V-0	ا	impératif

Les suffixes flexionnels servent, pour les noms en général, à distinguer le nombre, ainsi que les cas (de manière résiduelle et souvent employés de manière erronée) ; pour les noms adjectifs, ils servent aussi à distinguer le masculin du féminin ; pour les verbes à l'accompli ils servent à distinguer la personne (incluant le genre pour les deuxième et troisième personnes), et pour l'inaccompli et l'impératif à distinguer le nombre et le genre.

Dans la majorité des cas, il n'y a qu'un seul suffixe flexionnel. Dans un seul cas où il peut y avoir deux suffixes flexionnels de suite, c'est, pour un adjectif, quand la marque du féminin est suivie de la marque du duel. Détaillons : le féminin (flexionnel pour les adjectifs et dérivationnel pour les substantifs), en arabe, possède deux formes, une forme liée (ت) qui est adoptée quand un enclitique, un suffixe ou un complément de nom suit la forme, et une forme non liée (ة) utilisée sinon.

Mais l'adjectif ne peut pas avoir d'enclitique ni de complément de nom, sauf s'il est substantivé (et ce n'est plus un adjectif), et par conséquent le seul cas où l'adjectif peut avoir une forme liée, c'est quand il est suivi du suffixe du duel. Le duel est un cas assez rare. Du coup, pour simplifier la description et la programmation, nous proposons de créer, pour ce cas, un suffixe تين /tayain/.

Les suffixes flexionnels sont nombreux et peuvent avoir des fonctions différentes en fonction de la catégorie à laquelle ils s'appliquent. Certains suffixes sont vocaliques et n'apparaissent pas dans l'écriture courante. Aussi dans le tableau 1.8, nous donnons la liste des suffixes qui ont un support consonnantique selon les catégories. Nous n'indiquons pas la fonction, car elle peut être très variée ; par exemple ا (sans hamza) sert à marquer l'accusatif dans les noms, et le duel féminin dans les verbes. Une liste détaillée par fonction serait longue et peu utile pour la segmentation. Nous n'indiquons pas les vocalisations, car elles introduisent des différences qui ne serviront pas à la segmentation.

Tab. 1.8 : Suffixes flexionnels

Suffixe	**Catégories**
ا	N, V
ان	N, V
ين	N, V
و	N, V
ون	N, VI
ات	NA
ة	NA
تين	NA
ى	NA
ن	V
وا	VI, V0
ي	VI, V0
ت	VA
نا	VA
تا	VA
تما	VA
تم	VA
تن	VA

1.6 Types de segmentation

La segmentation peut se faire de plusieurs façons, suivant le degré de sophistication de l'analyse et la quantité de connaissances nécessaires pour l'effectuer. On parle de segmentation simple, de lemmatisation, de racinisation (anglais *stemming*), d'extraction de racine (anglais *root extractor*) et d'analyse morphologique.

La segmentation simple se limite à la première étape de l'analyse des mots : la séparation entre clitiques et mots minimaux. La catégorie morphosyntaxique des mots minimaux est utile pour sélectionner les clitiques.

L'analyse morphologique, pour l'arabe, consiste à disséquer la structure des mots minimaux, avec préfixe, suffixes, racine et schème. La sortie d'un analyseur morphologique est quelque chose comme : « troisième personne du pluriel du verbe dont le lemme est *jouer* », en indiquant le rôle de chaque affixe (ou schème) dans cette analyse. Chaque lemme n'est compatible qu'avec certains schèmes et affixes, en fonction de la catégorisation morphosyntaxique décrite à la section précédente. On parle de modèles d'association entre le lemme, les affixes et les schèmes, par exemple : le passé de telle catégorie de verbe s'effectue avec tel schème. Certains de ces modèles sont à usage unique, comme en français l'association entre *meilleur* et *bon*. Il faut ici remarquer que sans les diacritiques, la plupart des schèmes sont indistinguables, sauf pour les affixes et les voyelles longues (cas des racines faibles). Or restituer les diacritiques est en soi une tâche complexe, totalement non triviale, pour laquelle il n'existe pas de règles connues.

La lemmatisation vise à simplement trouver le lemme correspondant à la forme analysée. Plus simple qu'une analyse morphologique complète, elle a quand même besoin des mêmes données, sauf pour les modèles d'associations, qui seront remplacés par des listes des schémas compatibles avec telle catégorie de verbe ou de nom. La sortie est le lemme (avec la partie du discours à laquelle il appartient).

La racinisation est une traduction malheureuse de l'anglais *stemming* car, contrairement à ce que son nom indique, elle ne cherche pas à trouver la racine du mot. Elle consiste à enlever les affixes ; la sortie est le radical. C'est une version simplifiée de la lemmatisation, qui ignore les infixes. Nous utiliserons donc dans cette thèse le terme « extraction du radical » pour éviter toute ambiguïté.

L'extraction de racine cherche directement la racine d'un mot. Comme, sans diacritiques, la racine est identique au radical dans un certain nombre de cas (nom ou verbe non dérivé), on peut éventuellement considérer cette extraction comme utile dans une partie des segmentations.

Par exemple, pour l'entrée واقتلع, *et il déracine*, waʔiqtalaʕa, la sortie peut être :

- segmentation : وَ ٱقْتَلَعَ, wa-ʔiqtalaʕa ;
- analyse morphologique : troisième personne du singulier accompli de l'intensif du verbe قَلَعَ, qalaʕa, *déraciner* ;
- lemmatisation : lemme قَلَعَ, qalaʕa ;
- « racinisation » : radical قْتَلَعَ, qtalaʕ ; la racinisation n'a pas enlevé le t car il est à l'intérieur du radical.
- extraction de racine : racine قلع, qlʕ.

La racinisation traite les mots minimaux de la même manière que la segmentation simple traite les mots écrits. C'est évidemment la stratégie la moins coûteuse, mais ses résultats sont loin d'être fiables.

Le choix du type de segmentation dépend de l'objectif visé. Par exemple, la lemmatisation a en général simplement pour but de diminuer le nombre de mots et d'augmenter le nombre de contextes ; la racinisation peut être utilisée pour construire un outil de concordance dans de grands textes ; l'analyse morphologique est préalable à une analyse syntaxique et sémantique fine, et la segmentation simple sert à tout. En ce qui concerne cette thèse, nous cherchons à produire une segmentation utilisant le moins de connaissances préalables possible, qui sépare les clitiques du mot minimal et les affixes flexionnels du radical, sans aller jusqu'à l'extraction de racine ni à prendre en compte les affixes dérivationnels. Il s'agit donc d'une segmentation à visée essentiellement grammaticale et syntaxique, laissant de côté l'aspect lexical.

1.7 Conclusion

Dans ce chapitre, nous avons vu que la graphie arabe, dans son usage courant, introduit une multiplicité d'associations possibles entre une forme et un mot (un mot pouvant avoir une centaine de formes et une forme correspondre à un grand nombre de mots) ; pour diminuer ce grand nombre d'incertitudes, la sémantique et la syntaxe doivent être prises en compte. D'autre part, comprendre la morphologie du mot nécessite de recourir à une forme d'analyse syntaxique afin de pouvoir effectuer l'analyse morphologique.

Ces différentes difficultés remettent en cause, pour le traitement de l'arabe, l'ordre habituel des composants de la grammaire d'une langue ; au lieu d'une procédure séquentielle, on est amené ici à tout faire en même temps, du moins dans une certaine mesure. Ceci explique le retard pris dans le traitement automatique de l'arabe par rapport à d'autres langues.

Nous avons également vu que la morphologie, en particulier la morphologie verbale, mais aussi celle des noms, pose, indépendamment de la graphie, des difficultés particulières avec l'intrication des éléments entre eux, ce qui rend impossible une simple segmentation par découpage des éléments, en tout cas quand on dépasse le simple niveau des clitiques.

La conséquence de tout cela est que les techniques standards de segmentation, d'analyse morphologique ou de lemmatisation ne suffisent pas pour l'ASM, qui a besoin, pour sa segmentation, de techniques beaucoup plus avancées. Mais là apparaît une nouvelle difficulté : les techniques avancées, probabilistes ou à base d'apprentissage automatique, nécessitent de disposer d'ensembles de données prétraitées en open-data. Le chapitre suivant sera consacré à cet aspect des choses. Le chapitre trois abordera les techniques de segmentation existantes pour l'ASM.

Chapitre 2

Corpus en langue arabe

Sommaire

2.1 Introduction 45
2.2 Les corpus non annotés d'accès libre 47
2.2.1 SACS 1992-1997 47
2.2.2 Al-Raya 1996 48
2.2.3 Arabic Corpus Project 1999 48
2.2.4 TREC 2001-2002 48
2.2.5 Contemporary Arabic Corpus 2004-2005 49
2.2.6 Abbas & Smaili 2004-2005 49
2.2.7 Adjir Corpora 2005 50
2.2.8 Modern Standard Arabic Corpus 2005 50
2.2.9 Corpus Aljazira 2005 50
2.2.10 Open Source Arabic Corpus (OSAC) 2010 51
2.2.11 University of Jordan Arabic Corpus (UJAC) 2013 . . . 51
2.2.12 KACST 2015 52
2.2.13 Abu El-Khair 2016 52
2.2.14 Tashkeela 2017 53
2.3 Les corpus non annotés d'accès non libre 53
2.3.1 LDC 2001 54
2.3.2 ELRA 2001 54
2.3.3 Arabic Gigaword 2003-2011 55
2.4 Les corpus annotés d'accès libre 55

2.4.1 KALIMAT 2013 56
2.4.2 KSUCCA 2014 57
2.4.3 OSIAN 2019 57
2.5 Les corpus annotés non libres 58
2.5.1 NEMLAR 2003-2005 58
2.5.2 ATB 2003-2011 59
2.5.3 PADT 2004 60
2.5.4 OntoNotes 2013 61
2.5.5 The International Corpus of Arabic (ICA) 2014 61
2.6 Conclusion 62

2.1 Introduction

Un des éléments cruciaux de notre problématique est la difficulté à trouver des données numériques permettant de faire usage de techniques avancées de segmentation. Nous avons besoin d'un corpus, une collection de documents numériques, déjà segmentés, pour calculer les probabilités ou pour entraîner nos algorithmes.

Créer un tel corpus pour l'arabe standard moderne est une tâche difficile en raison du fait qu'il n'existe pas de locuteur natif de l'ASM. Chaque locuteur aura à l'esprit les règles qui sont propres à la variété d'arabe dont il est natif, et être capable d'analyser un texte en ASM demande des compétences particulières. Étant donné le coût en temps et en travail, voire le coût financier, les locuteurs qualifiés auront tendance à réserver le résultat de leurs travaux et à ne pas les mettre en open-data. Pourtant, il faudrait une grande quantité de documents segmentés pour notre objectif.

Ce chapitre passe en revue les corpus existants, leur type et leurs sources. Nous distinguons d'une part entre les corpus d'accès libre (sections 2.4 et 2.2) et non libres (sections 2.5 et 2.3), et d'autre part entre les corpus annotés (sections 2.4 et 2.5) et non annotés (sections 2.2 et 2.3). Par corpus d'accès libre, nous entendons soit de l'open-data, soit des corpus diffusables pour des usages non commerciaux. Par corpus non annotés, nous entendons non pas des corpus sans aucune annotation, mais des corpus n'ayant pas d'annotations linguistiques (sur les formes dans les textes), et en particulier concernant la segmentation et l'étiquetage des parties du discours. Mais certains corpus non annotés en ce sens peuvent tout de même s'avérer intéressants, comme nous le verrons. Pour faciliter le travail d'autres chercheurs, nous nous sommes efforcés d'indiquer pour chaque corpus comment (et si possible où) il peut être obtenu, ainsi que le type de format et de codage utilisés. En effet, l'écriture arabe a été codée de plusieurs façons avant l'apparition d'Unicode ; les codages les plus courants sont le codage Windows CP-1256 et l'UTF-8 (il existe également un codage pour MacIntosh, inexistant dans nos corpus).

Nous nous restreignons aux corpus concernant l'arabe standard moderne, même si nous citons quelques corpus annotés diachroniques, qui pourraient avoir une utilité indirecte (certaines annotations ne changent pas). Nous avons écarté les corpus dialectaux (Almeman et Lee, 2013 ; Graja, Jaoua et Hadrich-Belguith, 2010).

Comme beaucoup de corpus utilisent des articles recueillis auprès des principaux quotidiens, magazines et agences de presse en arabe, nous en faisons la liste dans la table 2.1. Nous n'indiquons que la presse qui autorise pour tous la collecte des archives d'actualités sur son site, mais ce n'est pas le cas pour tous (par exemple, Al Yaum d'Arabie Saoudite et Al Masryalyoum d'Égypte, cependant intégrés dans le corpus Abu-El-Kheir).

Tab. 2.1 : La presse arabe dans le monde

Pays	**Quotidien/Magazine**
Liban	Al-Akbar An-Nahar
Arabie Saoudite	Al-Jazira Hedaya Al-Watan
É.A.U. (Dubai)	CNN
Oman	Al Watan
Qatar	Al-Rayah
Irak	Kol El-Iraq Etthad
Tunisie	Agence Assabah
Égypte	Al Ahram Al Akhbar Gomhoria Akhbar-elryadah Elssyasa Al-Dawli
Bahrain	Akhbar El Khaleej
Jordanie	Addustoor
Palestine	Alhayat Algadidah Al-Quds Al a'maan
Chine	Agence Xinhua
France	Agence France Presse
Angleterre	BBC Al-Hayat Al-Quds Al-Arabic Asharq Al-Awsat

2.2 Les corpus non annotés d'accès libre

Cette section présente les projets de corpus d'accès libre sans annotations linguistiques (ils sont parfois annotés par mots-clés, auteurs, etc.) par ordre chronologique. La table 2.2 en donne un aperçu global, ne mentionnant toutefois que les projets aboutis encore disponibles de nos jours. Quinze projets ont abouti à un corpus plus ou moins grand, un projet n'a pas (encore) abouti (sect. 2.2.3), et un autre a disparu (sect. 2.2.7).

Tab. 2.2 : Corpus libres non annotés

Corpus	Mots	Uniques	Date
SACS	46 968	9 443	1992-1997
Al-Raya	219 978	30 096	1996
TREC	n.a.	n.a.	2001-2002
CAC	842 684	n.a.	2004-2005
Abbas & Smaili	10 000 000	n.a.	2004-2005
MSAC	22 449 469	698 468	2005
Al-Jazeera	408 255	n.a.	2005
OSAC	18 000 000	449 600	2010
UJAC	7 522 941	707 385	2013
KACST	732 780 509	7 464 396	2015
Abu El-Khair	1 525 722 252	3 303 723	2016
Tashkeela	(ASM) 867 913	n.a.	2017

2.2.1 SACS 1992-1997

Le corpus Saudi Arabian National Computer Science Conference[1], créé par Salem (1992), contenait 120 résumés des articles du colloque du même nom, et a été modifié et complété par I. Hmeidi, Kanaan et M. Evens (1997), qui ont ajouté 122 résumés. Il est constitué de 242 résumés des articles du colloque du même nom, jusqu'à l'année 1997, avec 46 968 mots et 9 443 mots uniques. Chaque résumé est étiqueté avec des informations telles que les auteurs, la source, le texte du résumé, les mots-clés. Chaque champ commence par trois caractères qui représentent le nom du champ, suivi d'un espace et du texte associé en format texte.

1. Pour obtenir le corpus SACS, il faut écrire à habusalem@cti.depaul.edu.

2.2.2 Al-Raya 1996

Le corpus Al-Raya[1], créé par Hasnah (1996), est constitué de 187 articles du journal Al-Raya, publiés au Qatar en 1996. Ce corpus contient 219 978 mots au total et 30 096 mots uniques.

Bien que très petits en termes de taille, les corpus SACS & Al-Raya méritent d'être mentionnés, car construits par des chercheurs travaillant avec des systèmes ayant des capacités de traitement limitées. Ils ont été créés à l'Illinois Institute of Technology aux premiers stades des expériences de recherche d'informations en arabe (Abu El-Khair, 2016 ; AbuEl-Khair, 2007).

2.2.3 Arabic Corpus Project 1999

Ce projet proposé par Saleh et Al-Haj (1999) pour l'ALECSO (Organisation de la Ligue arabe pour l'éducation, la culture et la science) avait pour objectif de construire un corpus qui illustre l'emploi réel de la langue arabe dans toutes les situations et à tous les âges. Il était convenu d'incorporer de la littérature, des articles scientifiques et des conférences sur les arts, les sciences et la technologie. Il devait inclure tous les dictionnaires et lexiques arabes, monolingues et multilingues. Autrement dit, le projet devait créer une banque numérique de la langue arabe et de son utilisation effective. Malheureusement, le projet n'a pas abouti jusqu'à présent, à l'exception de la mise en ligne de 400 livres (Saleh et Al-Haj, 2014).

2.2.4 TREC 2001-2002

La série de colloques TREC (Text Retrieval Conference), organisée par le National Institute of Standards and Technology (NIST) des Etats-Unis, porte sur la recherche d'information et a été l'occasion de produire des corpus pour tester différents algorithmes. Les colloques TREC-2001 (Voorhees et Harman, 2001) et TREC-2002 (Voorhees, 2002) ont porté sur l'arabe à partir d'articles de l'Agence France Presse (AFP) Arabic Newswire. La collection TREC-2001 était composée de 383,872 articles publiés entre 1994 et 2000, pour 523 359 mots ; la collection TREC-2002 était composée de 800 000 articles publiés entre 1994 et 2002. Ils étaient encodés en UTF-8. Les documents sont annotés en fonction d'un certain nombre de requêtes (25 pour TREC-2001 et 50 pour TREC-2002). Ces collections ont été largement utilisées pour évaluer les segmenteurs dans le cadre de la recherche d'information. La taille en nombre de mots et en nombre de mots uniques n'est pas indiquée. Ces corpus ne sont plus accessibles aujourd'hui dans la base

1. Pour obtenir le corpus Al-Raya, il faut écrire à inquiry@hbku.edu.qa.

de données TREC[1]. Ils ont été intégrés sous d'autres formes dans d'autres corpus, non libres (ATB par exemple). Nous avons toutefois cité ces corpus car ils sont souvent référencés dans la littérature.

2.2.5 Contemporary Arabic Corpus 2004-2005

Le Contemporary Arabic Corpus[2] a été créé à l'origine dans le cadre du mémoire de maîtrise de Al-Sulaiti (2004), puis complété et approfondi par Al-Sulaiti et E. S. Atwell (2005, 2006). Les sources principales du corpus sont des journaux, des courriers et des sites Web de 1990 à 2004. La dernière version est composée de plus de 843 000 mots ; le nombre de mots uniques n'est pas indiqué.

L'objectif principal de ce corpus était la recherche d'information et l'enseignement de l'arabe pour les locuteurs non natifs. Le corpus est l'un des meilleurs disponibles en raison de son équilibre dans les sujets, comprenant : nouvelles, télévision, journaux, religion, magazines, géographie, sport, éducation.

2.2.6 Abbas & Smaili 2004-2005

Abbas et Smaïli (2005) ont élaboré leur corpus[3] pour l'indexation documentaire ; il est composé de deux corpus bruts au format HTML en CP1256 puis convertis en UTF-8. Le premier est une collection d'articles du journal d'information *Akhbar El-Khaleeg* (Bahreïn,2004), et le deuxième (Abbas, Smaïli et Berkani, 2011, 2009) une collection d'articles du journal *Al-Watan* (Oman, 2004). La première partie du corpus contient plus de 5 690 articles, d'environ 3 millions de mots, répartis sur quatre thèmes : les actualités internationales (953 articles), les actualités locales (2 398 articles), l'économie (909 articles) et le sport (1 430 articles). La seconde partie contient 9,000 articles d'environ 10 millions de mots (nombre de mots uniques non précisé), répartis en six catégories : Culture (2 782 articles), Religion (3 860 articles), Économie (3 468 articles), Actualités locales (3 596 articles), Actualités internationales (2 035 articles) et Sports (4 550 articles). Une étape de pré-traitement des données est nécessaire pour préparer ces textes en vue de la prochaine étape de traitement. Ainsi, chaque article est pré-traité pour supprimer les signes de ponctuation, les chiffres et les mots vides.

1. Voir sur https://trec.nist.gov/data.html, qui semble ne contenir que des corpus en anglais.
2. Le corpus Contemporary Arabic est disponible sur http://www.comp.leeds.ac.uk/latifa.
3. Le corpus Abbas et Smaili est disponible sur https://sourceforge.net/projects/arabiccorpus/.

2.2.7 Adjir Corpora 2005

Abdelali (2005) a proposé un corpus, souvent cité, d'environ 113 millions de mots dans des textes recueillis dans des quotidiens arabes distribués entre 2004 et 2005 et nettoyés. Malheureusement, la page web du corpus n'existe plus (le site web a été abandonné) et il n'y a plus de traces de ce corpus.

2.2.8 Modern Standard Arabic Corpus 2005

Ce corpus brut[1] a été construit par Abdelali, Cowie et Soliman (2005) pour la recherche d'information et le traitement automatique de l'arabe. C'est une collection d'articles de journaux de dix pays arabes, à savoir : Égypte, Koweït, Oman, Algérie, Liban, Arabie Saoudite, Maroc, Jordanie, Qatar, Syrie et Irak. Codé en CP1256 et au format HTML, il contient 102 134 articles, avec 22 449 469 mots et 698 468 mots uniques.

Les journaux sont : Ahram (455 366 mots, 16 569 mots uniques), Alraialaam (1 160 203 mots, 97 580 mots uniques), Alwatan (4 714 199 mots, 122 467 mots uniques), Aps (2 512 426 mots, 52 481 mots uniques), Assafir (3 448 639 mots, 121 911 mots uniques), Jazirah (1 405 083 mots, 84 638 mots uniques), Morocco (3 306 137 mots, 19 092 mots uniques), Petra (989 140 mots, 45 896 mots uniques), Raya (612 409 mots, 55 868 mots uniques), Teshreen (1 467 368 mots, 49 067 mots uniques), Uruk (2 378 499 mots, 32 899 mots uniques).

2.2.9 Corpus Aljazira 2005

Il y a deux ensembles de données d'Aljazeera, collectés pour l'indexation documentaire (S. Mohamed, Ata et N. Darwish, 2005) :

- Aljazeera News Arabic Dataset (Alj-News)[2] : Il s'agit d'une collection de 1500 articles de presse de 2005 de l'agence de presse en ligne Aljazira. Ces articles sont répartis de manière équilibrée entre cinq catégories différentes, et très diversifiées, de taille totale 408 255 mots : économie (68 779 mots), art (86 034 mots), sport (84 371 mots), politique (89 719 mots) et science (79 352 mots). Le nombre de documents dans chaque catégorie est de 300 documents. La taille de l'ensemble d'entraînement est de 1200 documents, et 327 380 mots (240 textes pour chaque catégorie, économie : 54 688 mots, art : 68 554 mots, sport : 68 302 mots, politique : 73 092 mots et science : 62 744 mots), et la taille de l'ensemble de test est de 300 documents, et 80 875

1. Pour obtenir le Modern Standard Arabic Corpus, il faut écrire à aabdelali@hbku.edu.qa.
2. Alj-News est disponible sur http://filebox.vt.edu/users/dsaid/AljNews.tar.gz, .

mots (60 textes pour chaque catégorie économie : 14 091 mots, art : 17 480 mots, sport : 16 069 mots, politique : 16 627 mots et science : 16 608 mots). Chacune de ces cinq catégories correspond à une catégorie de l'ontologie portant le même nom, sauf que la catégorie Science est associée à la Santé dans notre ontologie. Il est encodé en UTF-8.

- Al-jazirah Magazine Arabic Dataset (Alj-Mgz)[1] : Il s'agit d'un ensemble de données en arabe collecté manuellement à partir du journal en ligne Al-jazirah. Cet ensemble de données est plus vaste que le précédent et comprend 4 462 articles publiés entre 2001 et 2005. Les documents de cet ensemble sont répartis de manière inégale entre huit catégories : Arts (406 articles), Culture (205 articles), Economy (283 articles), International-News (653 articles), Local-News (433 articles), Health (204 articles), Society (1 326 articles), Sport (952 articles).

2.2.10 Open Source Arabic Corpus (OSAC) 2010

Le corpus OSAC[2] a été créé par Saad et Ashour (2010b), à partir des sites suivants : BBC en arabe[3] (4 763 articles pour 1 860 786 mots et 106 733 mots uniques après suppression des mots vides), CNN en arabe (5 070 articles pour 2 241 348 mots et 144 460 mots uniques après suppression des mots vides), Aljazeera, Khaleej, celui de la Banque Centrale de Syrie, Altareekh, et quelques autres. Il comprend 22 429 documents, 18 183 511 mots, et 449 600 mots uniques, dans les catégories Économie (3 102 articles), Histoire (3 233 articles), Éducation, Famille (3 608 articles), Religion (3 171 articles), Sports (2 419 articles), Santé (2 296 articles), Astronomie (557 articles), Droit (944 articles), Histoires (726 articles), Recettes de cuisine (2 373 articles).

Tous les articles ont été encodés en UTF-8. Chaque document, codé en UTF-8, est au format HTML et accompagné d'un fichier XML l'annotant par mots uniques et titre.

2.2.11 University of Jordan Arabic Corpus (UJAC) 2013

Le corpus UJAC[4] relativement petit a été créé par Hammo et al. (2013), chercheurs de l'Université de Jordanie. Leur projet était de créer des outils et des ressources

1. Ce corpus est disponible sur http://filebox.vt.edu/users/dsaid/AlgMgz.tar.gz.
2. Le corpus OSAC est disponible sur https://sourceforge.net/projects/ar-text-mining.
3. Le corpus BBC peut être extrait de OSAC par les requêtes figurant sur https://sourceforge.net/projects/queries-for-osac-arabic-corpus/.
4. Pour obtenir le corpus UJAC, il faut écrire à saneyagi@yahoo.com.

pour le traitement automatique de l'arabe. Le corpus comprend des articles de quinze journaux et d'autres ressources de dix-neuf pays. Il comprend 61 037 articles, avec 7 522 941 mots et 707 385 mots uniques. Ce corpus est réparti dans huit catégories : politics (604 882 mots, 65 187 mots uniques), editorials (488 111 mots, 66 571 mots uniques), sports (994 032 mots, 70 514 mots uniques), quran (78 370 mots, 14 648 mots uniques), littérature (552 679 mots, 90 296 mots uniques), constitutions (106 791 mots, 14 139 mots uniques), dictionaries (4 587 594 mots, 365 857 mots uniques), technologie (110 482 mots, 20 271 mots uniques). Il est encodé en UTF-8, en texte plain. Ce corpus est annoté en XML pour l'analyse morphologique.

2.2.12 KACST 2015

Le projet King Abdulaziz City for Science and Technology[1] est l'un des projets stratégiques de l'Arabie Saoudite. Réalisé par A. Al-Thubaity (2015) et A. O. Al-Thubaity (2015), il contient sept cents millions de mots, depuis l'ère pré-islamique à l'époque moderne. Il est collecté dans différents pays et régions, en prenant en compte la nature et le volume de l'activité intellectuelle pour chaque période. En outre, il tient compte de la diversité des supports d'édition, tels que manuscrits, journaux, livres, magazines, périodiques scientifiques, etc. Il est également rassemblé à la fois dans des domaines scientifiques et non scientifiques. Le corpus a un site Web actif sur Internet ainsi que des outils de recherche avec analyse linguistique et statistique. Le corpus contient environ 869 800 sources de texte et d'essais, avec exactement 732 780 509 mots, et 7 464 396 mots uniques. L'une des difficultés ici est qu'il mêle des documents linguistiquement très différents, historiquement (sur plus de mille deux cents ans) et géographiquement.

2.2.13 Abu El-Khair 2016

Ce corpus[2], réalisé par Abu El-Khair (2016), comprend 5 222 973 articles, avec plus de 1,5 milliard de mots et 3,3 millions de mots uniques, codés pour une part en CP-1256 et pour une autre en UTF-8. Il couvre une période de quatorze ans et huit pays. C'est une collection d'articles de neuf journaux : Alriyadh (Arabie Saoudite, 858 188 articles, 271 353 697 mots et 1 451 320 mots uniques), Youm 7 (Égypte, 1 025 027 articles, 261 700 304 mots, 1 020 444 mots uniques), Alyaum (Arabie Saoudite, 888 068 articles, 237 914 494 mots, 1 319 996 mots uniques),

1. Le corpus KACST est disponible sur http://sourceforge.net/projects/kacst-acptool/files/?source=navbar .
2. Le corpus Abu El-Khair est disponible sur https://huggingface.co/datasets/abuelkhair-corpus/arabic_billion_words.

Alqabas (Kuwait, 817 274 articles, 233 741 575 mots, 1 260 511 mots uniques), Alittihad (Émirats, 349 342 articles, 139 962 699 mots, 932 628 mots uniques), Almustaqbal (Liban, 446 873 articles, 135 446 906 mots, 982 765 mots uniques), Tishreen (Syrie, 314 597 articles, 94 695 378 mots, 905 169 mots uniques), Al Masryalyoum (Égypte, 291 723 articles, 93 398 135 mots, 760 511 mots uniques), Echorouk Online (Yémen, 139 732 articles, 40 978 911 mots, 543 799 mots uniques), Saba News Agency (92 149 articles, 16 530 153 mots, 255 098 mots uniques).

Les thèmes couvrent la littérature, les arts, le sport, la religion, les sciences (biologie, géologie, physique-chimie, etc.), les sciences humaines (histoire, psychologie, économie, etc.) et les sciences appliquées (ingénierie, médecine, agriculture), et la culture. Il est donc bien représentatif de l'arabe. L'encodage est en CP-1256 et en UTF-8. Il a été marqué avec deux langages de balisage, à savoir : SGML et XML (en reprenant les balises de LDC, voir sect. 2.3.1). Les annotations concernent les documents (source, dialecte, etc.) et ne sont donc pas linguistiques.

2.2.14 Tashkeela 2017

Tashkeela[1] a été créé par Zerrouki et Balla (2017) à partir de textes dans d'anciens livres libres de droits, principalement des livres classiques islamiques (la majeure partie étant prise dans la bibliothèque Shamela), ainsi que de textes de *The Modern Standard Arabic* tirés du Web, au total 97 ouvrages. Les livres ont été vérifiés et vocalisés manuellement par des volontaires. Son grand intérêt est la vocalisation, mais le corpus est éclectique et mêle différents états de langue. Il contient 74 762 008 mots en arabe classique et 867 913 en arabe standard moderne ; 67 287 202 mots sont vocalisés (998 538 mots uniques, parmi lesquels 770 702 sont partiellement vocalisés).

2.3 Les corpus non annotés d'accès non libre

Les corpus non annotés d'accès non libres regroupés dans le tableau 2.3 sont en petit nombre, et en fait le troisième corpus, Arabic Gigaword, les regroupe tous, même si celui d'ELRA est encore commercialisé séparément.

1. Le corpus Tashkeela est disponible sur https://tashkeela.sourceforge.net.

Tab. 2.3 : Corpus non annotés, non libres

Corpus	Mots	Uniques	Date
LDC	76 000 000	666 094	2001
ERLA (An-Nahar)	24 000 000	n.a.	2001
ERLA (Al-Hayat)	18 639 264	n.a.	2001
Arabic Gigaword^{1nd}	391 619 000	650 000	2003
Arabic Gigaword^{2nd}	481 906 000	1 200 000	2006
Arabic Gigaword^{3nd}	576 799 000	1 500 000	2007
Arabic Gigaword^{4nd}	848 469 000	1 800 000	2009
Arabic Gigaword^{5nd}	1 077 382 000	2 000 000	2011

2.3.1 LDC 2001

Ce corpus a été créé par Graff et Walker (2001) au Consortium de données linguistiques (LDC) de l'Université de Pennsylvanie. Il est composé de 383 872 articles de l'Agence France Presse (AFP) - Arabic Newswire, publiés du 13 mai 1994 au 20 décembre 2000, avec 76 millions de mots et 666 094 mots uniques. Il a été balisé en SGML (style TIPSTER), pour diverses informations non linguistiques, et porté en UTF-8, de taille 209 Mo. Il est peut-être le plus utilisé dans le domaine de la recherche d'informations en arabe depuis longtemps (AbuEl-Khair, 2007).

2.3.2 ELRA 2001

L'ELRA[1] (European Language Resources Association) commercialise deux corpus. Le premier, intitulé An-Nahar Newspaper Text Corpus (W0027 dans le catalogue de l'ELRA) est constitué d'articles du journal libanais An-Nahar de 1995 à 2000. Chaque année contient 45 000 articles et 24 millions de mots. Chaque article est balisé en HTML pour des informations telles que le titre, la date, le pays, la catégorie et la page. Le second, intitulé Al-Hayat Arabic Corpus (W0030 dans le catalogue de l'ELRA), a été réalisé par une équipe de l'Université d'Essex, en collaboration avec Open University. Il contient 42 591 articles du journal Al-Hayat, pour 18 639 264 mots[2], de taille 268 mégaoctets, et répartis en catégories (Général, Voiture, Ordinateur, Informations, Économie, Science et Sport). Ce corpus est nettoyé des balises, des nombres et des caractères spéciaux.

1. Pour obtenir les corpus de l'ELRA, il faut commander à partir de `https://catalog.elra.info/en-us/repository/browse/ELRA-<numéroducorpus>`.
2. Le catalogue parle ici de *distinct tokens*, ce qui est probablement une confusion.

2.3.3 Arabic Gigaword 2003-2011

LDC a étendu son corpus initial pour constituer le corpus Arabic Gigaword[1] , contenant les articles des principaux journaux et agences de presse du monde arabe. Arabic Gigaword, codé en UTF-8 et au format SGML pour des annotations sur les documents et non sur les mots, a eu cinq éditions jusqu'à présent.

1. La première édition (Graff, Kong et al., 2003) incluait le premier corpus de LDC, ainsi que les corpus de l'ELRA, auxquels a été ajouté un corpus en arabe de l'Agence de presse chinoise Xinhua. Le tout comprenait 1 256 719 articles, pour 391 619 000 mots et 650 000 mots uniques.

2. La deuxième édition y ajoute un corpus d'articles de *The Ummah Press*. Le nombre d'articles est passé à 1 591 987 articles, avec 481 906 000 mots et environ 1,2 million de mots uniques.

3. La troisième édition y ajoute des articles d'Assabah (journal tunisien). Le nombre d'articles est passé à 1 994 735 articles, avec 576 799 000 mots et environ 1,5 million de mots uniques.

4. La quatrième édition y ajoutait Al-Ahram (Égypte), Asharq Al-Awsat (Angleterre) et Al-Quds Al-Arabi (Angleterre). Le nombre d'articles est passé à 2 716 995 articles, avec 848 469 000 mots et environ 1,8 million de mots uniques.

5. La cinquième (et dernière édition pour le moment) édition (R. Parker et al., 2011) ajoute plus des articles des mêmes sources que la troisième version . Couvrant neuf sources d'information de six pays différents, le corpus final comprend 3 346 167 articles, avec 1 077 382 000 mots et environ 2 million de mots uniques.

2.4 Les corpus annotés d'accès libre

Nous présentons ici les corpus libres pourvus d'annotations linguistiques, regroupés dans le tableau 2.4.

1. Les différentes versions du corpus Arabic Gigaword peuvent être commandées sur https://catalog.ldc.upenn.edu/LDC2011T11.

Tab. 2.4 : Corpus annotés libres

Corpus	Mots	Uniques	Date
Kalimat	18 167 183	n.a.	2013
KSUCCA	50 602 412	n.a.	2014
OSIAN	925 335 030	16 361 680	2019

2.4.1 KALIMAT 2013

Le corpus KALIMAT[1], réalisé par El-Haj et Koulali (2013), se compose de 20 291 articles de presse collectés à partir du journal *Al-Watan* (Oman), avec 18 167 183 mots, répartis en six catégories : culture (2 782 articles, 1 359 210 mots), économie (3 468 articles, 3 122 565 mots), actualités locales (3 596 articles, 1 460 462 mots), actualités internationales (2 035 articles, 855 945 mots), religion (3 860 articles, 1 555 635 mots) et sports (4 550 articles, 9 813 366 mots).

Chaque mot a été segmenté par l'analyseur Al-Khalil (figure 2.1), puis une analyse a été sélectionnée en fonction du contexte par l'application d'un algorithme de Viterbi (sect. 4.3.2), atteignant une précision de 96% selon eux.

word	vowels	prefix	stem	type	pattern	root	suffix
اشهر	اشْهرْ	#	اشهر	فعل أمر	افعَلْ	شهر	#
تشكيلة	تَشْكِيلَة	#	تشكيلة	مصدر مرة	تَفْعِيلَة	شكل	ة: تاء التأنيث
دورة	دَوْرَةُ	#	دورة	مصدر مرة	فَعْلَةُ	دور	ة: تاء التأنيث
طوال	طِوَالْ	#	طوال	مصدر أصلي	فِعَالْ	طول	#
منتقاة	مُنْتَقَاةِ	#	منتقاة	اسم مفعول	مُفْتَعَاةِ	نقو	ة: تاء التأنيث
نوعيات	نَوْعِيَّاتِ	#	ت	مصدر نوعيا صناعي	فَعْلِيَّاتِ	نوع	ات:تاء التأنيث

Fig. 2.1 : Analyse morphologique de Kalimat

Chaque mot a été également étiqueté par partie du discours, grâce au Stanford POS Tagger (sect. 5.2.10), un étiqueteur linéaire de parties du discours basé sur l'entropie maximale, entraîné sur le corpus PATB P1-3 (sect. 2.5.2). Il identifie 33 parties du discours telles que : nom (NN), verbe (VB), adjectif (JJ). L'étiqueteur a atteint une précision de 96,5% .

De plus, les articles sont étiquetés par entités nommées par le système ANER (Koulali et Meziane, 2012), et chacun est accompagné d'un résumé automatique par Gen-Sum ; les articles ont été groupés en 2057 clusters pour la production de résumé pour chaque cluster.

1. Le corpus KALIMAT est disponible sur https://sourceforge.net/projects/kalimat/, à noter cependant que le nombre de mots est inférieur de moitié à ce qui est annoncé dans l'article.

2.4.2 KSUCCA 2014

Le corpus King Saud University Corpus of Classical Arabic (KSUCCA)[1] a été créé par Alrabiah, Salman et E. Atwell (2014) et Al-Rabiah (2014). Il a été conçu et compilé dans le but de soutenir la recherche en linguistique, en linguistique informatique, en littérature et en histoire. Le corpus comprend 410 articles, et 50,602,412 mots, en texte brut, et il est librement disponible pour les chercheurs. Il se compose de livres sur le patrimoine islamique. Il est divisé en six catégories de sujets, à savoir : religieuse (150 articles, 23 645 087 mots), linguistique (56 articles, 70 939 66 mots), littérature (104 articles, 7 224 504 mots), sciences (42 articles, 6 429 133 mots), sociologie (32 articles, 2 709 774 mots), et biographies (26 articles, 3 499 948 mots). Il a été recueilli à partir de textes classiques du VIIe au XIe siècle.

Il dispose d'annotations morphologiques et d'étiquetage des parties du discours. Mais la langue concernée est l'arabe classique, et non l'ASM ; il est par conséquent inadapté pour notre usage.

2.4.3 OSIAN 2019

Le corpus Open Source International Arabic News Corpus[2] a été créé par (Zeroual, Goldhahn et al., 2019), il a été collecté à partir de six millions de pages Web provenant de 31 plateformes internationales de diffusion de nouvelles en arabe, toutes librement disponibles. Il a été construit en suivant les procédures de la Leipzig Corpora Collection (Goldhahn, Eckart et Quasthoff, 2012). Après nettoyage et filtrage, ils ont conservé environ 3,5 millions d'articles comprenant plus de 37 millions de phrases et 925 335 030 mots. Chaque article est annoté avec des informations de métadonnées, et chaque mot est annoté avec son lemme et sa partie du discours, à l'aide d'un étiqueteur morphosyntaxique indépendant de la langue, Treetagger, moyennant une adaptation à l'arabe (sect. 5.2.4). Cette adaptation a pris en compte le fait que la longueur moyenne des mots est plus courte en arabe (5 lettres) que dans la plupart des autres langues (7 à 12 lettres), selon S. H. Mustafa (2012). Dans le corpus OSIAN, 36% des mots ont une longueur supérieure à six lettres, ce pourcentage atteint 75% si l'on considère les mots dupliqués.

1. Le corpus KSUCCA est disponible sur https://sourceforge.net/projects/ksucca-corpus/.
2. Le corpus OSIAN est disponible en kit sur https://wortschatz.uni-leipzig.de/en/download/Arabic ; il doit être recomposé à partir des requêtes SQL fournies et d'un ensemble de fichiers.

2.5 Les corpus annotés non libres

Tab. 2.5 : Corpus annotés non libres

Corpus	Mots	Mots uniques	Date
Nemlar	500 000	n.a.	2003
ATB1 (Version 2.0)	140 265	n.a.	2003
ATB1 (Version 3.0 et 4.1)	145 386	n.a.	2005-2010
ATB2 (Version 2.0 et 3.1)	144 199	n.a.	2004-2011
ATB3 (Version 1.0 et 2.0)	340 281	n.a.	2004-2005
ATB3 (Version 3.2)	339 710	n.a.	2010
ATB4 (Version 1.0)	161 915	n.a.	2005
PADT	113 500	n.a.	2003
OntoNotes	2 900 000	n.a.	2013
ICA	80 000 000	1 272 766	2014

2.5.1 NEMLAR 2003-2005

NEMLAR[1] (Network Euro-Mediterranean Language Ressources) est un corpus relativement petit. Il se compose d'un demi-million de mots d'articles de treize catégories différentes (nouvelles politiques : 48 000 mots, débats politiques : 30 000 mots, religion : 29 000 mots, phrases courantes : 8 500 mots, nouvelles diffusées : 5 500 mots, affaires : 20 000 mots, littérature arabe : 30 000 mots, nouvelles générales : 100 000 mots, interviews : 56 000 mots, presse scientifique : 50 000 mots, presse sportive : 50 000 mots, entrées de dictionnaire : 52 000 mots, domaine juridique : 21 000 mots). La période des données incluses va de la fin des années 1990 à 2005. Le corpus est fourni en quatre versions différentes (texte brut, texte entièrement vocalisé, texte avec analyse lexicale arabe, texte avec étiquettes de parties du discours (POS) en arabe). Quelques modifications mineures ont été apportées au corpus lors de travaux dans le cadre d'un projet complémentaire à NEMLAR, appelé MEDAR[2] (langue arabe méditerranéenne et technologie de la parole). Les mots sont segmentés par un analyseur morphologique et annotés par partie du discours et vocalisés (Attia, Yaseen et Choukri, 2005).

1. Le corpus NEMLAR peut être commandé pour 300€ sur https://catalog.elra.info/en-us/repository/browse/ELRA-W0042/.
2. Le corpus MEDAR peut être commandé sur https://https://catalog.elra.info/en-us/repository/browse/ELRA-E0040/.

2.5.2 ATB 2003-2011

(Penn) Arabic Tree Bank[1] (Maamouri et Bies, 2004), abrégé en ATB ou en PATB, est la plus grande ressource de ce type pour l'Arabe Standard Moderne annotée avec des caractéristiques morphologiques, des gloses anglaises orientées morphologie et des arbres syntaxiques étiquetés dans le style prédicat-argument du Penn Treebank (M. P. Marcus, Santorini et Marcinkiewicz, 1993). PATB a été réalisé par Maamouri, Bies, Buckwalter et Mekki au Linguistic Data Consortium de l'Université de Pennsylvanie, et a été publié progressivement en plusieurs parties et en quatre versions. Les textes sources sont également distribués indépendamment dans Arabic Gigaword (ci-dessus). PATB a été principalement utilisé pour ses annotations morphologiques et ses formes de mots entièrement vocalisées. Le traitement des données avec des techniques d'apprentissage automatique a donné lieu à un certain nombre d'étiqueteurs morphologiques (Habash et Rambow, 2005 ; Hajič, Smrž, Buckwalter et al., 2005 ; N. A. Smith, D. A. Smith et Tromble, 2005) et de diacritiseurs (Habash et Rambow, 2007 ; Nelken et Shieber, 2005 ; Zitouni, Sorensen et Sarikaya, 2006). Les informations syntaxiques ont été exploitées, notamment pour le parsing (Kulick, Gabbard et M. Marcus, 2006), l'extraction de grammaire (Habash et Rambow, 2004) et l'attribution automatique de cas (Habash, Gabbard et al., 2007). PATB est décrit plus en détail dans Buckwalter (2004) et Maamouri, Bies et al. (2004).

ATB contient quatre corpus, dont voici la présentation :

- ATB1 , dont la version 2.0 (2003) contient 734 textes de l'Agence France Presse, avec 140 265 tokens et 168 123 après séparation des clitiques. Les versions 3.0 (2005) et 4.1 (2010) contiennent 145 386 tokens, 166 068 après séparation des clitiques, et 123 795 mots ;
- ATB2 , dont la version 2.0 (2004) contient 501 textes d'Al-Hayat distribués par l'agence de presse Ummah, avec 144 199 tokens, 168 297 après séparation des clitiques, et 125 698 mots. La version 3.1 (2011) contient le même nombre de tokens, mais 169 319 après séparation des clitiques, et 125 709 mots ;
- ATB3 , dont la version 1.0 (2004) et 2.0 (2005) contiennent 600 textes de l'agence An Nahar avec 340 281 tokens, 400 213 après séparation des clitiques, et 293 035 mots. La version 3.2 (2010) contient 599 textes de même origine avec 339 710 tokens, 402 291 après séparation des clitiques, et 292 554 mots ;

1. Les corpus ATB dans leurs différentes versions peuvent être commandés sur https://catalog.ldc.upenn.edu/LDC2010T08, de 3.500$ pour ATB4 à 4.500$ pour les autres.

- ATB4 , dont la version 1.0 (2005) contient 397 textes du quotidien tunisien Assabah avec 161 915 tokens et 146 491 mots ;

Le corpus ATB contient donc au total 2 231 textes, 791 210 tokens et 688 549 mots. Les mots sont annotés en parties du discours et segmentés par l'analyseur morphologique BAMA (sect. 3.2.7) et le lexique de Buckwalter. Ces annotations ont été manuellement contrôlées. De plus, les trois premières parties sont annotées sous forme d'arbres syntaxiques, conformément aux directives du Penn Arabic Treebank (M. P. Marcus, Santorini et Marcinkiewicz, 1993) développées en 2008 et 2009.

2.5.3 PADT 2004

PADT[1] (Prague Arabic Dependency Treebank) a été créé par Hajič, Smrž, Zemánek et al. (2004), de l'Institut de linguistique formelle et appliquée de l'Université Charles de Prague. Il fournit des annotations linguistiques inspirées de la théorie de la description fonctionnelle (Hajič, Hajičová et al., 2006 ; Hajičová et Sgall, 2003 ; Sgall, Hajičová et Panevová, 1986) : morphologie fonctionnelle, syntaxe de dépendance et représentation sémantique. Le PADT contient plus de 113 500 mots annotés morphologiquement et désambiguïsés manuellement. Dans un premier temps, l'analyseur BAMA avait été choisi (Smrž et Pajas, 2004). Mais pour des raisons de compatibilité théorique avec l'analyse fonctionnelle, un nouvel analyseur travaillant par inférence a été proposé, ElixirFM (Smrž, 2007a,b), voir la figure 2.2.

Form	All~Asilokiy~apu	اللاسلكيّة
Morph	Al + lAsilckiy~ + ap + u	
Tag	DET+ADJ+NSUFF_FEM_SG+CASE_DEF_NOM	
Gloss	the + wireless / radio + [fem.sg.] + [def.nom.]	
Lemma	[lAsilokiy~_1]	لاسِلْكِيّ
Root	*implicit in the lexicon*	

Fig. 2.2 : Analyse morphologique d'un mot dans PADT

L'entrée détaille la forme fléchie complète en translittération Buckwalter, identifie sa structure et la décrit avec des balises et des gloses. Elle fournit également la forme de citation du lexème que la forme de mot représente. D'autres informations pertinentes pour le lexème, comme sa racine dérivationnelle, peuvent être stockées

1. Le corpus PADT v2.0 peut être commandé pour 100$ sur https://catalog.ldc.upenn.edu/LDC2006T01.

implicitement dans le lexique Buckwalter, mais ne sont pas facilement disponibles dans l'arbre de dépendance.

Les données morphologiques et la méthodologie du PADT ont été utilisées pour former des étiqueteurs automatiques (Hajič, Smrž, Buckwalter et al., 2005). PADT a aussi été inclus dans les tâches partagées CoNLL 2006 et CoNLL 2007 sur la syntaxe de dépendance (Hajič, Hajičová et al., 2006) ainsi que dans d'autres expériences d'analyse (Corston-Oliver et al., 2006). Le PADT est discuté en détail dans Hajič et Zemánek (2004), Smrž (2007b), Smrž et Hajič (2008) et Zabokrtsky et Smrž (2003).

2.5.4 OntoNotes 2013

Le projet OntoNotes[1] (R. Weischedel et al., 2013) est le fruit d'une collaboration entre BBN Technologies, l'Université du Colorado, l'Université de Pennsylvanie et l'Institut des sciences de l'information de l'Université de Californie du Sud, dont le but est d'annoter un grand corpus comprenant divers genres (actualités, conversations téléphoniques, blogs, groupes de discussion Usenet, émissions de radio, talk-shows) dans trois langues (anglais, chinois et arabe). La version 5.0 est la version finale du projet OntoNotes, incluant un texte pivot dans les trois langues (la Bible). Elle comprend 2,9 millions de mots dont 1,5 million de mots en anglais, 800 000 mots en chinois et 300 000 mots en arabe.

Les annotations sont syntaxiques, incluant les structures prédicat-arguments en s'appuyant sur les systèmes d'annotation du LDC de l'université de Pennsylvanie (voir ci-dessus 2.3.1), ainsi que sémantiques, avec la désambiguïsation du sens des mots, chaque sens étant connecté à une ontologie, ainsi que la coréférence. Il n'y a pas d'annotation morphologique directe.

2.5.5 The International Corpus of Arabic (ICA) 2014

Alansary et Nagi (2014), des chercheurs de la bibliothèque d'Alexandrie, ont commencé à travailler sur ce corpus en 2006 afin d'aider les chercheurs dans le domaine du traitement de l'arabe. Le plan initial prévoyait 100 millions de mots. Il a été collecté auprès d'un grand nombre de sources afin d'être représentatif, dans onze catégories (sciences stratégiques, sciences sociales, sports, religion, littérature, sciences humaines, sciences naturelles, arts, bibliographie et quelques autres). Le nombre total d'articles dans le corpus s'élève à ce jour à 70 022 articles de quinze journaux (entre 1993 et 2014), avec environ 80 millions de mots et 1 272 766 mots

1. La dernière version du corpus OntoNotes peut être commandée sur https://catalog.ldc.upenn.edu/LDC2013T19, les précédentes sont aussi dans le catalogue.

uniques. Il a été rassemblé sur une période de vingt et un ans, de 1993 à 2014. La plupart des articles proviennent de sites de presse, de blogs et de forums, de livres électroniques, et des articles de recherche et mémoires académiques. Un million de mots représentatifs ont été analysés manuellement pour les tests, et environ 2 millions de mots ont été désambiguïsés manuellement en tant que données d'entraînement, et 50 millions de mots supplémentaires ont été désambiguïsés à l'aide de BAMAE 4.3.1.

Pour l'instant, le corpus n'est pas téléchargeable, il n'est accessible que pour des requêtes en ligne[1]. Les auteurs indiquent qu'ils travaillent sur le problème d'une licence pour ce corpus afin de pouvoir le distribuer.

2.6 Conclusion

Après ce passage en revue, nous constatons que la plupart des corpus libres et une bonne partie des corpus non libres ne contiennent aucune annotation qui pourrait permettre de valider ou d'entraîner un modèle de segmentation. Les corpus annotés morphologiquement sont une minorité (sept au total). Parmi eux, seuls trois sont libres, dont un est à écarter car il ne représente pas l'arabe standard moderne, et l'autre ne contient qu'une annotation morphosyntaxique, ce qui en laisse seulement un, le corpus Kalimat 2.4.1, d'une taille convenable pour ce que nous recherchons, qui possède une segmentation. Mais les annotations ont été produites automatiquement et n'ont pas été ou peu contrôlées.

Du côté des ressources annotées non libres, nous avons quatre corpus : Nemlar 2.5.1, Arabic Tree Bank 2.5.2, Prague Arabic Dependency Treebank 2.5.3, ICA 2.5.5. ATB, PADT sont petits (environ un million de mots en les additionnant), Nemlar 2.5.1, un peu plus grand, reste d'une taille très limitée ; de plus, les corpus ATB coûtent cher. Quant à ICA, son corpus n'est même pas téléchargeable.

Ceci montre bien la faiblesse des ressources disponibles, et la nécessité de construire une ressource libre de taille conséquente, ce qui constitue l'un des objectifs de cette thèse.

1. Pour effectuer ces requêtes sur ICA, il faut être enregistré sur https://www.bibalex.org/ica/ar/Default.aspx.

Deuxième partie

État de l'art

Introduction à l'état de l'art

Dans les chapitres 3 et 4, nous explorerons les différents segmenteurs existants pour l'arabe ; sans prétendre à l'exhaustivité, nous avons cherché à dresser le panorama le plus complet possible ; on pourra utilement compléter notre description avec Soudi, Bosch et Neumann (2007). Le chapitre 5 décrira ensuite les étiqueteurs de parties du discours, car nous avons vu, lors de notre problématique, que les deux processus ont des points communs et peuvent difficilement être complètement séparés.

Les segmenteurs peuvent être classés en fonction de quatre critères principaux : le type de sortie, le type d'entrée, le type d'algorithme et le type de connaissances utilisées.

En ce qui concerne la sortie, elle correspond à l'un des types de segmentation vus au chapitre 1. Les systèmes se répartissent en :

- Segmenteurs simples : ils découpent le mot en trois parties : proclitiques, mot minimal, enclitiques ; en fait, il y a peu d'outils de cette sorte, la plupart sont intégrés dans un analyseur morphologique ou un extracteur de radical.
- Analyseurs morphologiques : ceux-ci fournissent le découpage fin d'un mot en proclitiques, préfixe, radical, suffixes, enclitique, récupèrent le radical et le décomposent en racine et schème, pour fournir le lemme. Ils fournissent souvent aussi une diacritisation du mot, et parfois une traduction.
- Lemmatiseurs : comme un analyseur, ils récupèrent le lemme, mais ne font que cela.
- Extracteurs de racine : comme un analyseur, ils récupèrent la racine, mais ne font que cela.
- Extracteurs de radical : ils découpent le mot en trois parties : pré-radical, radical, post-radical. Certains (rarement) séparent en sous parties (clitiques, affixes) le pré- et le post-radical. L'objectif de la plupart est simplement de fournir le radical.

Tous ces systèmes s'appuient sur la segmentation de mots[1].

Beaucoup de segmenteurs sont conçus pour la recherche d'information (Kharashi et Sughaiyer, 2002 ; J. Xu, Fraser et R. M. Weischedel, 2002), dont les performances

1. Nous avions trouvé référence à une méthode de lemmatisation exceptionnelle sans segmentation proposée par l'entreprise de traduction Sakhr Software Company, sur le site *Ajeeb.com* disparu depuis. Cette méthode s'appuyait sur la traduction du mot en anglais, puis recourrait à un lemmatiseur anglais et enfin traduisait le lemme anglais en arabe. Au vu des différences entre l'anglais et l'arabe pour la sémantique, la morphologie et la syntaxe, cumulant les erreurs de traduction et les erreurs du lemmatiseur anglais, la méthode ne risquait pas d'être très performante.

sont meilleures si un plus grand nombre de formes est regroupé. L'extraction de racine regroupe plus de formes que l'extraction du radical, qui, à son tour, regroupe plus de formes que la lemmatisation ; en arabe, la recherche par radical ou racine est plus efficace que la recherche par mots-clés (Elrajubi, 2013). En revanche, pour l'analyse linguistique, comme étape préalable à l'analyse syntaxique ou sémantique, l'analyse morphologique ou, à défaut, la lemmatisation sont de meilleures solutions.

Le deuxième critère est l'entrée prise en compte par le segmenteur. Il y a des systèmes non contextuels qui prennent en entrée un mot et sont censés renvoyer plusieurs sorties si plusieurs segmentations sont possibles, et des systèmes contextuels qui prennent en entrée un texte (un mot et son contexte) pour trouver la segmentation qui vaut dans ce contexte.

Le troisième critère est le type d'algorithme choisi. Il y a trois grandes catégories d'algorithmes : par règles, probabilistes et par apprentissage automatique.

- Les **algorithmes par règles** fonctionnent à la manière d'un système expert, avec un moteur d'inférence qui analyse le mot en parties à l'aide d'automates à états finis (on parle de *finite state morphology*) ; des experts en linguistique ont élaboré des règles qui prédisent la combinaison correcte de morphèmes (une grammaire de morphèmes). Ce type d'algorithme a un avantage, c'est l'explicabilité. Mais en arabe, en raison de l'intrication de la racine et du schème, ces règles présentent un niveau de complexité supplémentaire par rapport à des langues ayant des structures morphologiques plus régulières, comme l'anglais ou même le français. Elles ont besoin de ressources plus ou moins importantes (au minimum un dictionnaire des clitiques et des affixes) et nécessitent une préparation coûteuse en temps. L'absence de diacritiques est problématique pour la reconnaissance des schèmes (à part pour les racines faibles). De plus, ils sont peu résistants aux fautes, nombreuses dans les corpus, et incapables de donner une solution si une donnée manque dans l'un de leurs dictionnaires. Or, il y a de nombreux mots en ASM qui ne sont pas dans les dictionnaires, en raison de variations régionales, d'emprunts ou de termes spécialisés.

- Les **algorithmes probabilistes** utilisent des modèles probabilistes, tels que les modèles de Markov, pour analyser les structures morphologiques des mots et trouver le découpage le plus probable. Chaque morphème est un état et il s'agit de prédire l'état suivant à partir des précédents, en se basant sur les ngrammes de morphèmes. Les probabilités sont constituées à partir de grands corpus annotés (déjà prédécoupés) ; le prétraitement peut être fait à la main ou généré par règles. Pour que les probabilités soient fiables, il faut que le corpus analysé soit grand. Elles sont plus résistantes aux erreurs et à

la complexité, mais elles peuvent ne pas capturer des variations morphologiques rares. Elles sont dépendantes du corpus initial, qui souffre des mêmes problèmes que ceux mentionnés pour les méthodes par règles.

Cependant, parmi ces algorithmes, la catégorie *letter successor variety*, fondée sur un algorithme de Harris (Harris, 1968) et la probabilité des n-grammes de caractères, mérite une mention spéciale, car elle n'a besoin d'aucun prédécoupage. Mais de tels systèmes ont tendance à sursegmenter les mots, et ils fonctionnent uniquement sur des données linguistiquement correctes, tandis que les entrées réelles foisonnent d'erreurs. Elle n'a jamais été testée sur l'arabe. Une méthode assez proche, travaillant également sur les limites de morphèmes, a en revanche été testée sur l'arabe (section 4.2.1).

- Les **algorithmes à apprentissage automatique** peuvent être utilisés pour apprendre des représentations morphologiques à partir de données d'entraînement. Elles apprennent à diviser les mots selon des exemples qui doivent être correctement segmentés (apprentissage supervisé) ; en dehors des exemples, aucune connaissance n'est normalement nécessaire. D'autres méthodes d'apprentissage peuvent apprendre les états sans exemples (apprentissage non supervisé).

  Ces méthodes fonctionnent un peu comme les modèles probabilistes, mais elles sont davantage capables de généraliser. Elles peuvent apprendre des morphologies complexes. Mais leur fiabilité est, pour beaucoup de modèles, directement liée à la taille et à la composition du corpus d'apprentissage. Enfin, leur interprétabilité pose un problème. Elles trouvent des solutions même pour des mots ou des séquences ne figurant pas dans le corpus de départ, sans qu'on sache comment, ce qui rend délicate la validation lorsque de grandes quantités de données sont traitées.

- Enfin, il existe bien entendu des **algorithmes hybrides**, qui combinent plusieurs méthodes ; par exemple, en intégrant des règles linguistiques avec des modèles statistiques.

Le dernier critère concerne la quantité de données fournies au programme. Les analyseurs morphologiques par règles sont ceux qui ont besoin du maximum de données : listes de clitiques et d'affixes, dictionnaires de lemmes, racines, radicaux, liste de schèmes, et liste de modèles d'association entre schèmes et lemmes (à la manière des groupes de conjugaison du français). Les lemmatiseurs ont aussi besoin de grandes listes de lemmes. Pour les extracteurs (de racine ou de radical), on appelle légers (*light stemming*) ceux qui n'utilisent que des listes de clitiques et d'affixes (et parfois aussi une liste de mots invariables), par opposition à ceux qui utilisent des listes de radicaux ou de racines. Enfin, les segmenteurs à apprentissage

automatique ont simplement besoin d'exemples de segmentation, donc de données beaucoup moins structurées, mais en assez grand nombre.

Dans les chapitres suivants, nous aborderons donc d'abord les segmenteurs non contextuels par règles, puis les segmenteurs complexes, probabilistes et/ou à apprentissage, contextuels et non contextuels, et enfin les étiqueteurs de parties du discours ; cependant, comme ce n'est pas le cœur de notre problématique, notre état de l'art à ce sujet sera beaucoup moins fouillé.

Chapitre 3

La segmentation par règles

Sommaire

3.1 Introduction 73
3.2 Segmenteurs 73
3.2.1 ALPNET 76
3.2.2 Al-Shalabi et Evens 77
3.2.3 Khoja 78
3.2.4 Morpho3 80
3.2.5 ISRI 80
3.2.6 Aljlay & Frieder 83
3.2.7 AraMorph (BAMA, SAMA) 84
3.2.8 Light10 (Al-Stem) 86
3.2.9 Light10 → Elrajubi 88
3.2.10 Light10 → CondLight 89
3.2.11 Goweder 91
3.2.12 Kadri & Nie 92
3.2.13 Al-Ameed 94
3.2.14 Morph2 96
3.2.15 ElixirFM 99
3.2.16 Bigrammes 100
3.2.17 Alkhalil 104
3.2.18 SARF 105
3.2.19 Tashaphyne 107

3.2.20 Zaraket 110
3.2.21 Assem 111
3.2.22 ARAMO 112
3.3 Conclusion 115

3.1 Introduction

Nous abordons dans ce chapitre les segmenteurs par règles, qui sont tous des segmenteurs non contextuels.

Parmi les algorithmes utilisés en général dans la segmentation par règles, on notera l'algorithme de Greedy Lemmatizer (Schmid, 1995) (utilisé par exemple pour le français), l'algorithme de Floresta (Afonso, Moreira et Santos, 2006) (pour le portugais), l'algorithme de MeCab (Kudo, Yamamoto et Matsumoto, 2002) (pour le japonais), l'algorithme de lemmatisation de FreeLing (Civera, Martínez-Barco et Garcia-Serrano, 2011) (pour l'espagnol), l'algorithme de HanLP (Han, Sun et Liu, 2019), spécialisé pour le chinois, et l'algorithme de Porter (M. F. Porter, 2001) (anglais, russe, allemand), et l'un des plus connus, Kimmo (Koskenniemi, 1983). Seul l'algorithme de Porter a été utilisé pour l'arabe, la plupart des segmenteurs créent leurs propres automates, à cause des contraintes particulières de l'arabe. On relèvera parmi eux trois algorithmes utilisant des graphes acycliques orientés (*Directed Acyclic Graphs*, DAG) (Blumer et al., 1985), et des algorithmes utilisant des arbres préfixes (*prefix tries*; voir ALPNet et AlKhalil).

Ce n'est que très récemment qu'une méthodologie de construction d'analyseurs morphologiques pour l'arabe a été proposée par Habash, Marzouk et al. (2022), presque quarante ans après la proposition équivalente pour les langues à morphologie plus simple (Koskenniemi, 1983). Cette méthodologie est mise en œuvre dans la plateforme CamelMorph (incluse dans Camel Tools[1]), équivalent donc pour ce type de langue de l'outil PC-Kimmo.

Concernant l'implémentation, elle se fait en général en Java ou en Python, avec une exception utilisant le langage Snowball (segmenteur Assem, sect. 3.2.21). Snowball est un langage de traitement de chaînes développé spécifiquement pour la lemmatisation à base de règles, incluant une bibliothèque de procédures *ad-hoc*. Il a été initialement conçu et développé par M. F. Porter (2001), et est maintenant maintenu en tant que projet communautaire[2]. Un compilateur traduit les programmes Snowball dans d'autres langages : actuellement Ada, ISO C, C#, Go, Java, Javascript, Object Pascal, Python et Rust sont pris en charge.

3.2 Segmenteurs

Nous avons répertorié vingt-quatre algorithmes, listés dans le tableau 3.1. À l'exception de ALPNET, de SARF, de BAMA (SAMA) et ARAMO, ils sont presque tous en open-source ou en diffusion libre non commerciale (accessibles auprès de

1. Ces outils open-source sont disponibles sur https://github.com/CAMeL-Lab/camel_tools.
2. Les sites https://snowballstem.org/ et https://github.com/snowballstem.

l'auteur) ; le code de certains n'est plus disponible ; le statut du code de Goweder et de Bigrammes est flou. Nous avons pris comme point de départ 1992, l'année où est publié un ouvrage fondamental définissant les attendus et les principes de l'analyse automatique de la morphologie en général (Sproat, 1992), car les algorithmes antérieurs sont encore balbutiants (et peu nombreux). Sur les vingt-quatre segmenteurs, neuf sont des analyseurs morphologiques (ALPNET, SARF, Shalabi-Evens, Morpho3, Aramorph, Morph2, AlKhalil, Zaraket, Aramo), dont les deux premiers avaient un code propriétaire et sont aujourd'hui disparus, tout comme l'analyseur Shalabi-Evens ; le dernier est en fait un analyseur restreint à certaines catégories.

Voici la signification des codes dans le tableau concernant les listes utilisées comme ressources (la définition précise des listes peut varier) :

- M : mots invariables ou *stopwords* ;
- $C(-)$: clitiques, $-$ quand tous ne sont pas inclus ;
- $A(-)$: affixes, $-$ quand tous ne sont pas inclus ;
- S : schèmes, avec éventuellement leur analyse ;
- $R(-)$: racines, $-$ quand il n'y a que les racines faibles) ;
- L : lemmes ;
- I : mots irréguliers avec leur racine, leur lemme ou leur radical.
- $Tout$: M, C, A, S, R, L, I, + modèles de combinaison lemmes-schèmes ;

Dans ce qui suit, nous nous sommes efforcés de retrouver et de mettre en forme les algorithmes utilisés par chaque segmenteur, ce qui a parfois nécessité d'analyser le code source quand il était disponible. De cette façon, le chercheur dispose d'une base de comparaison représentative et presque exhaustive, que nous compléterons par la suite (chapitre 4).

Les algorithmes présentés ne concernent que la partie segmentation, même quand le programme inclut d'autres fonctionnalités. La présentation suit globalement l'ordre chronologique ; toutefois Light10, Elrajubi et CondLight sont trois variantes du même algorithme, évoluant entre 2002 et 2018, et sont donc placés ensemble ; de même pour Aramorph, BAMA et SAMA.

Comme déjà signalé, toutes les méthodes suppriment le taṭwīl (prolongation de caractère, sect. 1.3) ; la suppression des signes de ponctuation est aussi commune à tous, ainsi que l'élimination des chaînes de caractères autres que l'arabe. Donc nous ne l'indiquons pas dans les algorithmes, pour simplifier la lecture.

Les segmenteurs à base de règles effectuent tous une normalisation orthographique (page 25), sous une forme ou sous une autre.

De nombreux auteurs ne font pas la différence entre clitiques et affixes. Très souvent, ils considèrent les séquences de proclitiques comme un seul proclitique,

Tab. 3.1 : Segmenteurs à base de règles

Segmenteur	Output	Ressources	Date
ALPNET	Racine, schème + tiroir	Tout	1989-1996
Shalabi-Evens	Racine, schème + tiroir	Tout	1998
Khoja	Racine	R, A, S, M	1999
Morpho3	analyse	Tout	2000
ISRI	Racine	A, C- (M v2, R- v3)	2001
Aljlayl-Frieder	Racine (lemme v.2)	A, C-	2002-2016
AraMorph	Mot vocalisé, lemme, découpage, glose	Tout	2002
AraMorph → BAMA	Mot vocalisé, lemme, découpage, glose	Tout	2004
BAMA → SAMA	Mot vocalisé, lemme, découpage, glose	Tout	2010
Light10 (Al-Stem)	Radical	AC-, M	2002-2007
Light10 → Elrajubi	Radical	AC-, M, I	2013
Light10 → Cond-Light	Radical	AC-, M	2018
Goweder	Découpage, Lemme	R, A, S, M	2004
Kadri & Nie	Radical	Tout	2006
Al-Ameed	Racine, radical	A, C, S, R	2006-2010
Morph2	Mot vocalisé, racine, radical, lemme, schème, POS	Tout	2006-2010
ElixirFM	racine, lemme, découpage, schème, vocalisation	Tout	2007
Bigrammes	Racines	C, A, R	2009
AlKhalil	Mot vocalisé, racine, radical (lemme + schème v2)	Tout	2010-2017
SARF	Mot vocalisé, racine, radical, lemme, schème	Tout	2010-2015
Tashaphyne	Radical		2010-2023
Zaraket	Affixes, radical, catégorie	Tout	2012
Assem	Radical	C, A	2018
ARAMO	Analyse des verbes	Tout	2022

Segmenteur : licence commerciale ; **Segmenteur** : diffusion limitée aux usages non commerciaux, sinon : open source ; Date : code introuvable.

et de même pour les séquences de suffixes, voire pour les séquences proclitiques + préfixes. Cette stratégie augmente la taille du dictionnaire mais diminue les calculs nécessaires.

Enfin, il y a une étape que nous ne mentionnons pas dans les algorithmes, parce qu'elle ne joue plus un rôle crucial depuis l'adoption généralisée d'Unicode, mais qui était fréquente avant, consistant à translittérer l'arabe en caractères latins (en utilisant, très souvent, le système de translittération de Buckwalter).

3.2.1 ALPNET

Dans la première moitié des années 1990, le centre de recherche de Xerox de Grenoble a élaboré le segmenteur ALPNET (Beesley, 1996) ; il s'agit d'un analyseur morphologique complet, dont le code source était propriétaire et commercialement exploité, mais plus disponible aujourd'hui. Les racines (environ 500) sont organisées en *tries* (arbres préfixes), séparées par catégories ; les schèmes (plusieurs centaines) également ; ils forment environ 85.000 radicaux. Les règles sont représentées dans une centaine d'automates à états finis, qui, dans la version publiée en 1996, ont été intersectés pour former un petit nombre d'automates grâce au *Xerox finite-state calculus*.

Algorithm 1 : ALPNET

Input : mot arabe w
Output : racine (lemme), schème
Data : $A \leftarrow$ affixes ; $R \leftarrow$ racines ; $S \leftarrow$ schèmes ; $M \leftarrow$ Mots invariables $P \leftarrow$ Patterns

```
1  Function Stem(w) :
2      Automate de nettoyage du mot ;
3      Automate de reconnaissance du mot ;
4      Application des règles de génération des voyelles (diacritiques) ;
5      Règles FST : /* Désintrication d'une racine et d'un schème, exemple */
6      if intersect w ∧ scheme = CaCaC then
7          return racine + schème
8      /* Idem, si le mot contient des diacritiques */
9      if w contain voyelle ∧ scheme = CVCVC ∧ V = ā then
10         return racine + schème
11     Automate générant la variante correcte de la racine ;
12     Trouver le tiroir correspondant au schème ;
13     return racine corrigée + schème + tiroir
```

Il n'y a malheureusement aucune information sur les règles utilisées, seulement sur les techniques utilisées.

3.2.2 Al-Shalabi et Evens

Le système d'analyse morphologique développé par R. Al-Shalabi et M. Evens (1998) constitue l'une des premières tentatives informatisées de lemmatisation de l'arabe fondée sur une approche linguistique structurée. Inspiré de l'algorithme de Fedaghi et Al-Anzi (1989), leur méthode (voir l'algorithme 2) repose sur l'identification des racines consonantiques, trilittères et quadrilittères, et sur l'association de ces racines à des schèmes pour reconstruire les lemmes. Contrairement aux approches entièrement basées sur des bases de données, ils ont intégré leur propre algorithme de découpage et de polarité pour gérer les désinences (suffixes, marques de genre, nombre, etc.), ce qui leur permet de traiter les formes fléchies avec plus de flexibilité.

Le système utilise une base lexicale préexistante pour la récupération des racines, mais enrichit cette ressource par une analyse algorithmique des formes complexes. L'évaluation a été réalisée sur un corpus de résumés de conférences nationales saoudiennes, et les résultats ont montré une précision de 90,57% dans la tâche de lemmatisation. Ce taux de réussite, remarquable pour l'époque, témoigne de la robustesse du système notamment dans le traitement des formes quadrilittères et des verbes irréguliers.

Son application reste très limitée, car il faut lui fournir des mots corrects pour lesquels on dispose des racines et des schèmes dans la base. Il va détecter le radical, puis chercher les consonnes de la racine, extraire la flexion, en déduire le schème et le chercher dans sa base pour trouver le lemme correspondant.

Algorithm 2 : Al-Shalabi & Evens

Input : Mot w
Output : Racine r, schème s, préfixe p, suffixe f
Data : $A \leftarrow$ affixes ; $R \leftarrow$ racines ; $S \leftarrow$ schèmes ; $L \leftarrow$ lexique $P \leftarrow$ Patterns

```
Function Stem(w) :
    Normaliser w (suppression des diacritiques, substitutions) ;
    Vérifier si w ∈ L ;
    Identifier les préfixes p et suffixes f à partir de A ;
    Extraire le radical N tel que N = w − p − f ;
    Function ExtractRoot(N) :
        Tester si N contient une racine trilittère ou quadrilittère ;
        Comparer les combinaisons possibles avec r ;
        Retourner la racine r ;
    Function IdentifyPattern(N) :
        Vérifier si N correspond à un schème de S ;
        Déduire la catégorie grammaticale ;
        Retourner le schème s ;
    Choisir l'analyse la plus probable selon le contexte ou la fréquence
     Retourner (r, s, p, f) ;
```

3.2.3 Khoja

Shereen Khoja a développé l'un des segmenteurs les plus connus, probablement le plus référencé, mais qui n'a fait l'objet d'aucune publication officielle, seulement connu par un rapport technique (Khoja et Garfield, 1999) réalisé pendant sa thèse et par un code source[1]. Il a également été porté dans Weka (Hall et al., 2009) par Saad et Ashour (2010a). C'est une extraction de racine, à cheval entre lemmatisation et segmentation.

L'algorithme de Khoja (alg. 3) vérifie que le mot n'appartient pas à la catégorie des mots invariables (prépositions, pronoms, etc.), supprime un clitique (l'article défini), puis supprime les affixes les plus longs possible. Par exemple, dans le mot يلاعبون (*ils jouent avec*), l'algorithme élimine le préfixe ي et le suffixe ون. Le radical potentiel qui en résulte est mis en correspondance par pattern-matching, en fonction de son nombre de lettres, avec un schème (ici ce serait CāCC). À partir de ce schème, on génère une racine potentielle de la forme CCC et on vérifie qu'elle existe bien à l'aide d'un dictionnaire de racines. Si le processus n'aboutit pas, on extrait des affixes moins longs, et on recommence. Les schèmes pris en compte sont sans diacritiques, ils représentent donc un très petit sous-ensemble de schèmes (ceux

1. Le code source de Khoja est disponible sur le site de l'université du Pacifique (Oregon), http://zeus.cs.pacificu.edu/shereen/ArabicStemmerCode.zip.

avec une voyelle longue essentiellement).

La procédure marche assez bien avec des racines fortes (M. Mustafa et al., 2017), elle est plus complexe avec les verbes faibles, car on n'aura que deux consonnes, qui correspondront à plusieurs verbes et la troisième lettre devra être générée à partir des différentes racines trouvées. La seule différence avec une lemmatisation complète, c'est qu'on ne vérifie pas si la racine trouvée est compatible avec ce schème.

Algorithm 3 : Khoja Stemmer

Input : mot arabe w
Output : sa racine
Data : $A \leftarrow$ affixes ; $R \leftarrow$ racines ; $S \leftarrow$ schèmes ; $M \leftarrow$ Mots invariables
Function `Stem`(w) **:**
  Normalisation
    Supprimer les diacritiques de vocalisation ;
  if $w \supset$ *nombres* **then**
    return Null ;
  if $w \in M$ **then**
    return Null ;
  Supprimer l'article défini (clitique) ;
  Supprimer les suffixes du plus long au plus court ;
  Supprimer les préfixes ;
  if *w matche avec un des schèmes de S de même longueur que w* **then**
    Extraire le schème trouvé de w pour trouver sa racine ;
  if *racine* $\in R$ **then**
    return racine ;
  Remplacer les lettres faibles de w par و ;
  Remplacer les variantes de hamza par ء ;
  if $w \in R$ **then**
    return w ;
return `Stem`(w) ;

Cet algorithme a été utilisé dans le cadre d'un système de recherche d'informations de l'Université du Massachusetts pour la session *cross-lingual track* TREC-2001 (sect. 2.2.4) du laboratoire NIST. Les auteurs (Larkey et Connell, 2001) rapportent que, bien que l'algorithme ait produit de nombreuses erreurs, il a considérablement amélioré les performances de leur système ; ces résultats ont été approfondis dans Larkey et Ballesteros (2002).

3.2.4 Morpho3

Attia (2000) a proposé l'analyseur morphologique MORPHO3[1], incluant également un extracteur de racine dans sa thèse de doctorat, qui a constitué une avancée notable. Conçu comme un analyseur à large couverture, il combine des règles linguistiques avec des statistiques lexicales pour générer les combinaisons possibles de préfixes et de suffixes, permettant ainsi d'extraire les racines et de produire une analyse morphologique complète (alg. 4), avec racine, radical, schème, glose des affixes et catégorie grammaticale.

Les résultats expérimentaux rapportés dans la thèse montrent qu'il atteint une précision globale de 94% sur des corpus arabes contemporains, avec un taux de reconnaissance des racines supérieur à 90%, y compris pour les formes quadrilittères et irrégulières. Le système couvre plus de 95% des mots d'un corpus généraliste sans erreur de segmentation, et propose plusieurs interprétations en cas d'ambiguïté morphologique, pondérées selon leur fréquence d'usage. Ces performances témoignent de la robustesse du système.

Algorithm 4 : MORPHO3

Input : Mot arabe fléchi w
Output : Analyse morphologique complète $(r, s, \text{affixes}, \text{cat})$
Function `Stem(`w`)` **:**
  Suppression des diacritiques ;
  Déterminer les préfixes, radicaux et suffixes potentiels ;
  Générer toutes les combinaisons possibles préfixes et suffixes ;
  Éliminer les combinaisons non valides par des règles morphologiques ;
  Identifier la racine r en extrayant le schème s du radical ;
  Vérifier la validité morphologique via la base lexicale ;
  Déterminer la catégorie *cat* d'après les affixes ;
  Annoter les affixes avec leur glose ;
  return $(r, s, \text{affixes}, \text{type})$

3.2.5 ISRI

Taghva, Elkoury et Coombs (2005), chercheurs de l'ISRI (*Information Science Research Institute*) de l'Université du Nevada, ont conçu en 2001 un segmenteur qui, contrairement à celui de Khoja, n'utilise pas de dictionnaire de racines, ce qui le rend plus rapide, et normalise la sortie même si un mot n'a pas été analysé.

Cet algorithme (alg. 5) a été porté par Hosam Algasaier dès 2001 dans la bibliothèque Python open-source NLTK[2]. La version intégrée a été améliorée par rapport

1. Pour obtenir le code de Morpho3, il faut écrire à `dinasaidelashqar@gmail.com`.
2. Isri est disponible sur `https://www.nltk.org/_modules/nltk/stem/isri.html`.

à la version initiale, avec un dictionnaire de 60 mots invariables, en ajoutant un cas de mot de longueur 6 qui n'avait pas été prévu, et en réservant la normalisation du hamza à l'initiale de mot (sinon l'ambiguïté augmente et provoque des erreurs de racine). Préfixes et suffixes sont confondus avec les proclitiques et les enclitiques. Certains clitiques moins fréquents ne sont pas inclus dans les listes. Les séquences de clitiques sont considérées comme un seul clitique (« préfixe »).

H. Algasaier a proposé un postraitement (la fonction *Post* à la fin de l'algorithme 5), dans le cas où la racine ne fait que deux lettres. Dans ce cas, on contrôle si elle s'identifie à une des racines ayant une lettre en double (deux des trois consonnes sont identiques), on vérifie si la consonne qui manque est la première, la deuxième ou la troisième, à partir d'une liste de racines ayant justement une lettre faible dans cette position. La liste des racines faibles reste beaucoup plus petite que la liste complète des racines (par ex. Khoja).

Algorithm 5 : ISRI Stemmer

Input : mot arabe w
Output : sa racine
Data : $A \leftarrow$ affixes, $M \leftarrow$ mots invariables, $R- \leftarrow$ racines faibles, $C- \leftarrow$ clitiques

Function `Stem(`w`)` **:**
  $w \leftarrow w$ sans diacritiques ;
  if $w \in M$ **then**
    return w
  Supprimer préfixes de longueur trois puis ceux de deux ;
  Supprimer suffixes de longueur trois puis ceux de deux ;
  Supprimer la conjonction و si le mot après w commence par و ;
  Normalisation
    Remplacer au début de w {أ آ إ} par ا ;
    Remplacer en début de w {ؤ ء ئ} par أ) ;
  if *longueur(w) = 4* **then**
    Enlever un préfixe, un suffixe ou une lettre faible de w ;
  else if *longueur(w) = 5* **then**
    Enlever préfixes, suffixes, infixes et lettres faibles ;
    if *longueur(w)* $= 4$ **then**
      go to 11
  else if *longueur(w) = 6* **then**
    Enlever préfixes, suffixes, infixes et lettres faibles ;
    if *longueur(w)* $= 4$ **then**
      go to 11
    else if *longueur(w)* $= 5$ **then**
      go to 13
  else if *longueur(w) = 7* **then**
    Enlever un suffixe de longueur 1;
    if *longueur(w) = 7* **then**
      Enlever un préfixe de longueur 1;
    if *longueur(w) = 6* **then**
      go to 17
  return w ;
Function `Post(`w`)` **:**
  if *longueur(w) = 2* **then**
    Comparer w avec les listes de racines faibles ;
    return la racine qui a les mêmes consonnes fortes que w ;
  return `Stem(`w`)` ;

3.2.6 Aljlay & Frieder

De manière indépendante, Aljlayl et Frieder (2002) proposent un algorithme très similaire[1] ; l'article ne fait pas référence à la méthode ISRI dont seul le code avait été diffusé l'année antérieure dans NLTK. Leur méthode est basée sur la normalisation orthographique suivie de la suppression des affixes tant que la longueur du mot est supérieure à trois caractères, et n'utilise pas de dictionnaire de racines (alg. 6). Les affixes les plus fréquents utilisés les « préfixes » incluent en fait aussi des proclitiques courants ; les deux proclitiques les plus fréquents, l'article défini et la conjonction de coordination (sect. 1.4) sont eux supprimés avant les « préfixes ».

Algorithm 6 : Light Stemming for Arabic Words (Aljlayl & Frieder)

Input : mot arabe w
Output : sa racine
Data : $A \leftarrow$ affixes (préfixes, suffixes) ; $C \leftarrow$ clitiques (restreints)
Function `Stem(`w`)` **:**
  Normalisation
    Supprimer les diacritiques ;
    Remplacer au début de w {أ ٱ إ} par ا ;
    Remplacer à la fin de w ى par ي ;
    Remplacer à la fin de w la combinaison ئ par ىء ;
    Remplacer à la fin de w ه par ة ;
  Suppression clitiques
    if *longueur(*w*)* ≥ 3 *caractères* **then**
      Supprimer au début de w le clitique "و" (wa) ;
    $Stem \leftarrow w$;
    forall $V \in$ *variantes de l'article défini* **do**
      if $V \subset w$ **then**
        $V = V +$ Caractères dans w après V ;
        $Stem = w$ - V ;
        Remplacer au début de $Stem$ أ ٱ إ par "ا" ;
  Suppression suffixes
    while *longueur(Stem)* ≥ 3 *caractères* **do**
      $Stem = Stem$ - le plus long suffixe possible ;
  Suppression préfixes :
    while *longueur(Stem)* > 3 **do**
      $Stem = Stem$ - le préfixe éventuel ;
  return `Stem(`w`)` ;

Quelques années plus tard, Aljlayl et Frieder (2016) ajoutent à leur segmenteur dans sa version 2 une procédure spécialisée sur les pluriels, à la suite de Goweder et

1. Pour obtenir le code, il faut écrire à ophir@ir.iit.edu.

al. (sect. 3.2.11), mais sans lexique, contrairement à eux. Ils utilisent la catégorie grammaticale des mots, les suffixes et un dictionnaire de schèmes de pluriel (39 schèmes). À la fin de la procédure précédente, quand il s'agit d'un nom (détecté s'il y a l'article défini ou un suffixe nominal et pas de préfixe verbal), l'algorithme commence par vérifier si le mot suit l'un des schèmes de pluriel. Si oui, il applique une série de règles de transformation pour générer la forme singulière du mot. Ils renvoient le singulier du nom, donc son lemme. L'évaluation de l'algorithme a montré qu'il était capable de lemmatiser avec succès une grande proportion de pluriels irréguliers en arabe.

Ils ont évalué leur méthode sur les collections et requêtes de référence **TREC** décrites à la sect. 2.2.4, avec trois mesures de performance : le rappel et la précision sur les onze points de rappel, ainsi que la moyenne de la précision (*mean average precision*) sur les n premiers documents récupérés. Pour eux, le rattachement à une racine fonctionne mieux que le rattachement à un lemme, car il regroupe une plus grande variété de formes en peu de classes.

Avec les titres et la description des sujets de **TREC-10**, leur algorithme atteint 0,3715 de précision moyenne, améliorant de 87,7% la lemmatisation par rapport à l'algorithme de référence. En expansant par *relevance feedback* (les 15 termes les plus pertinents parmi les 10 premiers documents sont ajoutés à la requête originale), il atteint 0,4312 en précision moyenne, améliorant de 71,3% la lemmatisation par rapport à la référence du concours.

3.2.7 AraMorph (BAMA, SAMA)

Après avoir travaillé sur ALPNET, Buckwalter et J. Parker (2002) élaborent, au Linguistic Data Consortium de l'Université de Pennsylvanie, leur propre segmenteur, AraMorph[1], dont la deuxième version sort en 2004 (Buckwalter, Tim, 2004). À partir de la version 2.0, il prend officiellement le nom de BAMA[2] (Buckwalter Arabic Morphological Analyzer) et n'est plus en open-source, ainsi que son successeur SAMA (Standard Arabic Morphological Analyzer)[3], qui sort en 2010. Peu de détails sont donnés sur SAMA, qui semble n'être qu'une version améliorée de BAMA avec le même algorithme, avec des listes de 1 328 préfixes, 945 suffixes, 79 318 radicaux (pour 40 654 lemmes), et trois tables de compatibilités morphologiques utilisées pour contrôler les combinaisons préfixe-radical (2 497 entrées), radical-suffixe (1 632 entrées) et préfixe-suffixe (1 180 entrées), incluant des gloses en anglais.

1. Le code source d'Aramorph est disponible sur https://sourceforge.net/projects/aramorph/, et aussi sur https://www.nongnu.org/aramorph/.
2. BAMA 2.0 peut être commandé sur https://catalog.ldc.upenn.edu/LDC2004L02.
3. SAMA v.3.1 peut être commandé sur https://catalog.ldc.upenn.edu/LDC2010L01.

L'idée de Buckwalter est d'intégrer les règles dans le lexique, pour en faire du *data-driven programming*. Il utilise par conséquent un dictionnaire de radicaux. Il fusionne les proclitiques et les préfixes sous le nom de préfixes, d'un côté, et les enclitiques et les suffixes sous le nom de suffixes, de l'autre, dans deux dictionnaires séparés. Il stocke les mots déjà analysés pour éviter d'avoir à refaire les analyses. En sortie, il fournit pour chaque solution le mot entièrement vocalisé, le lemme, le découpage du mot (en indiquant les caractéristiques grammaticales, plus une glose en anglais).

AraMorph et BAMA (SAMA) (alg. 7) utilisent un algorithme de force brute pour décomposer le mot en une succession de préfixes, radical et suffixes possibles (exemple dans le tableau 3.2). Pour chaque décomposition, on vérifie la présence des éléments dans les dictionnaires correspondants (le signe # dans le tableau indique l'absence), puis on vérifie dans les tables de compatibilités si la décomposition est licite.

Tab. 3.2 : Les analyses possibles du mot كتاب

Préfixe	Radical	Suffixe
كتاب	#	#
كتا	ب	#
كتا	#	ب
كت	اب	#
كت	ا	ب
كت	#	اب
ك	تاب	#
ك	تا	ب
ك	ت	اب
ك	#	تاب
#	كتاب	#
#	كتا	ب
#	كت	اب
#	ك	تاب
#	#	كتاب

Algorithm 7 : AraMorph

Input : mot arabe w
Output : radical, affixes, étiquette POS, tiroir, traduction
Data : $AC \leftarrow$ affixes et clitiques (préposés et postposés) ; $R \leftarrow$ dictionnaire des radicaux ; $Mod \leftarrow$ table des compabilités catégorielles

Function `Stem(`w`)` **:**
  if $w \in$ *cache* **then**
    return cache.w ;
  Normalisation
    Remplacer à la fin de w ى par ي ;
    Remplacer à la fin de w ه par ة ;
  Générer la liste des décompositions possibles Sol ;
  forall $V \in$ *décompositions* **do**
    if $\forall x \subset V | x \in$ *dictionnaire* **then**
      $Sol \leftarrow Sol + V$;
    else
      continue ;
  Ne garder dans Sol que les décompositions compatibles ;
return Sol ;

Il faut également mentionner ici Almorgeana, une variante de BAMA, utilisant les mêmes données, mais avec un moteur différent, réalisée par Habash (2007). Cette variante n'existe plus sous forme isolée, elle a été intégrée sous le nom Almor dans le segmenteur Mada (sect. 4.4.1).

3.2.8 Light10 (Al-Stem)

Le segmenteur Al-Stem a initialement été développé en 2002 par Kareem Darwish (K. Darwish, 2002a) ; la liste des affixes a été obtenue par la première étape de Sebawai (sect. 4.2.2), puis corrigée manuellement (K. Darwish, 2003).

Un segmenteur très semblable est développé la même année par Leah Larkey (Larkey et Ballesteros, 2002), repris et modifié par Larkey, Ballesteros et Connell (2007), qui ont développé plusieurs versions sous le nom de Light Arabic Stemmer, abrégé en Light (Light1, Light2, Light3, Light8, et Light10) ; enfin il a été modifié par David Graff et intégré dans Weka (Hall et al., 2009). Saad et Ashour (2010a), qui ont réalisé cette implémentation dans Weka à l'aide de graphes acycliques orientés (DAG), en ont présenté une version *standalone*. Ces versions diffèrent en fonction des proclitiques et suffixes utilisés ; l'algorithme (alg. 8) reste le même ; il est en open-source[1]. Il supprime les proclitiques les plus fréquents ainsi que

1. Le code source est disponible sur https://github.com/motazsaad/arabic-light-stemmer. pour la version *standalone*, et sinon sur https://github.com/MiladAlshomary/light10stemm

les suffixes les plus fréquents tant que la longueur du mot est supérieure à trois caractères.

Ce segmenteur a connu deux améliorations conséquentes, la première, par Elrajubi, en 2013, et la seconde sous le nom de Condlight, en 2018. Ces efforts d'amélioration de Light10, jusqu'en 2018, montrent que celui-ci continue à servir de référence dans la recherche d'informations en arabe (Anwar, Salama et Abdelhalim, 2013 ; Jaafar, Namly et al., 2017).

Algorithm 8 : Light10 Stemmer

Input : texte
Output : la racine de chaque mot
Data : $A \leftarrow$ affixes ; $M \leftarrow$ Mots invariables ;
Function `Stem(`w`)` **:**
  if $w \in M$ **then**
    return w ;
  Normalisation
    Supprimer les diacritiques ;
    Remplacer au début de w {إ أ آ} par ا ;
    Remplacer à la fin de w ى par ي ;
    Remplacer à la fin de w ة par ه ;
  Suppression clitiques
    if *longueur(w)* > 5 **then**
      Enlever préfixes et suffixes de longueur 3;
    if *longueur(w)* > 4 **then**
      Enlever préfixes et suffixes de longueur 2;
    if *longueur(w)* $> 3 \wedge$ *préfixe non enlevé à l'étape antérieure* **then**
      Enlever préfixe de longueur 1;
    if *longueur(w)* > 3 **then**
      Enlever suffixe de longueur 1;
  Supprimer les proclitiques de w ;
  Supprimer les suffixes de w ;
  return *stem*
return `Stem(`w`)` ;

La liste de mots invariables utilisée a été empruntée à Khoja, avec cinquante nouveaux mots ajoutés (annexe A.1).

Le Tableau A.3 montre les affixes et clitiques qui sont supprimés des mots arabes dans chaque version. Les préfixes de l'inaccompli les plus courants ne sont pas enlevés, ce qui fait que le résultat pour les verbes est un mot minimal plutôt qu'un radical.

Larkey, Ballesteros et Connell (2007) ont évalué l'efficacité de Light10 pour la

`er` pour la version intégrée dans Weka.

recherche d'informations à l'aide de données du corpus TREC (sect. 2.2.4). Dans ce contexte, il a surpassé les autres approches.

3.2.9 Light10 → Elrajubi

Elrajubi (2013) propose une amélioration de Light10, avec sa propre liste de mots invariables (annexe A.2), avec trois ajouts importants[1] :

- intégration des préfixes de l'inaccompli, ce qui permet d'obtenir le radical même pour les verbes à l'inaccompli ;
- ajout d'une liste de mots irréguliers, avec des règles spécifiques ; le segmenteur permet de plus aux utilisateurs d'ajouter leurs propres mots irréguliers ;
- ajout de contraintes entre préfixes, proclitiques, suffixes et enclitiques, qui bloquent certaines suppressions (mais ces règles ne sont pas explicitées, contrairement à celles de CondLight).

Le processus de suppression des préfixes et suffixes reste le même que celui de Light10. Dans l'algorithme 9, nous ne reproduisons pas les parties identiques à celui-ci.

1. Pour obtenir le code, qui est propriétaire, il faut écrire à osrjubi@yahoo.com.

Algorithm 9 : Light Stemmer by Osama Mohamed Elrajubi

Input : texte
Output : radical *stem*
Data : $A \leftarrow$ affixes ; $M \leftarrow$ Mots invariables ; $I \leftarrow$ Irréguliers ;
Function `Stem(`w`)` **:**
  if $w \in I$ **then**
    return w.radical ;
  Normalisation
    Cf Light10 sauf pour :
    Remplacer أ، إ، آ، ؤ، ئ par ا ;
  Suppression préfixes
    while *longueur(w)* > 2 **do**
      Enlever ال، و، ف، ب، ك، ل، س ;
  Suppression suffixes
    while *longueur(w)* > 2 **do**
      Enlever ون، ين، ان، ات، ة، ه، ي، ك، نا ;
  Méta-règles
    Si le suffixe supprimé était يه, ajouter ي à la fin ;
    Si le suffixe supprimé était ته, ajouter ه à la fin ;
    Si le mot se termine par ىء, remplacer par ء ;
  if *(longueur(stem)* ≥ 3*)* $\wedge$ $(stem \in R)$ **then**
    return *stem*
  else
    return error

Les méta-règles servent à corriger les erreurs de sur-segmentation causées par la suppression automatique des préfixes et suffixes. Elles restaurent des lettres supprimées à tort, car la suppression de finales comme يه ou ته peut tronquer le radical.

Elrajubi et Light10 ont été comparés sur quatre articles de presse rédigés en arabe tirés du site web d'Al Jazeera (sect. 2.2.9), avec 2 791 mots (le nombre de mots uniques n'est pas indiqué). Les résultats montrent que Light10 atteint une précision de 66%, tandis que le segmenteur d'Elrajubi atteint une précision de 88,25%. Cependant, malgré l'amélioration du taux de précision du système, Elrajubi ne fournit pas la racine correcte pour bon nombre de mots (328 sur 2 791).

3.2.10 Light10 → CondLight

CondLight (Al-Lahham, Matarneh et Hasan, 2018), tout comme l'algorithme d'Elrajubi (que les auteurs ne semblent pas connaître), cherche à corriger les défauts de Light10 en ajoutant des affixes et clitiques supplémentaires et en intégrant des

méta-règles (ici appelées conditions) sur les affixes et clitiques. Les éléments supplémentaires sont ف , ي , س , ل , ت, ب (quatre proclitiques et deux préfixes) et هن , هم, وا (un suffixe et deux enclitiques). En annexe, le tableau A.3 récapitule les évolutions concernant les affixes et clitiques utilisés par les avatars d'Al-Stem.

Concernant l'algorithme[1](voir l'algorithme 10), la principale modification est l'introduction des méta-règles qui bloquent la suppression de clitiques ou d'affixes sous certaines conditions.

- Ne supprimer هن, هم (des pronoms objets ou possessifs) à la fin d un mot que s'il ne commence pas par une séquence de proclitiques incluant l'article ال : ال, فال, كال, وال, بال, لل ; les pronoms objets sont incompatibles avec un nom, et les pronoms possessifs incompatibles avec l'article.
- Ne supprimer les proclitiques ب, ل, ف (prépositions) que si le mot se termine par ها, ان, ات, ين, ه، هن, هم (terminaisons nominales), car les prépositions ne sont compatibles qu'avec des noms.
- Ne supprimer le préfixe ي (préfixe de l'inaccompli) que si le mot se termine par ها, ون, هان, ة, وا (suffixes de l'inaccompli). De même pour le préfixe ت, qui n'est compatible qu'avec les suffixes ت, ها, ون, هان, ين, ة, وا.
- Ne supprimer le proclitique س (préverbe) que si le mot est suivi de ت ou ي (terminaissons verbales), car les préverbes ne sont compatibles qu'avec des verbes.
- Ne supprimer le suffixe (وا (suffixe inaccompli) que si le mot commence par ا, ي ou ت (préfixes de l'inaccompli).

Comme ces métarègles le montrent, cet algorithme réalise une grammaire de la morphologie de l'arabe, qui, comme nous l'avons vu au chapitre 1, inclut des éléments qui, dans d'autres langues, sont des éléments syntaxiques. Cette grammaire est réalisée sous forme de contraintes et non de règles de grammaire standard. Les deux premières conditions concernent la morphologie des noms, les dernières la morphologie des verbes.

CondLight et Light10 ont été comparés sur la collection TREC 2002 (sect. 2.2.4), CondLight a indexé 9% en moins de mots que Light10, pour un nombre similaire de publications, ce qui signifie que CondLight élimine certains termes sans utilité pour la recherche d'information. CondLight améliore d'environ 5% la récupération par rapport à Light10.

1. Pour obtenir le code de CondLight, il faut écrire à yasirlhm@zu.edu.jo.

Algorithm 10 : CondLight, Light10 conditionnel

Input : Mot w
Output : Radical r
Data : $A \leftarrow$ affixes ; $M \leftarrow$ Mots invariables ;
Function `Stem(`w`)` **:**
  Normalisation
    Supprimer les diacritiques ;
    {آ، إ، أ} → ا, ى → ي ;
  Suppression préfixes
    foreach *préfixe* $p \in A$ **do**
      if w *commence par* $p \wedge$ *condition associée* **then**
        Supprimer p de w ;
  Suppression suffixes
    foreach *suffixe* $s \in A$ **do**
      if w *finit par* $s \wedge$ *condition associée* **then**
        Supprimer s de w ;
  while *longueur*$(w) < 3$ **do**
    Annuler la suppression précédente ;
  return w ;

3.2.11 Goweder

Le segmenteur proposé[1] par Goweder et al. (2004) est le premier à avoir pris en compte le problème du pluriel en arabe. Ces pluriels irréguliers, dont on a vu au chapitre 1 qu'ils étaient 41% des pluriels avec 39 schèmes différents, représentent environ 10% des mots dans les corpus ; ils représentent donc un défi particulier pour la segmentation.

Ils ont d'abord élaboré un segmenteur léger inspiré de Khoja (sect. 3.2.3), sans dictionnaire, pour tester spécifiquement les pluriels à l'aide des 39 schèmes de pluriel possibles, puis ont amélioré sa précision (13% sur les pluriels irréguliers) en ajoutant des règles de compatibilité entre clitiques et schèmes. La précision atteignant 20% à peine, ils ont utilisé des statistiques pour construire des règles sur les racines (donc sur les trois consonnes de la racine) et non plus sur les schèmes, augmentant considérablement la précision (à 54% environ). Enfin, un dictionnaire a été construit avec ces règles sur un corpus conséquent (18,5 millions de mots) ; 12 000 pluriels identifiés par le système ont été contrôlés à la main et réduits à 3 600 ; l'utilisation de ce dictionnaire a permis d'atteindre une précision de 81%. Ils ont également testé un arbre de décision C4.5 pour générer les restrictions sur les consonnes, atteignant une meilleure précision que les règles générées à la main,

1. Pour obtenir le code, qui est propriétaire, il faut écrire à agowed@essex.ac.uk.

mais moins bonne que le dictionnaire.

Les résultats de ce segmenteur ont été évalués dans une tâche de recherche d'informations, en le comparant aux trois méthodes d'indexation standard (mot complet, racine et radical), sur une collection de 385 documents (provenant de 7 domaines différents) et un ensemble de 50 requêtes avec leurs jugements de pertinence. Les résultats (fig. 3.1) indiquent clairement que la détection des pluriels irréguliers améliore significativement la segmentation pour l'indexation (p(unilatéral) $< .01$ à la fois par le test de Signe et le test de rang signé de Wilcoxon) (Goweder et al., 2004) .

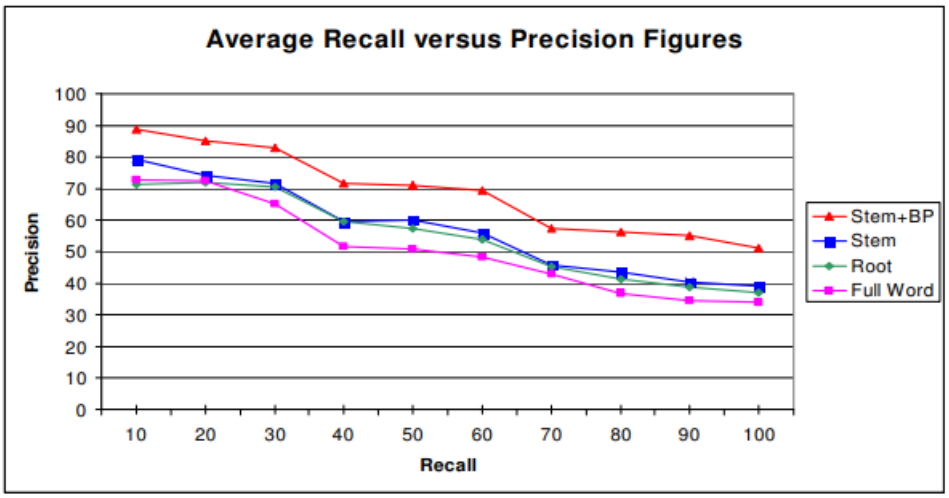


Fig. 3.1 : Rappel et précision moyens (BP = pluriel irrég.) (Goweder et al., 2004)

3.2.12 Kadri & Nie

Kadri et Nie (2006) proposent deux méthodes de segmentation dans le cadre de la recherche d'information, et les ont comparées.

Malheureusement, le code de la méthode la plus performante selon eux dans le cadre de la recherche d'information n'est plus disponible (communication personnelle de Nie), et l'algorithme n'est pas explicité dans leur article. Tout ce qu'on sait de cette méthode, c'est qu'elle enlève les clitiques puis les affixes ; comme il y a de nombreux faux découpages, ils exploitent les statistiques du corpus (Kadri, 2004). Plus précisément, ils génèrent l'ensemble des radicaux possibles pour un mot, et sélectionnent le radical le plus fréquent dans le corpus, qui a auparavant été converti en un dictionnaire de radicaux potentiels, générés à partir des mots du corpus, classés par fréquence. Cette méthode de sélection du radical aurait résolu beaucoup d'ambiguïtés de manière correcte.

Nous en savons davantage sur la deuxième méthode, un segmenteur léger[1] assez semblable aux précédents, dont l'algorithme (alg. 11) enlève les clitiques et affixes (fusionnés) les plus fréquents tant que le radical a plus de trois caractères. Les éléments antéposés les plus fréquents sont principalement des prépositions et l'article, les éléments postposés les plus fréquents expriment le nombre ou le genre.

Algorithm 11 : Light stemming Kadri & Nie

Input : mot arabe w
Output : sa racine
Data : $A \leftarrow$ affixes ; $R \leftarrow$ racines ; $S \leftarrow$ schèmes ; $M \leftarrow$ Mots invariables ; $I \leftarrow$ Mots irréguliers

Function `Stem(`w`)` **:**
  Normalisation
    Remplacer au début de w {إ أ آ} par ا ;
    Remplacer à la fin de w ى par ي ;
    Remplacer à la fin de w ة par ه ;
    Remplacer à la fin de w يء par ى ;
    Supprimer les diacritiques ;
  if $w \in M$ **then**
    return w
  if $w \in I$ **then**
    return w.racine dans I ;
  Suppression clitiques
    if *longueur(w)* ≥ 3 *caractères* **then**
      Supprimer au début de w le clitique "و" (wa) ;
return `Stem(`w`)` ;

Les deux approches ont été comparées sur la collection d'articles TREC (sect. 2.2.4), avec deux ensembles de sujets : TREC 2001 avec 25 sujets, et TREC 2002 avec 50, pour indexer avec le titre et le résumé des articles.

L'indexeur est un modèle de langue unigramme basé sur la divergence de Kullback-Leibler. Lorsqu'une requête et un document sont donnés, le score de pertinence de ce document par rapport à la requête est calculé en utilisant la divergence négative entre le modèle de langue de la requête et le modèle de langue du document (Zhai et J. D. Lafferty, 2001). En d'autres termes, on compare les distributions de probabilité des mots dans la requête et le document. Plus la divergence est faible, plus le document est considéré comme pertinent pour la requête. Le tableau 3.3 montre les différences entre les deux approches. L'évaluation se fait par la précision moyenne (*mean average precision*) sur onze points de rappel. Le rappel est utilisé comme deuxième mesure. La première approche (Linguistic-based stemming) donne systématiquement de meilleurs résultats (31% de précision

1. Pour obtenir le code, il faut écrire à jian-yun.nie@umontreal.ca.

moyenne) que la deuxième (28%). Donc, une segmentation légère de l'arabe n'est pas forcément la meilleure approche pour la recherche d'information ; elle échoue à regrouper des mots sémantiquement similaires dans le même index (plus les index regroupent de mots sémantiquement similaires, meilleures sont les performances).

Tab. 3.3 : Résultats en recherche d'information selon les deux méthodes

Collection	Mesure	Approche 1	Approche 2
TREC 2001	Précision moyenne	0.3326	0.3220
(25 topics)	Recall/4122	2704	2664
TREC 2002	Précision moyenne	0.2828	0.2671
(50 topics)	Recall/5909	4301	4443
TREC 2001-2002	Précision moyenne	0.3107	0.2868
(75 topics)	Recall/10031	7181	7121

Le tableau 3.4 donne quelques exemples pour comparer les deux méthodes.

Tab. 3.4 : Exemples de radicaux trouvés par les deux approches

Mots	Approche 1	Approche 2
عراقيين	عراق	عراقي
البوسنيه	بوسن	بوسني
مهرجان	مهرجان	مهرج

3.2.13 Al-Ameed

Al-Ameed (2006) a proposé dans sa thèse un algorithme qualifié d'extracteur léger de racine et de radical (alg. 12), donnant de meilleurs résultats que Light10 et d'autres algorithmes selon Al-Nashashibi, Neagu et Yaghi (2010b). Ce système vise à améliorer la recherche d'information en arabe en combinant une segmentation légère inspirée des travaux de K. Darwish (2002a) et un filtrage linguistique, sans recourir à une lemmatisation complète.

Ce modèle repose sur une approche light stemmer optimisée, mais enrichie par :

- Une liste étendue de préfixes et suffixes basée sur l'analyse de corpus arabes.
- Des règles de filtrage morphologique pour éviter le sur-découpage.

- Une validation lexicale pour garantir que le radical obtenu est linguistiquement plausible.
- Une intégration dans un moteur de recherche pour tester l'impact sur la précision du rappel.

Les résultats démontrent l'efficacité de son système dans le cadre de la recherche d'information, en le comparant à Light10 et à une recherche sans segmenteur. Il a obtenu sur le corpus de test une précision de 78,6%, un rappel de 72,4% et une F-mesure de 75,4%, surpassant les deux autres configurations. Light10 a atteint une précision de 74,8% et un rappel de 68,2%, et la recherche sans stemming 71,3% de précision et 64,5% de rappel. Ces résultats sont confirmés par Al-Nashashibi, Neagu et Yaghi (2010b) dans leur étude comparative sur l'extraction de radical, ont montré que sur leur corpus, parmi les dix méthodes testées, Al-Ameed a obtenu la meilleure précision (80%).

Algorithm 12 : Al-Ameed

Input : mot w
Output : radical r
Data : A ← affixes ; S *gets* schèmes ; R ← radicaux ;
Function `Stem(`w`)` **:**
  Normalisation
    Remplacer آ إ أ par ا ;
    Remplacer ى par ي ;
    Supprimer les diacritiques ;
  Suppression affixes
    Comme dans l'algorithme 3.2.9 ;
  if *(radical* $\geq m$*)* $\wedge$ *(radical conforme aux schèmes)* $\wedge$ *radical* $\in R$ **then**
    return radical ;
  else
    return w ;

Al-Nashashibi, Neagu et Yaghi (2010a) ont ajouté un algorithme correctif (alg. 13), susceptible de s'appliquer à tous les extracteurs légers, qui ne gèrent pas correctement certaines catégories de mots, qui représentent pourtant environ 30% des mots arabes dans les textes courants. Il s'agit notamment :

- des mots contenant des lettres faibles (voir chapitre 1) ;
- des mots contenant un hamza (ء) ;
- des mots où une voyelle longue a été supprimée par l'algorithme ;
- des mots avec consonnes géminées.

Cet algorithme :

- Identifie les schèmes irréguliers non pris en charge par le stemmer initial.
- Applique des règles spécifiques pour restaurer les racines correctes.
- Utilise une base de 5 405 racines trilittères et quadrilittères comme référence de validation.

Les auteurs ont testé leur approche sur un corpus interne couvrant huit types de textes. Ils ont fait gagner 14% de précision au modèle d'Al-Ameed et 7 à 8% aux autres modèles testés.

Algorithm 13 : Correctif pour segmenteur léger

Input : Mot w
Output : Radical s
Data : A ← affixes de Al-Ameed ; R ← liste des racines

```
Function Stem(w) :
    Normalisation
        {أ، إ، آ} → ا, ى → ي ;
        Supprimer les diacritiques ;
    s ← w ;
    while longueur(s) ≥ 3 caractères do
        s = s - le plus long affixe possible ;
    Extraire r, la liste des consonnes de la racine ;
    if r contient ا، ي، و then
        Appliquer les règles de restauration ;
    if r contient ء then
        Mettre le hamza comme consonne de r dans sa position ;
    if r est géminé then
        Supprimer la gémination ;
    if r a deux consonnes et qu'une voyelle longue manque then
        La rétablir selon le schème ;
    if r ∈ R then
        return r, s ;
    else
        return w ;
```

3.2.14 Morph2

Hadrich Belguith et N. Chaâben (2006) ont proposé Morph2, un segmenteur écrit en Java qui produit une analyse morphologique d'un mot avec ou sans diacritiques. La version 2 par N. K. Chaâben, Hadrich Belguith et Ben Hamadou (2010) contient en plus une étape dans laquelle on génère à partir de l'analyse la forme diacritée, puis on compare avec la forme d'entrée pour valider l'analyse. Il dispose d'un

dictionnaire de racines, d'un dictionnaire de clitiques et d'affixes, et étiquette le texte en sortie à l'aide d'une analyse morphologique complète, en cinq étapes (figure 3.2). Il a été intégré dans le système MASPAR (Aloulou, Hadrich Belguith et Ben Hamadou, 0200)[1].

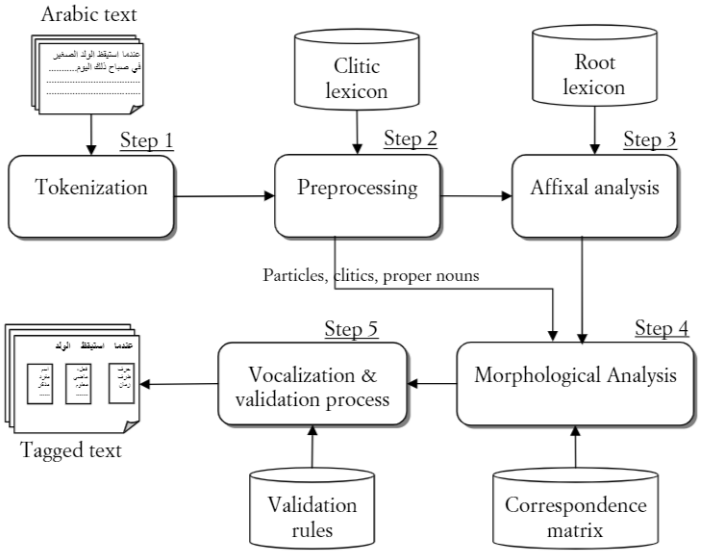


Fig. 3.2 : Morph2 (N. K. Chaâben, Hadrich Belguith et Ben Hamadou, 2010)

L'algorithme (alg. 14) tokenise d'abord avec le système Star de Belguith, Baccour et G. Mourad (2005), puis chaque mot est découpé en mot minimal avec clitiques ; à ce stade, on vérifie si le mot est une particule, un nombre, une date ou un nom propre. Ensuite, le mot minimal est analysé en affixes et radical, selon la méthode proposée par Ben Hamadou (1993), qui consiste à contrôler grâce à un modèle d'association, la compatibilité entre la racine et le schème incluant les affixes. La catégorie morphosyntaxique (POS, genre, nombre, temps, personne, etc.) est déduite à l'étape suivante selon la méthode développée par Belguith Hadrich (1999), déterminant d'abord la partie du discours (POS) puis en déduisant les autres caractéristiques en fonction du schème.

C'est grâce aux schèmes que, dans la version 2, le mot est entièrement vocalisé. La figure 3.3 montre l'analyse du mot non vocalisé تمدّان (ils se développent), décomposé en {مدد; ت - ان}. Or la séquence d'affixes {ت - ان} correspond à un schème qui n'a qu'une vocalisation possible : **taCaCaCān** ; il suffit de remplacer les consonnes du schème par celles de la racine pour obtenir le mot entièrement vocalisé **tamadadān**. Ensuite, on vérifie si des règles particulières ne devraient pas s'appliquer sur l'output (par exemple, une séquence de deux syllabes identiques

1. Pour obtenir le code, il faut écrire à nouha.chaaben@escs.usf.tn.

remplacée par une consonne géminée, comme ici), puis à comparer le résultat, **tamaddān**, avec les diacritiques éventuellement présents dans le mot d'origine. S'il y a une différence entre les diacritiques attestés et produits, la solution est rejetée de la liste des solutions.

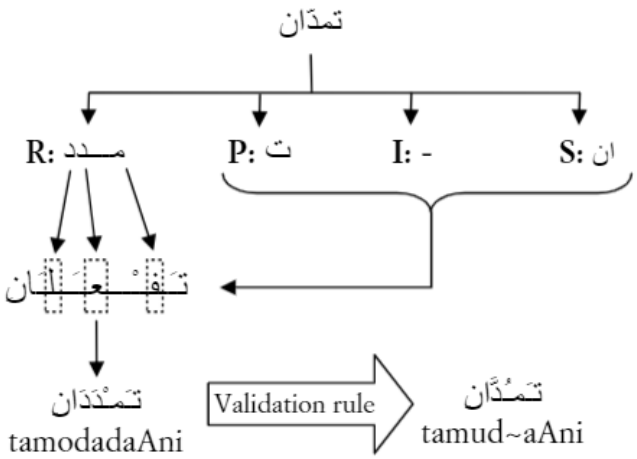


Fig. 3.3 : Vocalisation (N. K. Chaâben, Hadrich Belguith et Ben Hamadou, 2010)

Algorithm 14 : Morph2

Input : texte arabe
Output : pour chaque mot, décompositions racine + schèmes + clitiques avec vocalisation
Data : $A \leftarrow$ affixes ; $C \leftarrow$ clitiques ; $R \leftarrow$ racines ; $S \leftarrow$ schèmes ; $Mod \leftarrow$ compatibilités ;

Function `Stem(`w`)` **:**
  Stocker la forme du mot d'origine F ;
  Découper le mot en mots minimaux + clitiques ;
  Normalisation
    Remplacer à la fin de w ى par ي ;
    Remplacer au début de w {أ آ إ ؤ ىء ء } par ا ;
  Filtrage
    if $w \in$ *particule, nombre, date ou nom propre* **then**
      return case ;
  Détection des schèmes
    $Sol \leftarrow$ toutes les décompositions {préfixe, racine + infixe, suffixe} ;
    foreach $t =$ *{préfixe, infixe, suffixe} et* r *racine* **do**
      if t *compatible avec* r *dans* Mod **then**
        $Sol \leftarrow Sol + t$;
      else
        continue
  Analyse morphologique
    foreach $s \in Sol$ **do**
      Déduire de t et r la partie du discours POS ;
      Extraire de POS ses caractéristiques propres ;
      Conserver les caractéristiques compatibles avec t ;
      Ajouter ces caractéristiques à s ;
  Vérification
    foreach $s \in Sol$ **do**
      À partir du schème générer une vocalisation complète V ;
      Appliquer les règles d'ajustement ;
      if V *incompatible avec* F **then**
        Retirer s de Sol ;
  return Sol

3.2.15 ElixirFM

ElixirFM[1] (Smrž, 2007a,b), écrit en Haskell (Hudak, 2000), un langage fonctionnel déclaratif et en Perl, est un analyseur morphologique à base de règles utilisant le

1. Le code source peut être obtenu sur https://github.com/otakar-smrz/elixir-fm.

lexique de BAMA (voir l'algorithme 15), enrichi par des informations provenant des annotations syntaxiques du Prague Arabic Dependency Treebank (sect. 2.5.3). Sa particularité est d'une part sa facilité d'utilisation, car il est essentiellement déclaratif et les données sont complètement indépendantes du code, contrairement à la plupart des segmenteurs, et d'autre part le fait qu'il implémente des concepts de morphosyntaxe fonctionnelle, destinés à faciliter l'interfaçage de la morphologie avec la syntaxe.

L'analyseur construit des *tries* (arbres préfixes) pour produire toutes les combinaisons des formes de BAMA à comparer à la forme attestée en fonction de la grammaire déclarative. Il parcourt donc toutes les possibilités. Cette grammaire contient toutes les compatibilités, et les analyses incohérentes sont donc abandonnées.

Algorithm 15 : ElixirFM : Analyse morphologique fonctionnelle

Input : Mot w
Output : lemme, catégorie, fonction syntaxique
Function `ElixirFMAnalyze`(w) **:**
  Normalisation
    Supprimer les diacritiques ;
    Remplacer {آ، إ، أ} par ا, ى par ي ;
  List ← lemmes de BAMA compatibles avec w ;
  foreach $l \in$ *List* **do**
    Récupérer le paradigme correspondant dans la grammaire ;
    Générer les formes possibles selon ce paradigme ;
    Comparer avec w ;
  Catégorisation
    Déterminer la catégorie grammaticale ;
    Extraire les traits morphosyntaxiques (genre, nombre, temps, etc.) ;
  Associer les fonctions syntaxiques selon les dépendances ;
  Utiliser les annotations du Prague Arabic Dependency Treebank ;
  return lemme, catégorie, fonction syntaxique ;

3.2.16 Bigrammes

I. I. Hmeidi et al. (2009) ont proposé d'utiliser les bigrammes pour l'extraction de racines. Cette méthode vise à améliorer la performance du processus d extraction et à accroître la couverture de ce processus, et fonctionne bien dans le cas de textes contenant des erreurs typographiques (Khreisat, 2006). Un bigramme est une sous-séquence de 2 lettres d un mot. Par exemple, la liste des bigrammes du mot فسيكفيكهم |fsykfīkhm| est {فس, سي, كف, في, يك, كه, هم}. Une base de ∼ 9 000 racines a été extraite du dictionnaire de référence *Lisan al'arab*, et ces racines ont

été découpées en bigrammes.

L'article est mal écrit et peu rigoureux, beaucoup de points ne sont pas clairs, le code n'est pas disponible et ne semble pas être open-source. Le principe général de l'algorithme est de construire quatre racines candidates à partir du mot :

- en enlevant les suffixes de longueur 1;
- en enlevant les préfixes de longueur 1;
- en enlevant tous les suffixes ;
- en enlevant tous les préfixes.

Puis on compare chacune de ces racines candidates à la base de racines ; pour chaque racine candidate R_i, on récupère toutes les racines de la base ayant au moins un bigramme en commun (s'il n'y en a aucune on renvoie le mot d'origine), et on sélectionne la racine la plus semblable selon deux mesures.

La première mesure est la similitude Dice présentée dans l'équation 3.1 :

$$D(R_i, W_j) = \frac{2|R_i \cap W_j|}{|R_i| + |W_j|} \tag{3.1}$$

où $|R_i \cap W_i|$ est le nombre de bigrammes identiques entre le mot et la racine, |R_i| est le nombre de bigrammes de la racine et |W_j| est le nombre de bigrammes du mot. La racine ayant la plus grande mesure de Dice est choisie.

L'équation 3.2 donne la deuxième mesure basée sur la distance de Manhattan :

$$D(R_i, W_j) = \sum_{h=1}^{k} |(R_{ih} - W_{jh})| \tag{3.2}$$

où W_j est la racine dans la base, h est l'indice qui court sur les bigrammes et k le nombre total de bigrammes. On ne sait pas ce qui se passe quand il y a plus de bigrammes dans une des formes. La racine ayant la plus faible distance est choisie.

Un post-traitement est ensuite appliqué, au cas où plus d'une racine serait choisie ; si la première lettre de la racine candidate est un préfixe possible, on l'enlève puis on refait les calculs ; sinon on renvoie la racine qui a le même premier bigramme (quid s'il n'y en a pas ?).

L'algorithme fonctionne en cinq étapes (l'article est trop peu précis pour que nous puissions mettre en forme du pseudo-code) :

1. Extraction d'un premier candidat racine avec les procédures standard (normalisation, suppression des suffixes, des préfixes, et des infixes les plus courants. À la sortie de ce processus, le candidat doit avoir au moins trois lettres.

2. Modification de ce candidat racine ; les modifications sont adhoc, nombreuses, peu claires et non justifiées. Par exemple, on doit supprimer les lettres ط ,د quand elles suivent respectivement {ض ,ظ ,ط ,ص} et {ز ,ذ ,د} ; ou supprimer le *chaddah* sur la lettre en deuxième position, alors qu'il a déjà été éliminé lors de la normalisation ; ou encore, si le candidat racine contient des suffixes, enlever la première lettre – alors que les suffixes ont déjà été enlevés à l'étape précédente.

3. Produire les quatre candidats racine.

4. Une fois sélectionnées les racines de la base à comparer avec les racines candidates, appliquer un nouvel ensemble de modifications sur celles-ci. Ces modifications ont essentiellement pour but de gérer les cas de racine faible (racine avec une lettre faible, cf chapitre 1)

   - le candidat se termine par ا, le remplacer par waw و, et si ça ne marche pas, par ي ;
   - il se termine par و), le remplacer par ا, et sinon par ي ;
   - il se termine par ي), le remplacer par ا et sinon par و ;
   - il se termine par ى), le remplacer ا, sinon par و, et sinon par ي.
   - il contient ا, le remplacer par و et sinon par ي ;
   - il contient ي, le remplacer par و et sino par ا ;
   - il contient waw (و, le remplacer par ا et sinon par ي.

5. Appliquer la mesure choisie et sélectionner la racine.

L'algorithme a été testé sur 50 799 mots extraits du Coran (50 484 à racine trilitère, 298 quadrilitère, 12 pentalitère, 5 hexalitère) et sur 26 994 mots extraits du corpus SACS (sect. 2.2.1, 25 871 trilitère, 167 quadrilitère, 944 transcription mots étrangers). La précision à partir de Dice est présentée dans le tableau 3.5, et l'exactitude (*accuracy*) pour les deux mesures est présentée dans les tableaux 3.7 et 3.6.

Tab. 3.5 : Précision

Corpus	**trilitères**	**quadrilitères**	**pentalitères**	**hexalitères**
Dice				
Coran	85,56%	96,980%	100%	100%
Manhattan				
Coran	80,68%	91,61%	100%	100%
SACS	80,422%	97,006%	-	-

Tab. 3.6 : Accuracy sur Dice

Corpus	**trilitères**	**quadrilitères**	**pentalitères**	**hexalitères**
Sans mots étrangers				
Coran	89,659%	96,980%	100%	100%
SACS	88,851%	100%	-	-
Avec mots étrangers				
Coran	85,560%	96,980%	100%	100%
SACS	80,685%	100%	-	-

Tab. 3.7 : Accuracy sur Manhattan

Corpus	**trilitères**	**quadrilitères**	**pentalitères**	**hexalitères**
Sans mots étrangers				
Coran	88,433%	91,611%	100%	100%
SACS	88,384%	97,006%	-	-
Avec mots étrangers				
Coran	80,680%	91,611%	100%	100%
SACS	80,422%	97,006%	-	-

3.2.17 Alkhalil

Alkhalil[1], dont une première version a été développée à l'Université de Pennsylvanie (Boudlal et al., 2010), a ensuite eu une version 2 largement améliorée par l'équipe de Traitement du Langage Naturel d'Oujda (Boudchiche et al., 2017) de l'Université Mohammed V au Maroc, avec le soutien de l'Organisation Éducative, Culturelle et Scientifique de la Ligue Arabe (ALECSO), avec un pourcentage de mots analysés dépassant 99% et une rapidité fortement augmentée. Il a été intégré dans le logiciel libre Tashkeela[2] (Soudi, E. Atwell et Hedlund, 2012). Il renvoie la racine, le radical et, depuis la version 2, le lemme et son schème.

L'algorithme (alg. 16) élimine les mots invariables, et normalise les mots restants en supprimant les diacritiques. Mais le point original par rapport à la plupart des autres méthodes, c'est que, comme pour Morph2, ces diacritiques sont stockés en mémoire afin de rejeter les résultats d'analyse incompatibles avec eux. Ensuite, le mot est considéré comme une série de constituants (proclitiques + radical + enclitique) ; le système propose toutes les segmentations concevables en parcourant les listes de proclitiques et d'enclitiques. Il ne conserve que les segmentations compatibles avec les proclitiques et les enclitiques associés.

Passons à l'analyse du radical. En l'absence de signes diacritiques, le même radical peut donner lieu à plusieurs interprétations. Par conséquent, pour chaque segmentation validée à l'étape précédente, le système commence par vérifier si la racine appartient au dictionnaire des mots irréguliers ; si oui, la racine du mot irrégulier (indiquée dans le dictionnaire) est renvoyée et l'analyse s'arrête. Sinon, trois processus sont lancés en parallèle, pour tester trois hypothèses sur le mot :

- Nom primaire (non dérivé) : on vérifie si le radical appartient à la liste des noms primaires et si ce radical est compatible avec les clitiques observés.
- Nom dérivé : le système commence par vérifier si les clitiques découverts sont compatibles avec les noms. Puis il identifie les radicaux et les schèmes possibles.
- Verbe ; le système vérifie si les clitiques découverts sont compatibles avec les verbes. Puis il identifie les radicaux et les schèmes possibles.

.

Enfin, pour intégrer facilement le programme dans d'autres applications, une API est disponible[3].

1. Code source sur https://sourceforge.net/projects/alkhalil/.
2. https://github.com/Anwarvic/Arabic-Tashkeela-Model.
3. http://oujda-nlp-team.net/?p=1299%26lang=en.

Algorithm 16 : Alkhalil

Input : texte arabe
Output : pour chaque mot, racine, radical, + lemme et schème (v2)
Data : $C \leftarrow$ clitiques ; $R \leftarrow$ racines ; $S \leftarrow$ schèmes $\supset$ affixes ; $I \leftarrow$ irréguliers ; $M \leftarrow$ Mots invariables

```
Function Stem(w) :
    if w ∈ M then
        return w ;
    Orig ← w avant normalisation ;
    Normalisation
        Supprimer les diacritiques ;
    Supprimer les clitiques de w ;
    Générer des hypothèses pour w en utilisant les trois phases ;
    foreach hypothèse do
        Générer une version de w avec diacritiques ;
        if ¬ diacritiques(w) ⊃ Orig then
            supprimer hypothèse ;
    return hypothèses restantes ;
```

3.2.18 SARF

SARF (Alkhalifa, Al-Saydi et Al-Khalifa, 2014 ; Makhlouta et Harkous, 2010) est un analyseur morphologique développé entre 2010 et 2014 à l'Université Américaine de Beyrouth en collaboration avec le laboratoire *Microsoft Advanced Technology Lab* du Caire, dans le cadre d'un service Microsoft en ligne appelé *Arabic Toolkit Service* pour traiter automatiquement des textes arabes. Il n'a fait l'objet d'aucune publication officielle. Les auteurs ont publié sur Sourceforge la partie génération de l'algorithme[1], mais le code source de l'analyseur proprement dit n'a jamais été public. Le service proposé par Microsoft a fermé en 2018.

En sortie, SARF fournit toutes les analyses possibles (ex. fig. 3.4), chacune avec affixes, radical décomposé en racine et schème, catégorie morphosyntaxique (POS), genre, nombre, personne. Ces analyses sont classées par ordre de fréquence. Si le mot n'est pas valide, SARF le signale. Comme avec Morph2 ou AlKhalil (sections 3.2.14 et 3.2.17), les diacritiques de la forme d'entrée sont conservés pour validation.

La figure 3.6 montre les composants de Sarf. Le dictionnaire des radicaux est une extension de celui d'AraMorph (sect. 3.2.7), ajoutant des noms propres et des noms de lieux extraits de sources en ligne. Il y a des règles pour extraire les affixes et clitiques et des règles pour tester la compatibilité entre radical et affixes. Sarf

1. https://sourceforge.net/projects/sarf/.

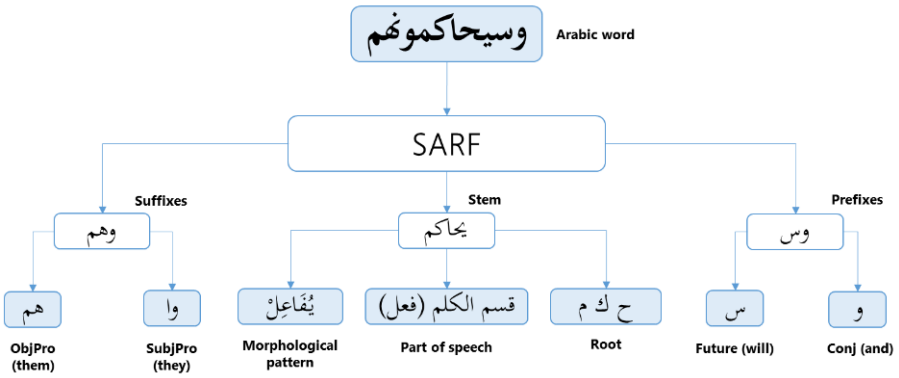


Fig. 3.4 : Exemple de segmentation fait par Sarf

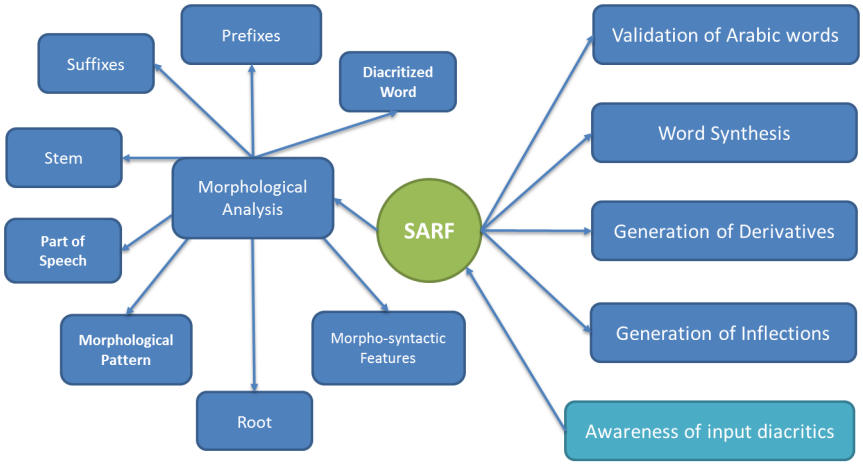


Fig. 3.5 : Fonctionnalités de Sarf

construit des graphes acycliques orientés (DAG) pour clitiques et affixes (confondus mais distingués en antéposés / postposés), et un arbre préfixe (*trie*) sur les radicaux indexés par racine (Aoe, 1989).

L'algorithme (alg. 17) produit une séquence de morphèmes, avec ses caractéristiques morphémiques, grâce aux graphes d'affixes et de clitiques (comme dans l'implémentation de Light10 pour Weka ou comme Tashaphyne), vérifie la compatibilité de la concaténation et construit des arbres de solutions. L'API OnMatch est appelée pour chaque solution et prend en compte les règles de priorité et de sélection des caractéristiques définies par l'utilisateur. Une règle de sélection définit les caractéristiques morphologiques à inclure dans l'arbre de solutions. Une règle de priorité est un ordre sur les caractéristiques à suivre lors de la construction de l'arbre, de manière à ce que les caractéristiques ayant une priorité plus éle-

vée apparaissent à des niveaux supérieurs. L'utilisateur peut rejeter une solution manuellement.

Algorithm 17 : Sarf Stemmer

Input : texte arabe
Output : racine, catégorie, clitiques et affixes, schème
Data : $A \leftarrow$ affixes (préfixes, suffixes) ;
Function `Stem`(w) **:**
  Normalisation
    Supprimer les diacritiques ;
  Décompositions
    Générer un graphe (DAG) à partir des préfixes ;
    Générer un graphe (DAG) à partir des suffixes ;
    Générer toutes les décompositions de w à l'aide des deux graphes ;
  Produire une séquence de morphèmes pour w ;
  Construire les structures de graphe acyclique ;
  while *préfixe et suffixe incompatibles* **do**
    Choisir la décomposition selon OnMatch API ;
    return w sans les affixes ;
  Extraire la racine du radical en enlevant le plus d'infixes possible ;
return radical et racine ;

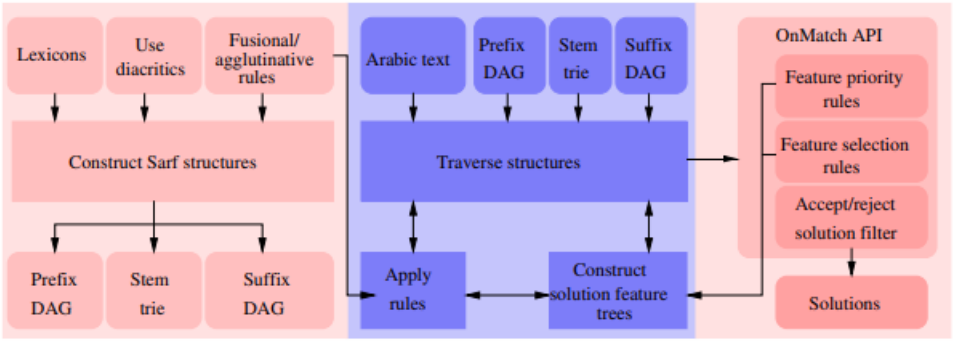


Fig. 3.6 : Diagramme de Sarf (Alkhalifa, Al-Saydi et Al-Khalifa, 2014)

3.2.19 Tashaphyne

Tashaphyne est un segmenteur léger[1] écrit en Python en 2010 par T. Zerrouki, et qui a connu ensuite trois versions avant la publication officielle de 2023 (Al-Khatib, Zerrouki, Abu Shquier et al., 2023). Il permet aux utilisateurs de personnaliser les

1. Le code source de Tashaphyne est disponible sur https://github.com/linuxscout/tashaphyne ou sur le site original https://pypi.org/project/Tashaphyne/.

options de segmentation et les données, en particulier pour modifier la liste des affixes et clitiques confondus et séparés en antéposés / postposés.

L'algorithme (alg. 18) utilise des graphes acycliques orientés (DAG), plus précisément des arbres imbriqués inspirés par Alur, Chaudhuri et Madhusudan (2006). Il y a un DAG pour les préfixes et un autre pour les suffixes. Ces graphes sont générés à partir du mot et du dictionnaire d'affixes concerné. Ils contiennent dans une représentation compacte toutes les décompositions possibles du mot. Les deux graphes analysent le mot en parallèle pour proposer des radicaux potentiels. Parmi ces différentes possibilités, on choisit le radical le plus court possible (avec l'affixe le plus long possible donc), après avoir éliminé les affixes incompatibles entre eux. Enfin, une procédure spécifique retire les infixes, pour fournir la racine.

Par exemple, supposons que la liste de « préfixes » soit {و, وال, والم, وك, وكال, وكالم, وب, وبال, وبالم}, et les mots d'entrée soient {والمكتب, وبالمكتب, والمكتب}. On construit l'arbre imbriqué de préfixes (figure 3.8), qui enregistre la position de chaque dernière lettre d'un préfixe potentiel (cercle double dans la figure). Les positions sont ensuite extraites de droite à gauche, par exemple {1, 2, 4, 5} pour le mot d'entrée وبالمكتب. Les préfixes possibles sont donc {و, وب, وبال, وبالم}. On choisit par défaut le préfixe le plus long وبالم, ce qui donne كتب comme partie droite (radical + suffixes). Pour extraire le « suffixe », on procède de même, à ceci près qu'on ira de gauche à droite.

On teste ensuite si le préfixe et le suffixe sont compatibles (s'ils appartiennent à des catégories compatibles, en excluant par exemple un préfixe nominal avec un suffixe verbal. Sinon, on élimine cette décomposition et on en prend une autre, fournissant le radical le plus court possible, jusqu'à ce qu'on trouve une décomposition où suffixe et préfixe sont compatibles. Par exemple, après filtrage des incompatibilités, la procédure génère 20 hypothèses pour le mot فسيكتبونهما dans la figure 3.7 ; celle qui a le radical le plus court (كتب, avec le préfixe فسي et le suffixe ونهما) est en position 16.



Fig. 3.7 : Segmentation de فسيكتبونهما (Al-Khatib, Zerrouki, Shquier et al., 2021)

Algorithm 18 : Tashaphyne Stemmer

Input : texte arabe
Output : radical, racine
Data : $A \leftarrow$ affixes (préfixes, suffixes) ;
Function `Stem`(w) **:**
  Normalisation
    Supprimer les diacritiques ;
  Radical
    Générer un graphe (DAG) à partir des préfixes ;
    Générer un graphe (DAG) à partir des suffixes ;
    Générer toutes les décompositions de w à l'aide des deux graphes ;
    Supprimer les décompositions où préfixe et suffixe sont incompatibles
      Choisir la décomposition avec le préfixe et le suffixe les plus longs ;
    return radical = w sans les affixes ;
  racine $\leftarrow$ radical - infixes ;
return radical, racine ;

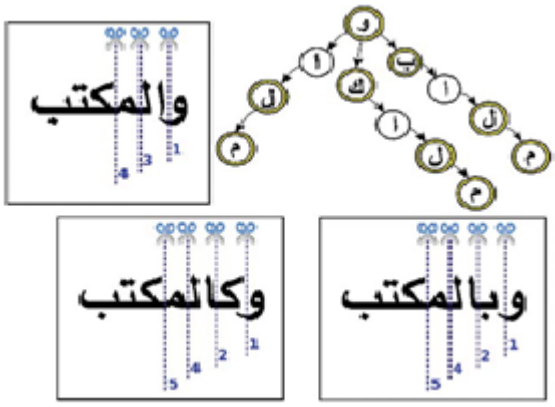


Fig. 3.8 : Extraction de préfixes par arbre imbriqué (Al-Khatib, Zerrouki, Shquier et al., 2021)

3.2.20 Zaraket

Zaraket et Makhlouta (2012) construisent un analyseur morphologique avec des lexiques d'affixes et des règles de concaténation d'affixes et de radicaux inspirées des règles des manuels de morphologie ainsi que des règles de concaténation des analyseurs existants (alg. 19). Ces règles indiquent si deux affixes sont compatibles entre eux (R_{pp} et R_{xx}) ou si le radical est compatible avec tel affixe (R_{ps} et R_{sx}). Chaque élément (affixe, radical) possède une étiquette, et les règles contiennent une expression régulière qui donne l'étiquette résultante à partir des étiquettes des éléments (Spencer, 1991).

Leur méthode considère trois types d'affixes :

- Affixes atomiques tels que ي |ī|, qui peuvent se connecter directement aux racines selon R_{ps} ou R_{sx}.
- Affixes partiels tels que س |sa|, qui doivent se connecter à d'autres affixes avant de se connecter à un radical.
- Affixes composés, concaténations de morphèmes atomiques et partiels et d'autres affixes composés, qui peuvent se connecter aux racines selon R_{ps} ou R_{sx}.

Les règles R_{pp} gèrent la concaténation entre préfixes et les règles R_{xx} la concaténation entre suffixes. Une règle de concaténation $r \in R_{pp} \cup R_{xx}$ prend les étiquettes de catégorie des affixes a_1 et a_2 et vérifie s'il y a un motif dans lesquels elles figurent ensemble. Si c'est le cas, la règle prend les étiquettes de a_1 et a_2 et génère

l'affixe $a = r(a_1, a_2)$ avec l'étiquette déduite ; le nouvel affixe n'est pas forcément une simple concaténation, il peut y avoir une modification.

Les données ont été consituées de la manière suivante :

- Les motifs proviennent des manuels de morphologie.
- Les affixes atomiques et partiels proviennent des lexiques BAMA et SAMA (sect. 3.2.7).
- Les affixes composés présents dans ces lexiques sont générés par règles.
- Les règles partageant les mêmes motifs sont regroupées.

L'analyseur a atteint un taux de précision de 95% lors de la génération et de la comparaison des affixes avec les lexiques BAMA et SAMA, et les règles ont permis de réduire les incohérences dans les lexiques de 20%, la redondance de 15%, et la couverture lexicale a été augmentée de 10%.

Algorithm 19 : Zaraket analyzer

Input : mot w
Output : radical, affixes, étiquettes
Function `MorphAnalyze(`w`)` **:**
  Normalisation
    Supprimer les diacritiques ;
    Remplacer {آ، إ، أ} par ا, ى par ي ;
  Combinaison
    Identifier les affixes ;
    Appliquer les règles de compatibilité ;
    Générer toutes les combinaisons valides $(a_1, a_2, ..., a_n)$;
  Concaténation
    Appliquer les règles de concaténation R_{ps} et R_{sx} ;
    Valider les formes générées selon les contraintes morphosyntaxiques ;
  Validation
    Vérifier que le radical existe dans le lexique ;
    Vérifier que sa catégorie est compatible avec celle des affixes ;
  return radical, affixes, étiquettes ;

3.2.21 Assem

Chelli (2018) a proposé un segmenteur léger[1] écrit en Snowball et inspiré par l'algorithme de Porter (M. Porter, 1980) développé pour l'anglais.

L'algorithme d'Assem (alg. 20) commence par déterminer la catégorie du mot (nom ou verbe) à partir des proclitiques, suffixes et enclitiques :

1. On peut le télécharger dans plusieurs langages de programmation sur https://arabicstemmer.com/.

- Liste des proclitiques : لل ، و ، ف ، لاو ، لاف ، إأ ، اأ ، ؤأ ، آأ ، أأ ، أ , اك ، لاب ، لا تسن ، تسى ، أس
- Liste des suffixes et enclitiques : نس ، تس ، يس ، كك ، بب ، ل ، ك ، ب ، ل ، امك ، مه ، نه ، اه ، مك ، ان ،ه ، ك ، ي تست ، مك ، نك ، ان ، ين ، ة ، نيت ، نات ، ت ، تا ، و ، ا ، ن ، امه ، نا ، نت ، ات ، و

Puis il segmente le mot en fonction des règles spécifiques à sa catégorie.

Algorithm 20 : Assem Stemmer

Input : mot w
Output : la racine de w
Data : $A \leftarrow$ affixes ; $M \leftarrow$ Mots invariables ;
Function `Stem(`w`)` **:**
  Récupérer la catégorie de w (nom, verbe) ;
  if $w \in M$ **then**
    return w ;
  Normalisation
    Supprimer les diacritiques de vocalisation ;
    Remplacer au début de w {أ ٱ إ} par ا ;
    Remplacer à la fin de w ى par ي ;
    Remplacer à la fin de w ة par ه ;
  Suffixes
    if w *verbe* **then**
      if *longueur(w)* ≥ 4 *caractères* **then**
        Supprimer les suffixes verbaux de w ;
    if w *is nom* **then**
      if *longueur(w)* ≥ 3 *caractères* **then**
        Supprimer les suffixes nominaux de w ;
  Supprimer les proclitiques généraux ;
  Préfixes
    if w *verbe* **then**
      if *longueur(w)* ≥ 4 *caractères* **then**
        Supprimer les préfixes verbaux de w ;
    if w *nom* **then**
      if *longueur(w)* ≥ 3 *caractères* **then**
        Supprimer les suffixes nominaux de w ;
  return w après suppressions ;

3.2.22 ARAMO

El Abdi, Ben Ali et Ben Yahia (2022) ont proposé un analyseur d'unités syntagmatiques appuyé, au lieu d'un dictionnaire, sur une ontologie suivant le modèle

WSMO (Keller et al., 2004), réalisée à l'aide de la plateforme open-source de description d'ontologies de l'Université de Stanford, Protégé 5.2.0. Cette ontologie est une hiérarchie de 209 classes, et dispose d'une interface en langage naturel selon Hakkoum, Kharrazi et Raghay (2018). La figure 3.9 montre les classes des deux premiers niveaux de cette ontologie. Comme on le voit, l'analyseur est orienté sur les syntagmes verbaux ou prépositionnels, et ne traite pas les noms. L'idée de se concentrer sur ce type de structure vient du fait que c'est la plus complexe, puisque souvent équivalent à une phrase entière.

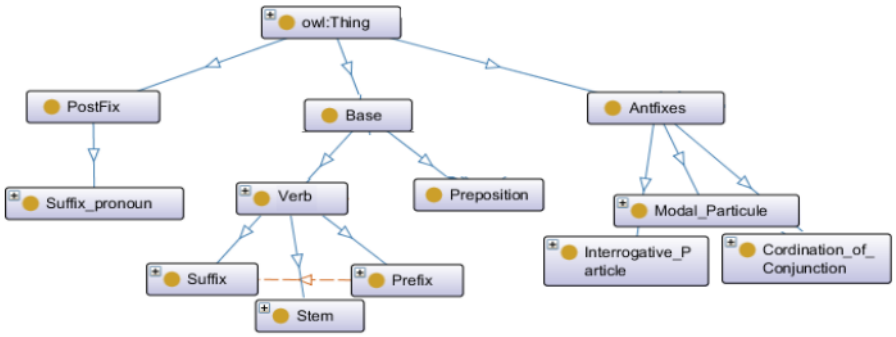


Fig. 3.9 : Classes supérieures d'Aramo (El Abdi, Ben Ali et Ben Yahia, 2022)

Postfix et *Antfix* correspondent aux clitiques (avec les sous-classes associées à chaque clitique) sauf pour les prépositions, traitées à un deuxième niveau. Par rapport à notre analyse de départ (chapitre 1), ils distinguent donc un niveau de plus. La figure 3.10 détaille les caractéristiques du verbe, qui sont celles qui influencent le schème et les affixes verbaux ; par exemple, distinguant, pour le temps, l'accompli, l'impératif et l'inaccompli et ses différentes sous-catégories (présent, futur...). L'ontologie contient 140 instances, avec 72 caractéristiques de ce type. Les syntagmes prépositionnels sont au nombre de 152 (puisqu'ils ne prennent pas en compte les noms, mais seulement les pronoms).

Son architecture est présentée figure 3.11. L'algorithme n'est pas très détaillé, et de surcroît passe par dessus la principale difficulté, distinguer un nom d'un verbe.

L'évaluation de ce système a été fait sur 55 000 syntagmes verbaux et prépositionnels extraits manuellement du corpus Tashkeela (sect. 2.2.14), à l'aide du raisonneur HermiT inclus dans la plateforme Protégé. Il a été comparé à deux autres systèmes, celui de Smine et Mankai (2006) et Madamira (sect. 4.4.1) ; la mesure est faite sur le nombre de cas où chaque système a déterminé correctement la totalité des caractéristiques morphosyntaxiques d'un syntagme. Le tableau 3.8 montre des résultats qui pour eux sont encourageants.

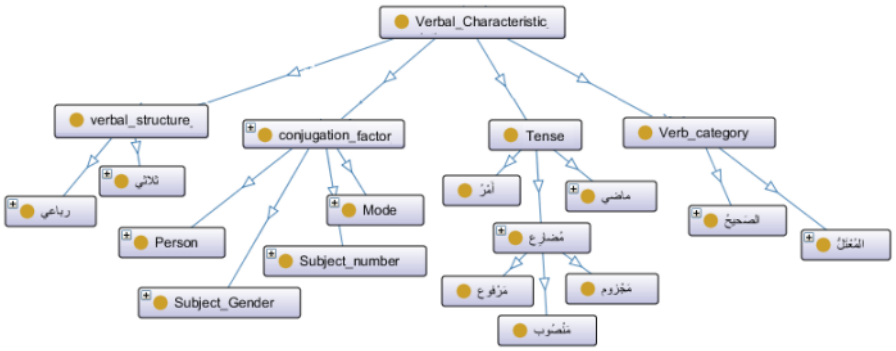


Fig. 3.10 : Classes liées au verbe dans Aramo (El Abdi, Ben Ali et Ben Yahia, 2022)

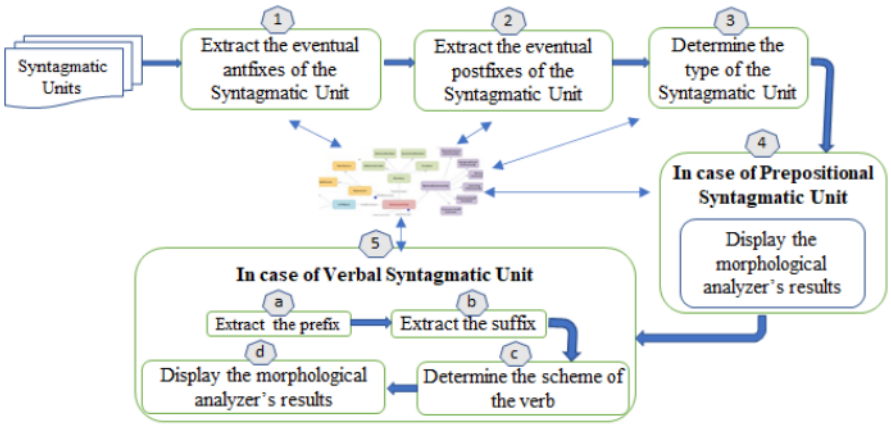


Fig. 3.11 : Architecture d'Aramo (El Abdi, Ben Ali et Ben Yahia, 2022)

On peut quand même estimer que le choix des données d'entrée ne pouvait que favoriser un système qui ne fonctionne que sur ce type de données, alors que Madamira, par exemple, accepte n'importe quel type de données en entrée. De surcroît, le fait de sortir ces données de leur contexte ne pouvait que défavoriser un segmenteur contextuel comme Madamira.

Tab. 3.8 : Évaluation d'Aramo (El Abdi, Ben Ali et Ben Yahia, 2022)

	Smine	**Madamira**	**Aramo**
Précision	70,56%	83,32%	88,22%
Rappel	62,12%	75,15%	78,14%
F-mesure	65,33%	78,45%	82,15%
Performance	26	383	360

3.3 Conclusion

Nous avons pu voir que les segmenteurs à base de règles avaient de nombreux points communs, mais aussi des différences importantes. Il est d'usage d'opposer segmenteurs légers - sans dictionnaire de racines ou de radicaux - et les autres ; mais en réalité, il y a beaucoup plus de nuances. Le type de règles utilisées est aussi un critère ; il y a des segmenteurs qui appliquent des règles de découpage à gauche et à droite de manière indépendante, et d'autres qui utilisent des métarègles qui conditionnent l'application des règles de découpage. Il y a des segmenteurs dits légers qui utilisent des dictionnaires de radicaux, même s'ils sont plus petits (mots irréguliers ou racines faibles par exemple). Dans les segmenteurs légers, c'est sans doute Light10 qui a donné lieu à la plus grande variété d'implémentations, ce qui est un indice de son succès.

Du côté des analyseurs morphologiques, BAMA (SAMA) se détache nettement car il a fait l'objet de nombreuses utilisations. L'analyseur à base d'ontologie aurait été intéressant s'il n'était pas si limité et son évaluation si peu convaincante.

On notera par ailleurs l'importance qu'a eue la recherche d'information, ici plus précisément l'indexation documentaire, comme critère d'évaluation. Mais l'indexation documentaire est très loin d'être le seul domaine où les segmenteurs sont utiles. En particulier, en ce qui nous concerne, ce sont les conséquences sur la sémantique du vocabulaire et l'analyse syntaxique qui intéressent notre équipe de recherche.

Indépendamment du modèle utilisé, il existe un certain nombre de post-traitements ou de pré-traitements qui améliorent les résultats de ces segmenteurs non contextuels. Ainsi H. Al-Shalabi et al. (2022) indique que c'est le cas pour la détection des mots étrangers arabisés.

Ces segmenteurs sont la base d'une partie des segmenteurs que nous allons voir au chapitre suivant.

Chapitre 4

La segmentation complexe

Sommaire

4.1 Introduction 118
4.2 Segmenteurs non contextuels probabilistes 118
4.2.1 Chen & Gey 118
4.2.2 Sebawai 120
4.3 Segmenteurs contextuels par sélection 122
4.3.1 BAMAE, BASMA 123
4.3.2 SAFAR 125
4.4 Segmenteurs contextuels probabilistes 128
4.4.1 MADA, MADAMIRA 129
4.4.2 Farasa 130
4.4.3 Stanford NLP 132
4.5 Segmenteurs contextuels neuronaux 133
4.6 Modèles d'apprentissage séquentiel 135
4.6.1 LSTM 135
4.6.2 GRU 137
4.7 Conclusion 138

4.1 Introduction

Nous allons maintenant examiner les systèmes plus complexes ; parmi ceux-ci, nous distinguerons ceux qui combinent un algorithme non contextuel avec une couche de sélection à base probabiliste ou à base d'apprentissage automatique, et les algorithmes complètement à base d'apprentissage automatique. Enfin, nous décrirons les modèles à d'apprentissage séquentiel sur lesquels certains segmenteurs sont basés.

4.2 Segmenteurs non contextuels probabilistes

Les segmenteurs probabilistes sont des segmenteurs qui apprennent sur la base d'un corpus à générer automatiquement les règles qui seront ensuite utilisées comme dans un segmenteur à base de règles. Nous avons répertorié deux segmenteurs de ce type dans le tableau 4.1.

Tab. 4.1 : Segmenteurs probabilistes

Segmenteur	Input	Output	Date	Probas	Langage
Chen & Gey			2002	WordRank	
Sebawai			2002	Markov	

4.2.1 Chen & Gey

A. Chen et Gey (2002) proposent un critère de découpage original, appelé WordRank. L'intuition derrière cette méthode est que les limites entre les morphèmes adjacents indiquent la justesse les unes des autres. Autrement dit, si une hypothèse de morphème a une limite correcte (ou incorrecte), son voisin aurait simultanément une limite correcte (ou incorrecte) du côté correspondant. Cela indique également que la qualité des limites d'un morphème dépend des limites de ses voisins, ce qui est similaire à PageRank (Brin et Page, 1998), où l'importance d'une page Web dépend de l'importance des pages qui lui sont liées[1].

Ils utilisent un algorithme d'analyse de liens, similaire à l'algorithme HITS initialement proposé pour classer les pages Web (Kleinberg, 1999), pour calculer la Valeur de Bordure Extérieure (EBV), qui estime la qualité des limites de morphèmes pour chaque hypothèse, à partir des Valeurs de Bordure de gauche (LBV) et de droite (RBV). À l'étape initiale, les LBV et RBV de toutes les hypothèses

1. Pour obtenir le code, il faut écrire à csjcg2@gmail.com.

de mots valides sont initialisées à 1, puis elles sont calculées itérativement par l'algorithme 21.

Algorithm 21 : Iterative Calculating for LBV and RBV

Input : mot arabe w, $G(S, E_{LN}, E_{RN})$, n, k
Output : $(LBV^{(k)}, RBV^{(k)})$
Data :
Function `Stem(`w`)` **:**
  Initialize $LBV^{(0)}$, $RBV^{(0)}$;
  for $i \leftarrow 1$ *to* k **do**
    for $j \leftarrow 1$ *to* n **do**
      Mettre à jour $LBV(S[j])^{(i)}$ grâce à l'équation 4.1 ;
    for $j \leftarrow 1$ *to* n **do**
      Mettre à jour $RBV(S[j])^{(i)}$ according to Eq.;
    for $j \leftarrow 1$ *to* n **do**
      Normaliser $LBV(S[j])^{(i)}$ according to Eq.;
    for $j \leftarrow 1$ *to* n **do**
      Normaliser $RBV(S[j])^{(i)}$ according to Eq.;
  return $(LBV^{(k)}, RBV^{(k)})$;
return $(LBV^{(k)}, RBV^{(k)})$;

Le score EBV est calculé comme suit[1] :

$$EBV(w) = \overrightarrow{LBV(w)} \cdot \overrightarrow{RBV(w)} \tag{4.1}$$

L'EBV ne suffit pas car les combinaisons de mots (par exemple, les collocations) auraient des scores élevés de LBV et RBV pour leurs limites correctes gauche et droite, ce qui entraîne un biais en faveur des mots longs lors de la recherche de la segmentation optimale. En effet, les valeurs de bordure extérieure (LBV et RBV) se concentrent uniquement sur l'extérieur des hypothèses de mots, en ignorant l'intérieur. Ils comblent cette lacune en incorporant un indicateur de la limite à l'intérieur des hypothèses de mots, appelé Valeur de Bordure Intérieure (IBV), basé sur l'observation suivante : « Il existe généralement des indicateurs forts de limites de mots à l'intérieur des combinaisons de mots, tandis qu'ils sont rares dans les mots réels. »

Ils ont appliqué l'algorithme de Viterbi (cf sect. 4.3.2) pour la recherche de la segmentation optimale avec les scores WordRank, en maximisant la fonction

1. Différentes combinaisons ont été testées, avec l'addition, le minimum et la multiplication. La différence est négligeable.

objective suivante :

$$S^* = \arg max_{w_1,..w_n} = u \prod_{i=1}^{N} WR(w_i) \tag{4.2}$$

Dans cette formulation, où u représente une séquence continue de caractères, $w_1 \oplus \cdots \oplus w_n$ est la segmentation possible de u où w_i est une hypothèse de mot valide. $WR(w)$ est le score WordRank associé à w. Puisqu'ils ont considéré la segmentation des mots comme un problème d'optimisation, la segmentation avec la valeur la plus élevée est la solution résultante.

4.2.2 Sebawai

K. Darwish (2002a) propose Sebawai[1], l'un des premiers analyseurs multiplateforme open-source, et également l'un des premiers à générer automatiquement les règles ; K. Darwish et Oard (2003) montrent ses performances dans la recherche d'information en le comparant à Alpnet (sect. 3.2.1). Voir aussi K. Darwish (2002b). À la différence d'Al-Stem (qui deviendra Light10, sect. 3.2.8), qu'il réalise à la même époque, c'est un analyseur morphologique « peu profond » (*shallow*).

Le système commence par collecter des statistiques à partir de paires mot-racine, dont la liste peut être constituée soit manuellement, en utilisant un dictionnaire électronique, soit en utilisant un analyseur morphologique préexistant tel que ALPNET (3.2.1) ou MORPHO3 (3.2.4) ; c'est cette dernière solution que l'auteur a choisie.

Un premier module, assurant l'entraînement, compare chaque mot à sa racine et détermine préfixe, suffixe et schème du radical. Par exemple, pour la paire {وكتابهم |wkt?bhm|, كتب |ktb|}, il génère (entre autres) و |w| comme préfixe, هم |hm| comme suffixe et فعال (CCāC) comme schème, et il accroît leur fréquence de 1. Il tient compte des préfixes ou suffixes vides avec le symbole #. Les schèmes Sebawai sont légèrement différents des schèmes souvent utilisés, car les radicaux sont obtenus une fois les clitiques et affixes enlevés. Les schèmes utilisés par les autres modèles incluent en général les affixes (pas les clitiques). Par exemple, le schème usuel mCCwC inclut le préfixe م (m-). Le schème découvert automatiquement par Sebawai est simplement فعول CCwC.

À la fin du traitement, ce module a :

- dérivé une liste de préfixes et de suffixes ;
- construit des schèmes de radicaux ;
- calculé la probabilité d'apparition de préfixes, suffixes ou schèmes.

1. Le code source est disponible sur https://github.com/NNLP-IL/Arabic-Resources.

Les probabilités suivantes sont calculées sur la base des fréquences (S est une chaîne et T une séquence CCC avec éléments insérés) :

- P(S commence un mot, S est un préfixe).
- P(S termine un mot, S est un suffixe).
- P(T est un schème)
- P(S est un préfixe | S commence un mot).
- P(S est un suffixe | S termine un mot).

Le deuxième module construit à partir d'un mot les combinaisons possibles de préfixes-suffixes-schèmes, et génère une liste classée de racines possibles. La racine est détectée à partir des hypothèses de radicaux et des schèmes qu'on peut y découvrir. Par exemple, pour le mot ايمان **ʔymān** (qui n'a pas de clitiques), les hypothèses sont indiquées dans le tableau 4.2.

Tab. 4.2 : Hypothèses de radicaux

Stem	Préfixe	Schème	Suffixe	Racine
ايمان	#	فعال CyCāC	#	امن
يمان	ا	فعال CCāC	#	يمن
مان	اي	فعل CCC	#	مان
ايم	#	فعل CCC	ان	ايم
يما	ا	فعل CCC	ان	يما

Le système a considéré comme non réalisables les formes ايما |ʔymā| et يم |ym|. C'est juste pour la première, mais la deuxième est en fait réalisable (racine يمم |ymm|), mais le système ne savait pas comment le traiter. Les hypothèses sont ordonnées en fonction du produit des probabilités d'apparition des éléments. Ceci suppose que leurs probabilités sont indépendantes, ce qui n'est pas totalement correct car certaines combinaisons ne sont pas autorisées. L'utilisation du système nécessite un lissage. Les racines générées sont vérifiées dans une liste de 10 000 racines extraites automatiquement d'une copie électronique du dictionnaire *Lisan al-Arab*.

Algorithm 22 : Sebawai

Input : mot w
Output : racine
Data : $A \leftarrow$ affixes ; $R \leftarrow$ racines ; $M \leftarrow$ Mots invariables ;
Function `Stem(`w`)` **:**
  if $w \in M$ **then**
    return Null ;
  Normalisation
    Remplacer à la fin de w ى par ي ;
    Remplacer au début de w {إ أ آ ؤ ئ ء} par ا ;
  if *P(S commence w) > seuil* **then**
    $w \leftarrow w - S$
  if *P(S termine w) > seuil* **then**
    $w \leftarrow w - S$
  if *P(T est un schème) > seuil* **then**
    return w - T
  else
    return Null ;

4.3 Segmenteurs contextuels par sélection

Ces segmenteurs utilisent un analyseur morphologique parmi les segmenteurs non contextuels déjà vus pour générer des segmentations puis sélectionnent la bonne segmentation en fonction du contexte. BAMAE/BASMA (sect. 4.3.1) se base sur l'analyseur BAMA (SAMA) (sect. 3.2.7), et sélectionne une analyse par des règles linguistiques et la probabilité des séquences ; SAFAR (sect. 4.3.2) se base sur son propre analyseur, et sélectionne par un modèle probabiliste (Viterbi sur HMM, qui est aussi le modèle de sélection choisi dans Kalimat 2.4.1, dans un algorithme malheureusement non disponible).

Tab. 4.3 : Segmenteurs contextuels par sélection

Segmenteur	Input	Output	Date	ML	Langage
BAMAE, BASMA			2007-2015	Règles & Prob.	
Safar		Lemm.	2015	HMM, Viterbi	Java

4.3.1 BAMAE, BASMA

Alansary a développé deux modèles pour désambiguïser les sorties du célèbre analyseur morphologique développé par Buckwalter (sect. 3.2.7). Tout d'abord, en utilisant des règles linguistiques basées sur les étiquettes des mots voisins extraites du corpus *International corpus of Arabic* (sect. 2.5.5), il a développé le Buckwalter Arabic Morphological Analyzer Enhancer (BAMAE) (Alansary, 2012), conçu pour :

- corriger les limitations du BAMA original ;
- compléter des informations morphologiques manquantes ;
- désambiguïser les multiples solutions proposées.

La sortie de l'algorithme est :

- Lemme
- Racine
- Schème du radical
- Nombre, genre, détermination
- Cas grammatical ou mode
- Vocalisation

La désambiguïsation est effectuée sur trois niveaux à partir de la sortie enrichie de BAMA (le code de l'analyseur original a dû être modifié pour se prêter à ce traitement) :

- Niveau interne : éliminer les solutions impliquant des concaténations incompatibles entre radical et affixes.
- Niveau contextuel : des règles linguistiques ad hoc sont appliquées en fonction des étiquettes des mots successifs.
- Niveau fréquentiel : si les deux niveaux précédents n'ont pas permis de déterminer la meilleure solution, prendre la solution la plus fréquente.

BAMAE atteint une précision de 92% et un rappel de 89%.

(Alansary, 2019) a perfectionné l'algorithme sous le nom de BibAlex Standard Arabic Morphological Analyzer (BASMA), à partir de la Bibliotheca Alexandrina, dans le même but, intégrant davantage de statistiques de corpus. La figure 4.1 présente l'architecture de BASMA, en partant du texte d'entrée et des solutions proposées pour chaque mot, afin de prédire la meilleure catégorie grammaticale pour chaque mot, et d'en déduire les autres informations.

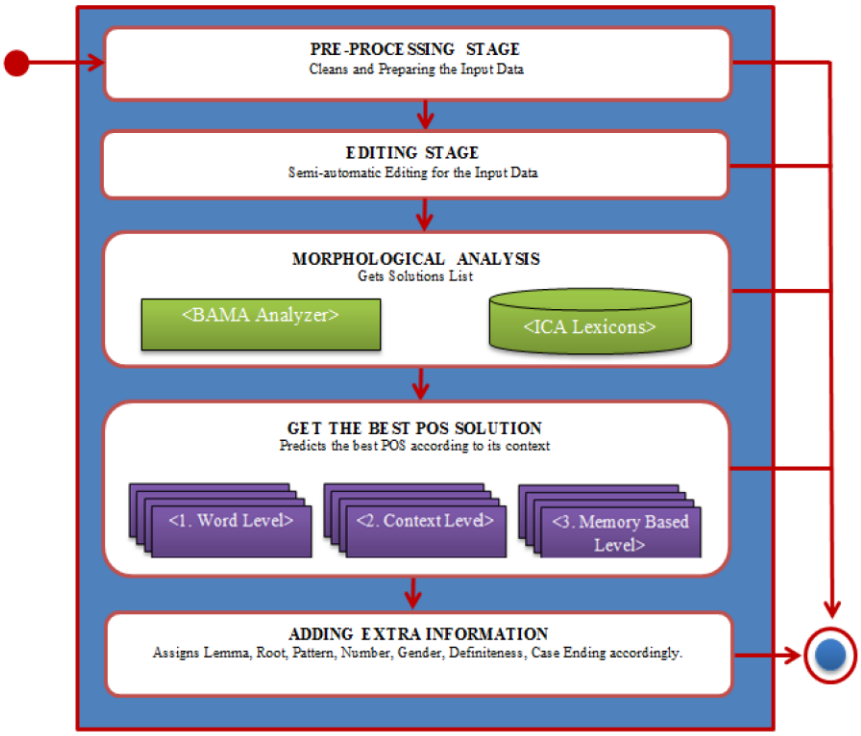


Fig. 4.1 : Architecture de BASMA (Alansary, 2019)

La différence entre BASMA et BAMAE n'est pas très explicite ; les deux algorithmes sont pratiquement identiques (alg. 23). BASMA semble donner une plus grande importance à la catégorie grammaticale, et surtout, reposer sur un beaucoup plus grand ensemble de données textuelles pour le calcul des fréquences. Pour évaluer BASMA, un jeu de données de test à l'aveugle (1 000 000 de mots représentatifs) a été traité, et les résultats ont été comparés à une version annotée manuellement. Le taux de précision obtenu était de 93,37 %, tandis que le taux de rappel atteignait 96,9 %. Ces pourcentages devaient augmenter avec la mise en œuvre des améliorations prévues et l'exploitation de volumes de données plus importants.

Algorithm 23 : BASMA : BibAlex Standard Modern Arabic Morphological Analyzer

Input : mot w
Output : Analyse morphologique enrichie M
Data : Lexiques enrichis (préfixes, stems, suffixes, racines, schèmes), règles linguistiques, tables de compatibilité

Function `BASMA(`w`)` **:**
  Analyse initiale
    $C \leftarrow$ toutes les analyses candidates de w par BAMA (modifié) ;
  Correction lexicale
    Corriger les erreurs de segmentation, lemmes, racines, glosses ;
    Ajouter racine et schème au résultat ;
  Application des règles linguistiques
    foreach *analyse* $c \in C$ **do**
      Ajuster {nombre, genre} selon POS et schème ;
      Ajuster définitude et cas selon contexte ;
  Désambiguïsation
    if $|C| > 1$ **then**
      Appliquer règles de désambiguïsation (mot, contexte, mémoire) ;
      $c^* \leftarrow$ meilleure analyse ;
    else
      $c^* \leftarrow C$;
  Enrichissement
    Ajouter traits supplémentaires : lemme, racine, schème, entité nommée, diacritisation ;
  return c^* ;

4.3.2 SAFAR

Jaafar et Bouzoubaa (2015) ont développé SAFAR[1] (Software Architecture for Arabic), une plateforme en Java dédiée au traitement automatique de l'arabe, incluant un segmenteur. Cette plateforme est orientée sur l'interopérabilité et la réutilisabilité, utilisant XML, la TEI (Burnard et Bauman, 2008), LMF (ISO 24613:2008) ainsi que les normes recommandées par l'Organisation Arabe pour l'Éducation, la Culture et la Science (ALECSO). Elle intègre aussi des ressources (corpus, dictionnaires) et d'autres outils (syntaxiques, sémantiques) pour l'ASM d'une part et pour le dialecte marocain de l'autre. Elle permet également le benchmarking des segmenteurs (Jaafar et Bouzoubaa, 2014), et l'intégration de nouveaux corpus ; plus de détails dans Bouzoubaa et al. (2021).

1. Le code source de Safar est disponible sur https://github.com/arabic-digital-humanities/adhtools.

Leur analyseur (voir figure 4.2) prend en entrée une phrase, la normalise et élimine les caractères n'appartenant pas au script arabe, lance ensuite sur chaque mot un segmenteur de clitiques contrôlé par un lexique de mots minimaux, puis une sélection du bon lemme par une procédure probabiliste. La partie segmentation non contextuelle donne en sortie toutes les segmentations possibles, mais la sortie de l'algorithme contextuel est uniquement le lemme, et ne donne pas la segmentation qui a permis d'y arriver.

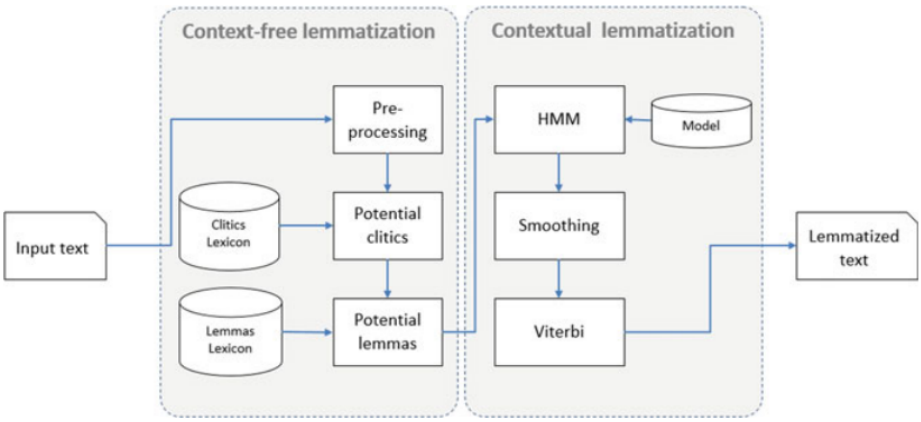


Fig. 4.2 : Architecture du lemmatiseur SAFAR, Jaafar et Bouzoubaa (2015)

Le segmenteur de clitiques extrait du mot toutes les séquences possibles proclitiques - mot minimal - enclitique, puis élimine celles dont le mot minimal ne figure pas dans leur lexique, qui contient aussi 720 particules (mots invariables). Par exemple, comme on le voit dans la figure 4.3, le mot سيكتبهم a quatre découpages possibles, dont seul le plus à gauche ({س - يكتب - هم}) contient un mot minimal figurant dans le lexique, avec cinq lemmes possibles (faire écrire, écrire, dicter, plier, apprendre à écrire). Afin de couvrir les cas non répertoriés, le lemme d'un mot hors lexique est lui-même.

À l'étape suivante, la bonne analyse et le bon lemme sont sélectionnés par une combinaison HMM/Viterbi (Namly et al., 2020). Cette étape est réalisée en utilisant un Modèle de Markov Caché (HMM), une technique de lissage et l'algorithme de Viterbi(Viterbi, 1967). Ce choix s'explique par le fait que la technique HMM a déjà prouvé son efficacité dans ce domaine (B. Mohamed et Azzeddine, 2016) ; il est confirmé par ailleurs par la comparaison des résultats de Safar avec ceux du modèle de langage généraliste proposé par Song et Croft (1999).

Le HMM est ici défini par les paramètres suivants :

- un ensemble de lemmes l_i qui constituent les états cachés ;

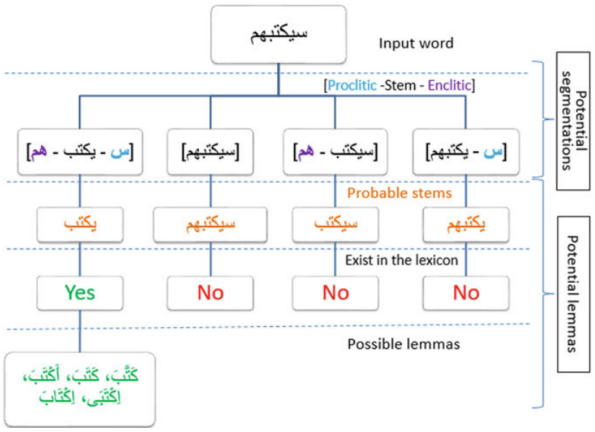


Fig. 4.3 : Lemmatisation sans contexte de SAFAR, Jaafar et Bouzoubaa (2015)

- π_i : probabilité de chaque l_i ;
- a_{ij} : probabilité de transition de l_i vers l_j ;
- $b_i(t)$: probabilité d'émission du mot t depuis l_i ;
- une séquence de mots t_k observée.

Pour les mots inconnus du corpus (forcément en nombre non négligeable), les probabilités seront au départ à zéro, ce qui pose problème pour la recherche du chemin optimal. D'où l'intérêt d'une technique de lissage attribuant une probabilité non nulle à tous les éléments de ces matrices. Pour cela, les auteurs utilisent la méthode de réduction absolue (Ney et Essen, 1991) avec la même logique que chez B. Mohamed et Azzeddine (2016) sauf pour le paramètre Z_i. Les probabilités initiales, de transition et d'émission sont ajustées par réduction absolue, afin d'éviter les zéros dans les matrices, en modifiant au départ les probabilités d'émission de la manière suivante :

$$b_i(t) = \begin{cases} \frac{freq_{i,t} - 0.5}{freq(l_i)} & \text{si } freq_{i,t} \neq 0 \\ \frac{N \cdot 0.5}{freq(l_i) \cdot Z_i} & \text{sinon} \end{cases} \tag{4.3}$$

où $freq_{i,t}$ est le nombre d'occurrences du mot t émis par l_i, N le nombre total de mots, Z_i nombre de mots observés non annotés avec l_i.

La probabilité de transition est initialisée ainsi :

$$a_{ij} = \frac{\max(freq_{ij} - 0.5, 0)}{n_i} + \frac{0.5}{n_i} \cdot \frac{n_j}{N \cdot N_{l_i}^{+}} \tag{4.4}$$

où $freq_{ij}$ est le nombre de transitions observées de l_i vers l_j, n_i le nombre total de transitions sortantes de l_i, $N_{l_i}^{+}$ le nombre de mots pour lesquels le lemme l_j apparaît après l_i.

Et la probabilité initiale π_i est initialisée comme suit :

$$\pi_i = \frac{\max(m_i - 0.5, 0)}{M} + \frac{0.5}{M} \cdot \frac{freq(l_i)}{N \cdot N_i^+} \tag{4.5}$$

où m_i : nombre d'occurrences du lemme l_i en début de séquence, M le nombre total de phrases, N le nombre total de mots, N_i^+ : nombre de mots pour lesquels le lemme l_i apparaît en position initiale.

L'algorithme de Viterbi (Viterbi, 1967) (algorithme 24) est une méthode de programmation dynamique qui fournit une solution optimale par maximum de vraisemblance pour estimer la séquence d'états d'un HMM. À un instant t, $\delta_t(i)$ désigne la probabilité maximale d'obtenir le lemme l_i, $\psi_t(i)$ désigne le lemme précédent dans la séquence la plus probable, o_t désigne le mot observé.

Algorithm 24 : Viterbi pour HMM (Forney, 1973)

1 **Initialisation :** $\forall$ lemme $i \in [1, \dots N]$:

$$\delta_1(i) = \pi_i \cdot b_i(o_1), \quad \psi_1(i) = 0$$

2 **Récursion :** $\forall\ t \in [2, \dots T] \wedge \forall$ lemme $j \in [1, \dots N]$:

$$\delta_t(j) = \max_i \left[\delta_{t-1}(i) \cdot a_{ij}\right] \cdot b_j(o_t)$$

$$\psi_t(j) = \arg\max_i \left[\delta_{t-1}(i) \cdot a_{ij}\right]$$

3 **Valeurs finales :**

$$\Delta = \max_i \delta_T(i), L_T = \arg\max_i \delta_T(i)$$

4 **Lemmes antérieurs :** $\forall\ t \in [T-1, T-2, \dots 1]$:

$$L_t = \psi_{t+1}(L_{t+1})$$

4.4 Segmenteurs contextuels probabilistes

Les analyseurs de cette catégorie calculent la segmentation la plus probable sur la base d'informations non contextuelles (en particulier, la probabilité des associations clitiques / mot minimal et des associations affixes, radical et schème), ainsi que sur la base d'informations contextuelles. Ils utilisent des modèles probabilistes

et éventuellement de l'apprentissage automatique non neuronal. Madamira constitue un vecteur de caractéristiques dont certaines sont contextuelles transmises à des SVM (Vapnik, 1995) ; concernant Farasa, qui utilise SVMRank, la description ne permet pas de savoir à quel endroit intervient la prise en compte du contexte, la partie non contextuelle est en revanche décrite en détail (la sensibilité au contexte n'apparaît que lors des tests). Stanford utilise un CRF pour gérer les informations non contextuelles et des règles linguistiques pour intégrer les informations contextuelles sur les catégories grammaticales des mots voisins. Sous une forme ou sous une autre, ces segmenteurs combinent des modèles probabilistes et des informations linguistiques fournies par les experts.

Tab. 4.4 : Segmenteurs contextuels probabilistes

Segmenteur	**Input**	**Output**	**Date**	**ML**	**Langage**
Mada, Madamira			2009-2014	SVM & Mod. Lang.	Java
Farasa			2016	SVM Rank	Java
Stanford	Simple	Règles, CRF.		2014	Java

4.4.1 MADA, MADAMIRA

MADA, réalisé par Habash, Rambow et Roth (2009) et dont la dernière version s'intitule MADAMIRA[1] (Pasha et al., 2014), utilise, pour produire un ensemble d'analyses pour chaque mot, les données et règles de BAMA (sect. 3.2.7), mais avec un moteur différent, à l'origine appelé Almorgeana (Habash, 2007), qui a été intégré sous l'appellation Almor. Pour sélectionner la meilleure analyse, dix-neuf caractéristiques sont calculées pour chacune des analyses proposées dans la première phase :

- un ensemble de 14 *Support Vector Machines* (Fan et al., 2008 ; Waldvogel, 2008) entraînés sur ATB (sect. 2.5.2) prédit 14 caractéristiques (POS, nombre, genre, personne...)
- 3 modèles de langage basés sur les ngrammes de caractères prédisent le lemme, les diacritiques et le mot minimal ;
- 1 caractéristique est déterminée par le fait que c'est ou non l'analyse par défaut pour Almor ;

1. Pour obtenir le code, suivre la procédure sur https://github.com/CAMeL-Lab/WIDH_2020_Arabic_Text_Analysis/tree/master/MADAMIRA, qui implique de demander une licence pour usage non commercial à l'université de Columbia.

- et 1 caractéristique (la validité de la séquence de diacritiques) est calculée par règles.

Chacune de ces caractéristiques est pondérée par heuristique (mais l'utilisateur peut régler lui-même ces pondérations), ce qui permet d'attribuer un score à chaque analyse. Celle dont le score est le plus élevé est sélectionnée. Pour prendre en compte les mots inexistants dans le lexique, Mada peut générer des hypothèses de segmentation et les ajouter aux hypothèses d'Almor. En sortie, il fournit une analyse morphologique complète, avec lemme, segmentation, mot minimal, forme vocalisée.

La version Madamira intègre certains composants inspirés par Amira (Diab, 2010) tout en introduisant des améliorations techniques, comme une implémentation java plus rapide et prévue pour être exécutée dans une architecture client-serveur, le remplacement de l'appel aux logiciels tiers qui étaient utilisés dans Mada et Amira par des composants internes, et le remplacement des SVM standards de la première version par des SVM linéaires, le remplacement des données et règles de BAMA par celles de SAMA ; elle ajoute également la segmentation des mots du dialecte égyptien, avec l'analyseur CALIMA.

L'entraînement a été effectué sur le corpus ATB (parties 1, 2 et 3) (sect. 2.5.2) pour l'ASM ; pour plus de détails sur la répartition en termes d'entraînement, de développement et de test, voir Diab, Habash et al. (2013). MADAMIRA a été testée sur un corpus de référence annoté (environ 25 000 mots pour l'ASM) et a obtenu une exactitude (*accuracy*) de 99,2% pour la segmentation.

4.4.2 Farasa

Abdelali, K. Darwish et al. (2016) ont proposé FARASA[1], un système de traitement de l'arabe, incluant un segmenteur de clitiques avec sélection de la meilleure analyse, tout comme Madamira. Les auteurs soulignent qu'il est beaucoup plus rapide que Madamira ou Stanford. Lors d'un benchmark interne réalisé sur un corpus de 7,4 millions de mots, FARASA a mis 129 secondes contre plus de 2,5 heures pour Madamira. Pour la performance, Alfaidi et al. (2023) ont comparé Farasa à Madamira et CAMeL Tools, et concluent qu'il fonctionne bien pour l'ASM, bien que moins bien sur les dialectes arabes. En sortie, il fournit simplement une segmentation (pas de lemme ni de forme vocalisée).

Le segmenteur de clitiques, après normalisation, produit toutes les combinaisons possibles de proclitiques (ف, و, ل, ب, ك, ال, س) et d'enclitiques (ا, ة, ت, ك, ن, و, ي, ات, ان, ون, وا, ين, كما, كن, كم, ه, ها, هما, هم, هن, نا, تما, تم, تن) (appelés respectivement préfixes et suffixes) ; sont écartées celles qui sont interdites par un ensemble de règles

1. Le code de FARASA est disponible sur https://farasa.qcri.org/segmentation.

(ex. l'article est forcément le dernier proclitique, et n'est pas compatible avec les enclitiques pronominaux). Dans chacune de ces combinaisons, un mot minimal est postulé (voir l'algorithme 25).

Puis un certain nombre de caractéristiques sont calculées pour chaque analyse :

- probabilité qu'une séquence de caractères initiale soit un préfixe ;
- probabilité qu'une séquence de caractères finale soit un suffixe ;
- probabilité du mot minimal ;
- probabilité de trouver ce mot minimal suivi de suffixe ;
- probabilité conditionnelle d'un préfixe étant donné un suffixe ;
- probabilité conditionnelle d'un suffixe étant donné un préfixe ;
- booléen : le mot minimal (ex. كتاب) peut-il être produit par un modèle à partir d'un schème (ex. **CCāC**) et d'une racine (ex. **ktb**) ; les modèles sont produits par la procédure décrite dans K. Darwish, Abdelali et Mubarak (2014) ;
- booléen : le mot minimal apparaît-il dans le corpus Aljazeera (sect. 2.2.9) ;
- booléen : le mot minimal apparaît-il dans un lexique de noms de personnes et de lieux extrait de Wikipedia par la procédure décrite dans K. Darwish, Magdy et A. Mourad (2012) ;
- booléen : le mot minimal apparaît-il dans AraComLex, lexique avec 31 753 lemmes, dont 24 976 noms et 6 777 verbes (Attia, Pecina et al., 2011) ;
- booléen : le mot minimal apparaît-il dans le lexique d'Aramorph (sect. 3.2.7) ;
- écart entre la longueur du mot minimal et la longueur moyenne des mots.

Les poids des caractéristiques sont apprises avec le classificateur *Support Vector Machine for Ranking* (Joachims, 2006), avec un noyau linéaire et un facteur de compromis entre les erreurs d'entraînement et la marge égal à 100, choisi après des expériences effectuées sur un ensemble de développement. L'analyse la mieux classée est sélectionnée.

Farasa présente deux variantes. Dans la première, *Farasa_Base*, le classificateur est utilisé pour classer toutes les analyses. Elle utilise également une petite liste de mots invariables suivis d'enclitiques qui modifient leur forme, par exemple مما (**mimā**), provenant de من+ ما (**min + mā**). Dans le deuxième, *Farasa_Lookup*, le

classificateur n'est pas utilisé pour les mots qui, pendant l'entraînement, n'ont eu qu'une seule segmentation ou qui apparaissent au moins 5 fois avec la même segmentation dans plus de 70% de leurs occurrences.

L'entraînement a été effectué sur le corpus PATB (parties 1, 2 et 3) (sect. 2.5.2). Les modèles de langage unigramme sont générés à partir d'un corpus d'Aljazeera, composé de 114 758 articles avec 94 millions de mots. Les auteurs font remarquer que les tests effectués sur un sous-ensemble de l'ATB posent problème en raison de sa diversité lexicale limitée, d'où des résultats artificiellement élevés. Ils ont donc créé un nouvel ensemble de test composé de 70 articles de Wiki-News (de 2013 et 2014) couvrant sept thèmes (politique, économie, santé, science et technologie, sport, arts et culture) de manière équilibrée, avec 18 271 mots. Les résultats des deux variantes de Farasa sont comparés avec ceux de Madamira. Selon eux, *Farasa_lookup* (accuracy 98.94%) fonctionne légèrement mieux que Madamira (accuracy 98.76%).

Algorithm 25 : Farasa segmenter

Input : mot w
Output : segmentation préfixes, radical, suffixes
Data : Lexiques de préfixes, suffixes, racines ; règles et schèmes ;
Function `Segmenter(`w`)` **:**
  Normalisation ;
  Génération des segmentations
    $C \leftarrow$ combinaisons de préfixes, radicaux et suffixes qui peuvent donner w ;
    Supprimer de C les combinaisons invalides selon les règles ;
  Extraction des caractéristiques
    foreach $c \in C$ **do**
      Calculer la probabilité des préfixes, radicaux et suffixes ;
      Utiliser les lexiques (fréquences) ;
      Évaluer la conformité aux schèmes morphologiques ;
  foreach $c \in C$ **do**
    Appliquer SVM-rank ;
  return segmentation avec score le plus élevé

4.4.3 Stanford NLP

Le segmenteur open-source Stanford[1] (Monroe, Green et C. D. Manning, 2014, incluant l'arabe) (Toutanova et al., 2003, original), fait partie du pipeline de trai-

1. Le code source de Stanford est disponible sur http://nlp.stanford.edu/software/segmenter.shtml.

tement du langage humain de l'Université de Stanford pour l'arabe, qui comprend aussi un tokeniseur, une reconnaissance d'entités nommées, un découpage en phrases et un parseur de dépendances syntaxiques. Il peut gérer une grande variété de textes et de dialectes arabes, y compris l'arabe standard moderne et les dialectes parlés, avec des performances élevées et des taux d'erreur faibles. Il n'effectue que l'analyse en clitiques.

Ce segmenteur utilise un modèle d'entropie maximale qui est entraîné sur un grand corpus de textes arabes pour effectuer la segmentation, accompagné de règles manuelles. Il faut noter qu'il travaille au niveau des caractères, en examinant les trois caractères précédant le caractère à traiter, et au niveau des catégories grammaticales des mots voisins avec les règles manuelles.

Plus précisément, il utilise le classificateur de séquence à champ aléatoire conditionnel (CRF) (J. Lafferty, McCallum et Pereira, 2001) qui définit une distribution $p(Y|X;\theta)$, où X = $\{x_1 ... x_N\}$ est la séquence d'entrée observée et Y = $\{y_1 ... y_N\}$ est la séquence d'étiquettes à prédire. Y^* est choisi selon la règle de décision suivante :

$$Y^* = \arg max_Y \sum_{i=1}^{N} \theta^T \phi(X, y_i, ..., y_{i-3}, i) \tag{4.6}$$

avec ϕ une carte des caractéristiques (feature map) employées par le modèle CRF markovien de troisième ordre (4-gramme) du segmenteur, et les y_i des étiquettes qui peuvent avoir l'une des valeurs suivantes :

- I : continuation d'un segment,
- O : espace à l'extérieur d'un segment,
- B : début d'un segment,
- F : caractères étrangers pré-groupés,
- Rew : gère des règles orthographiques spécifiques à l'arabe.

L'étiquette *Rew* entraîne les réécritures orthographiques suivantes :

- Si on segmente le proclitique لـ, et le caractère suivant est ل, il faut commencer un nouveau segment, l'article défini ال.
- Si on segmente un enclitique après ت, le réécrire en ة.
- Si on segmente un enclitique après ي, le réécrire en ى.

4.5 Segmenteurs contextuels neuronaux

Un segmenteur contextuel neuronal devrait apprendre automatiquement les règles de segmentation sur la base d'un corpus, par pure induction à partir des données.

Il n'y en a qu'un seul, à notre connaissance, Stanza, qui utilise une cascade de bi-LSTM. Il est intéressant en soi, mais la partie contextuelle ne fonctionne pas pour l'arabe, où seule la partie non contextuelle avec détection du radical et du lemme fonctionne.

Stanza (Qi, Dozat et al., 2018 ; Qi, Y. Zhang et al., 2020) est une bibliothèque open-source de traitement du langage naturel (NLP) en Python[1]. Elle constitue un pipeline (architecture fig. 4.4) qui génère des annotations à partir de texte brut, en se basant sur des réseaux neuronaux, et ce, sans avoir besoin de connaissances linguistiques préalables. Pour la partie qui nous intéresse, à savoir la tokenisation et la segmentation pour les langues de type arabe, le processus se déroule en deux étapes :

- Lors de la tokenisation, les mots sont classés en mots minimaux et non minimaux (appelés « multi-words »).
- Ensuite, les mots non minimaux sont séparés en composants. La tokenisation des phrases est effectuée en même temps.

Comme pour Stanford, le contexte pris en compte est limité aux caractères voisins du segment considéré, du moins dans la première étape.

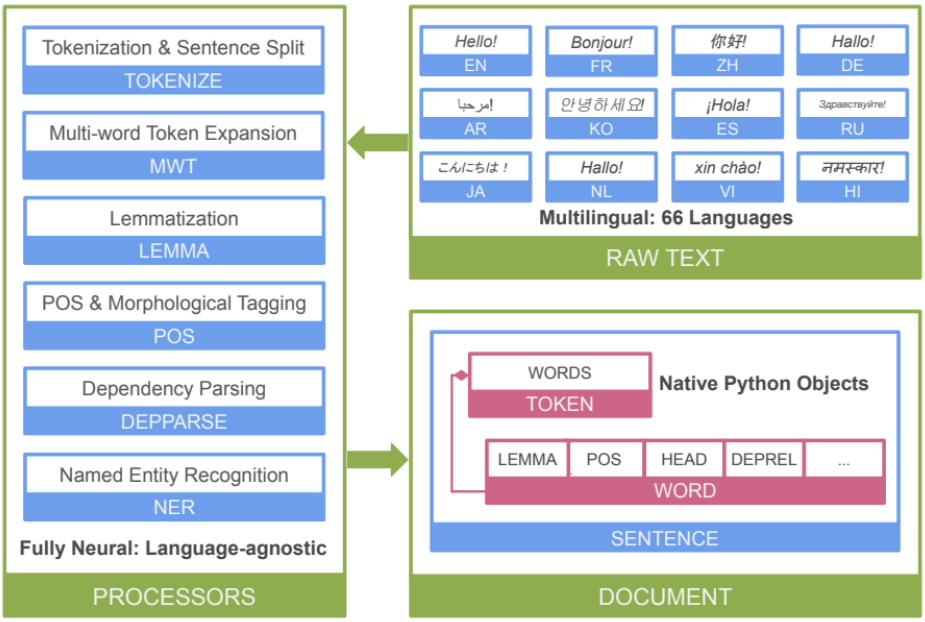


Fig. 4.4 : Vue d'ensemble du pipeline NLP neuronal de Stanza

1. Le code est disponible https://stanfordnlp.github.io/stanza/.

La première étape est réalisée à l'aide d'un réseau neuronal LSTM (voir sect. 4.6.1) bidirectionnel (BiLSTM) qui étiquette chaque caractère d'un mot comme fin de token, fin de phrase, membre d'un mot minimal, membre d'un mot non minimal ou autre, en se basant sur les espaces blancs et la distinction alphanumérique. Une fois toutes les tokenisations possibles réalisées, on effectue toutes les segmentations possibles des mots non minimaux. La troisième étape (lemmatisation) utilise un autre BiLSTM avec un perceptron multicouche (MLP) pour l'attention, dont la sortie est une liste des morphèmes contenus dans le mot. Enfin, une dernière couche de BiLSTM avec des classificateurs binaires étiquette les tokens eux-mêmes.

La nature des données d'apprentissage pour l'arabe n'est pas détaillée, la seule référence étant le Prague Arabic Dependency Treebank (sect. 2.5.3).

4.6 Modèles d'apprentissage séquentiel

Dans ce qui précède, nous avons déjà présenté les algorithmes à base de règles et à base probabiliste (CRF, HMM, Viterbi) utilisés pour la segmentation de l'arabe. Dans cette section, nous présentons les modèles utilisés pour l'apprentissage neuronal de la segmentation. Il s'agit essentiellement de LSTM et de ses variantes (le seul modèle utilisé jusqu'à présent), auquel nous ajoutons son concurrent plus récent et qui va nous intéresser plus loin, GRU (sect. 4.6.2), qui sont les algorithmes de référence dans le domaine selon Goodfellow, Bengio et Courville (2016).

4.6.1 LSTM

LSTM (Long Short-Term Memory) est un modèle neuronal récurrent conçu par Hochreiter et Schmidhuber (1997) pour capturer les dépendances à long terme dans les données séquentielles, tout en évitant le problème de disparition du gradient. Il est largement utilisé dans le traitement du langage naturel et l'analyse de séries temporelles.

Dans l'architecture se succèdent des états cachés mémorisant des données antérieurement transmises. À un instant t, chaque cellule c reçoit une entrée x_t, possède un état c_t, une sortie antérieure h_{t-1} et trois portes multiplicatives, f_t, i_t et o_t, qui servent à calculer la sortie h_t :

- Porte d'oubli (Forget Gate) : décide ce qui doit être supprimé des états précédents

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.7}$$

  - f_t : ce qu'on va garder des états antérieurs ;

  - σ : fonction sigmoïde, qui assure que les sorties sont comprises entre 0 et 1;
  - h_{t-1} : état caché précédent ;
  - x_t : entrée actuelle
  - W_f et b_f : sa matrice de poids et son biais.

- Porte d'entrée (Input Gate) : décide quelles informations doivent être ajoutées à la mémoire.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.8}$$

  - i_t : ce qu'on va garder de l'entrée ;
  - W_i et b_i : sa matrice de poids et son biais.

- État caché candidat :

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4.9}$$

  La tangente hyperbolique tanh *permet de centrer les valeurs entre* $[-1, 1]$.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad \text{(Nouvel état de la mémoire)} \tag{4.10}$$

- Porte de sortie (Output Gate) : décide quelles informations doivent être transmises à l'état caché.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.11}$$

$$h_t = o_t \cdot \tanh(C_t) \quad \text{(Nouvel état caché)} \tag{4.12}$$

  L'état caché actuel est obtenu en appliquant tanh *à la mémoire mise à jour* C_t*, filtrée avec la porte de sortie* o_t.

Les calculs sont effectués dans l'ordre suivant :

1. $f_t \leftarrow \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
2. $i_t \leftarrow \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
3. $\tilde{C}_t \leftarrow \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
4. $C_t \leftarrow f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$
5. $o_t \leftarrow \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
6. $h_t \leftarrow o_t \cdot \tanh(C_t)$

4.6.2 GRU

Gated Recurrent Unit (GRU) (Cho et al., 2014) est, comme LSTM, un modèle neuronal récurrent, mais basé sur les transformers et plus simple et plus léger, tout en offrant des performances comparables (A. Zhang et al., 2023).

GRU n'a que deux portes qui modulent dynamiquement la mémoire à chaque instant t ; les conventions de notation sont les mêmes que pour la section précédente, sauf pour les portes r_t et z_t :

- Porte de réinitialisation (Reset Gate) : décide combien d'information passée doit être oubliée.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \quad \text{(quantité d'information passée)} \tag{4.13}$$

- Porte de mise à jour (Update Gate) : décide l'information à conserver du passé.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \quad \text{(quantité d'information conservée)} \tag{4.14}$$

- État caché candidat :

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t]) \tag{4.15}$$

- État caché final (interpolation entre l'état précédent et l'état candidat)

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t \quad \text{(État caché final)} \tag{4.16}$$

Les calculs sont effectués dans l'ordre suivant :

1. $r_t \leftarrow \sigma(W_r \cdot [h_{t-1}, x_t])$
2. $z_t \leftarrow \sigma(W_z \cdot [h_{t-1}, x_t])$
3. $\tilde{h}_t \leftarrow \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t])$
4. $h_t \leftarrow (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$

4.7 Conclusion

Il apparaît qu'il y a finalement peu d'algorithmes contextuels, et aucun algorithme contextuel à base neuronale pour l'apprentissage de la segmentation en arabe. Les techniques utilisées pour les algorithmes contextuels combinent des règles linguistiques et des modèles probabilistes, en particulier les HMM avec Viterbi ou leur forme plus évoluée dans les CRF. Safar avec Viterbi ne fournit malheureusement pas de segmentation désambiguïsée, et la version d'Al-Khalil accompagnée de Viterbi réalisée pour segmenter Kalimat n'a pas fait l'objet d'une publication et son code n'est pas accessible. Il ne reste donc que Madamira, Farasa et Stanford en lice dans la catégorie des algorithmes contextuels disponibles.

Il y a par conséquent la place pour des modèles à base d'apprentissage neuronal, à l'image de Stanza, qui effectueraient une analyse contextuelle sur les mots, avec des algorithmes plus récents, ce qui est notre ambition.

Concernant le type d'apprentissage à utiliser, il est intéressant de comparer les performances des architectures LSTM et GRU. Sellami et Daoui (2025) les ont comparés pour la prévision du cours de clôture de l'indice MASI (Marché des Actions Casablanca) sur une période de cinq ans. Les deux modèles ont été entraînés avec des configurations similaires sur des données multivariées. Les résultats montrent que LSTM surpasse GRU en termes de précision, avec une erreur absolue moyenne (MAE) de 1,6% contre 2,2% pour le GRU. Le coefficient R^2 atteint 0,95 pour LSTM, contre 0,92 pour GRU, indiquant une meilleure capacité de généralisation. Toutefois, GRU présente un avantage en termes de rapidité d'entraînement, ce qui peut être pertinent dans des contextes computationnels contraints. Ces résultats indiquent que LSTM est plus adapté aux séries temporelles financières complexes, bien que le GRU reste une alternative efficace et plus légère.

L'étude de Chung et al. (2014) effectue également une comparaison sur plusieurs tâches de modélisation séquentielle : reconnaissance vocale, modélisation musicale et traitement de texte. Les auteurs montrent que les deux modèles surpassent largement les réseaux récurrents classiques (RNN), avec une amélioration de la perplexité allant jusqu'à 25 à 30% selon les corpus. GRU, bien que plus simple et plus léger, offre des performances comparables en termes de précision, avec une différence souvent inférieure à 1 à 2%, avec une réduction du nombre de paramètres de l'ordre de 20 à 30% et une convergence plus rapide, pouvant atteindre la cible avec 10 à 20% d'itérations en moins. Ces résultats suggèrent que GRU constitue une alternative efficace à LSTM, particulièrement adaptée aux environnements contraints en ressources ou nécessitant une optimisation rapide.

Plus près de notre contexte de recherche, Yang et Zhou (2020) les ont comparé sur un corpus d'avis. L'étude révèle que GRU est environ 29,3% plus rapide à entraîner que LSTM, tout en offrant une meilleure performance sur les petits corpus

contenant des textes longs. En termes de précision, rappel et F-mesure, GRU dépasse LSTM avec des gains respectifs de 23,45%, 27,69% et 26,95%. Sur des corpus volumineux, LSTM conserve une légère supériorité en termes de précision globale. Ces résultats suggèrent que le choix entre LSTM et GRU dépend de la taille et de la nature du corpus : GRU est plus avantageux pour des tâches rapides sur des données textuelles limitées, tandis que LSTM reste robuste pour des corpus massifs et variés. Mais dans le domaine des données textuelles, l'avantage de LSTM reste très faible. D'où notre intérêt pour tester GRU ici.

Chapitre 5

Étiquetage des parties du discours

Sommaire

5.1 Introduction 142
5.2 Modèles pour l'arabe 144
5.2.1 APT de Khoja 144
5.2.2 Arabic SVM POS tagger 145
5.2.3 Salma 147
5.2.4 TreeTagger 148
5.2.5 MADAMIRA POS 149
5.2.6 Farasa POS tagger 149
5.2.7 Etiqueteur par colonie d'abeilles 150
5.2.8 LSTM POS Tagger 150
5.2.9 CAMeL POS tagger 151
5.2.10 Stanford POS Tagger 151
5.2.11 ALP POS tagger 152
5.2.12 Alma 153
5.3 Conclusion 154

5.1 Introduction

Dans ce chapitre, nous allons explorer les techniques d'étiquetage automatique des parties du discours en arabe (en anglais *part of speech tagging*, souvent abrégé en *POS tagging*), en gardant à l'esprit que notre thème principal est la segmentation, et que ce qui nous intéresse dans l'étiquetage des catégories grammaticales sont les interactions possibles avec notre sujet. Notre objectif n'est donc pas un état de l'art aussi complet que nous l'avons tenté pour la segmentation ; nous nous contenterons d'une revue d'ensemble qui a pour but d'indiquer les principes généraux, la manière de surmonter les difficultés particulières liées à l'écriture et à la morphologie de l'arabe, et les modèles les plus récents et les plus efficaces dans ce domaine. Pour plus de détails sur le domaine en général et l'applicabilité des différentes méthodes en fonction des types de langues, on pourra se reporter à Guilder (1995), Karthik Kumar, Sudheer et Avinesh (2006) et Kumawat et Jain (2015), et, plus spécifiquement pour les étiqueteurs pour l'arabe, à Alluhaibi et al. (2021).

Les travaux de référence dans ce domaine soulignent l'importance d'adapter les modèles d'étiquetage aux spécificités linguistiques de l'arabe, notamment sa richesse en flexions et en dérivations, et l'ambiguïté de ses conventions de script ; fondamentalement, on retrouve ici les problèmes évoqués dans notre problématique. Sur le problème particulier des étiquettes, on pourra se reporter utilement à Zeroual, Lakhouaja et Belahbib (2017).

Il existe différentes approches pour l'étiquetage automatique des parties du discours. La figure 5.1 illustre les principales méthodes employées dans ce domaine qui se répartissent en supervisées et non supervisées.

Dans les méthodes supervisées, on trouve trois catégories :

- Les méthodes *rule-based* reposent sur un ensemble de règles linguistiques définies manuellement par des experts, associées à des heuristiques fondées sur des informations lexicales, morphologiques ou syntaxique (Brill, 1992) pour assigner l'étiquette convenable au mot dans son contexte. En combinant ces méthodes avec les stochastiques, les experts peuvent être remplacés par une extraction automatique à partir d'un corpus annoté (Lv et al., 2016), ou une extraction combinée à des patrons de règles définies manuellement (Brill, 1995), appelée *transformation-based learning*.

- Les méthodes stochastiques ou probabilistes se fondent sur des probabilités calculées à partir de corpus annotés, à partir de modèles de langage et/ou de modèles de Markov cachés ou HMM (Brants, 2000, par ex.), qui modélisent les dépendances de chaque étiquette en fonction des étiquettes précédentes, en utilisant des probabilités de transition et d'émission. On citera en exemple

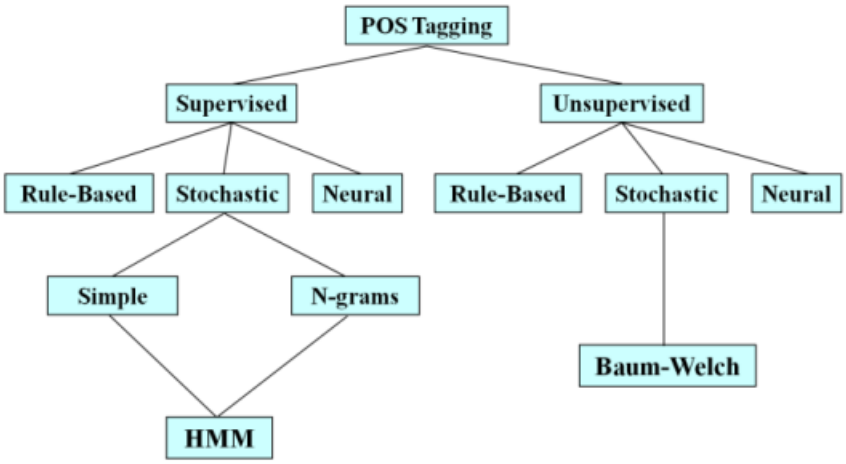


Fig. 5.1 : Catégories d'étiqueteurs POS (Salameh, 2018)

l'algorithme de Viterbi, que nous avons déjà présenté (sect. 4.3.2) appliqué à la segmentation. Les HMM étant limités à capturer des dépendances de premier ordre, on a introduit les champs conditionnels aléatoires, ou CRF, que nous avons également vu dans le cadre de la segmentation (sect. 4.4.3). On retrouve également, comme pour la segmentation, l'utilisation de SVM (Gimenez et Marquez, 2003).

- Les méthodes neuronales, qui reposent essentiellement sur des variantes du perceptron multi-couche, par exemple Marques et Lopes (1996).

Parmi les méthodes non supervisées, on trouve également trois catégories ; mais l'approche rule-based est ici peu représentée. On trouve plus fréquemment des modèles probabilistes, comme l'algorithme Baum-Welch, permettant d'induire les étiquettes et les règles, et surtout, des modèles neuronaux de type récurrents, en particulier ceux qui corrigent le problème de disparition du gradient, comme LSTMs (Long Short-Term Memory) et GRUs (Gated Recurrent Units), présentés au chapitre précédent, et appliqués par exemple dans Alharbi et al. (2019). Les modèles basés sur les transformers, comme BERT (Devlin et al., 2019), qui apprennent une représentation contextuelle bidirectionnelle des mots, surpassent les performances des modèles précédents sur de nombreuses tâches de TALN, dont l'étiquetage POS (Peters et al., 2018).

5.2 Modèles pour l'arabe

Voici les dix principaux étiqueteurs pour l'arabe que nous présentons ici, par ordre chronologique ; comme indiqué ci-dessus, cette liste n'a pas vocation à être exhaustive. Les entrées sont des suites de mots, indiquées S dans les algorithmes, parfois découpées par syntagmes de base (*base phrase chunking*), ou directement extraites, et les sorties une séquence d'étiquettes, notée E dans les algorithmes. On notera que pratiquement tous les étiqueteurs commencent par une phase de normalisation orthographique (comme pour les segmenteurs), que nous ne rappelons pas dans la description. La plupart font à la fois la tokenisation et l'étiquetage, et ceux qui fonctionnent par règles utilisent de la segmentation.

Tab. 5.1 : Étiqueteurs pour l'arabe

Système	**Méthode**	**Date / Version**
APT de Khoja (sect. 5.2.1)	Règles + HMM	2001
Arabic SVM POS Tagger (sect. 5.2.2)	Règles + SVM	2004, 2009 (v2.06 Amira)
Salma POS Tagger (sect. 5.2.3)	Règles	2013
TreeTagger/arabe (sect. 5.2.4)	Arbre de décision	2013
MADAMIRA POS Tagger (sect. 5.2.5)	Règles + MaxEnt	2014 (v2.1)
Farasa POS Tagger (sect. 5.2.6)	SVM + Règles	2016
BCO (Bee Colony Optimization) (sect. 5.2.7)	Colonie d'abeille	2018
LSTM POS Tagger (sect. 5.2.8)	Embedder + biLSTM	2019
Stanford Arabic Tagger (sect. 5.2.10)	ME Markov | CRF	2019 (v2)
CAMeL Tools POS Tagger (sect. 5.2.9)	SVM | CRF | BERT	2021 (v1.0)
ALP POS Tagger (sect. 5.2.11)	HMM + Viterbi	2023
Alma POS Tagger (sect. 5.2.12)	Mémoire	2024

5.2.1 APT de Khoja

Khoja (2001) a développé le premier système d'étiquetage morpho-syntaxique pour l'arabe, APT (Automatic Arabic POS-Tagger). Cet étiqueteur combine les règles avec un modèle probabiliste (HMM+Viterbi) ; voir l'algorithme 26. Le jeu d'étiquettes utilisé dans APT comprenait 177 étiquettes, dont 131 tirées du jeu d'étiquettes du British National Corpus pour l'anglais, combinées avec celles nécessitant par la grammaire de l'arabe (103 noms, 57 verbes, 9 particules, 7 résiduels, 1 ponctuation). APT a atteint un taux de précision de 86% (Abumalloh et al., 2016). Selon Khoja, l'utilisation d'un nombre plus élevé de tags permettrait d'améliorer l'efficacité de l'étiquetage.

Algorithm 26 : APT : Automatic Arabic POS Tagging (Khoja, 2001)

```
Function FirstOutput(S) :
    foreach mot ∈ S do
        Extraire la racine et le schème après suppression des affixes ;
        if racine ∈ lexique then
            assigner étiquette la plus probable ;
        else
            appliquer règles heuristiques d'après le mot voisin à droite et à
              gauche pour inférer l'étiquette ;
Function Output(FirstOutput) :
    Appliquer une désambiguïsation statistique :
        1. Calculer les probabilités bigrammes/trigrammes de mots
        2. Appliquer l'algorithme de Viterbi pour la meilleure étiquette
return Séquence d'étiquettes
```

5.2.2 Arabic SVM POS tagger

Diab, Hacioglu et Jurafsky (2004), s'inspirant pour l'arabe des idées avancées par Hacioglu et Ward (2003), Joachims (1998) et Kudoh et Matsumoto (2000) et du modèle *Yamcha* pour l'anglais, proposent un analyseur basé sur SVM et entraîné sur le corpus ATB (sect. 2.5.2), atteignant un taux de 95,49% d'étiquetage correct. Cet analyseur effectue également la tokenisation et les syntagmes de base (*chunks*). Par la suite, l'équipe de Mona Diab a amélioré cet étiqueteur sous le nom d'AMIRA 2.06 (Diab, 2010), bien que cette version ne soit pas disponible. Cette mise à jour affiche de meilleures performances, notamment une segmentation atteignant 99,2% de précision et un taux d'étiquetage correct supérieur à 96%. La séquence d'entrée est produite par un découpeur en syntagmes de base. Les étiquettes sont celles de ATB.

Algorithm 27 : Arabic SVM (Diab, Hacioglu et Jurafsky, 2004)

```
Function Segment(S) :
    foreach mot ∈ S do
        Extraire affixes, clitiques potentiels ;
        Appliquer un classificateur SVM binaire pour détecter les
         frontières de clitiques ;
        Segmenter le mot en radical et clitiques ;
        return Segment
Function Tag(S) :
    foreach Segment do
        Récupérer les caractéristiques des Segments voisins ;
        Ajouter la position du Segment traité ;
        Appliquer l'algorithme 28 pour prédire l'étiquette ;
return Séquence d'étiquettes
```

L'algorithme calcule les étiquettes elles-mêmes (les x_i sont les caractéristiques calculées, les y_i sont les étiquettes) :

Algorithm 28 : SVM Multiclasse pour l'étiquetage POS

Input : $\{(x_i, y_i)\}_{i=1}^{n}$ avec $x_i \in \mathbb{R}^d$, $y_i \in \{1, \ldots, K\}$
Output : $\hat{y} = \arg\max_k \; w_k^\top x$
Data : Initialiser poids $w_1, w_2, \ldots, w_K$;
Initialiser paramètre de régularisation $C > 0$;
foreach (x_i, y_i) **do**
  foreach $k \neq y_i$ **do**
    Calculer la marge : $m_{ik} = w_{y_i}^\top x_i - w_k^\top x_i$;
    if $m_{ik} < 1$ **then**
      Mettre à jour les poids w_{y_i} et w_k pour réduire la perte ;
Optimiser

$$\min_{\{w_k\}} \; \frac{1}{2} \sum_{k=1}^{K} \|w_k\|^2 + C \sum_{i=1}^{n} \xi_i$$

foreach x_i **do**
  foreach k **do**
    Calculer $s_k = w_k^\top x$;
    $\hat{y} = \arg\max_k \; s_k$;
return la séquence des $\hat{y}_i$;

Un post-traitement permet de générer des étiquettes syntaxiques selon le modèle « inside - outside - beginning » et de contrôler les étiquettes précédemment produites.

5.2.3 Salma

Sawalha, E. Atwell et Abushariah (2013) proposent SALMA, un étiqueteur open-source à base de règles, avec des ressources lexicales (dictionnaires d'affixes, de racines, de schèmes et de mots irréguliers avec des entités nommées) et un jeu d'étiquette propre avec 22 catégories morphosyntaxiques pour chaque morphème d'un mot. Comme le montre la figure 5.2, il constitue la dernière étape d'un pipeline partant de la tokenisation jusqu'à la vocalisation, en passant par la lemmatisation et le stemming. 71,21% des morphèmes de texte du Contemporary Arabic Corpus (sect. 2.2.5) ont été correctement étiquetés.

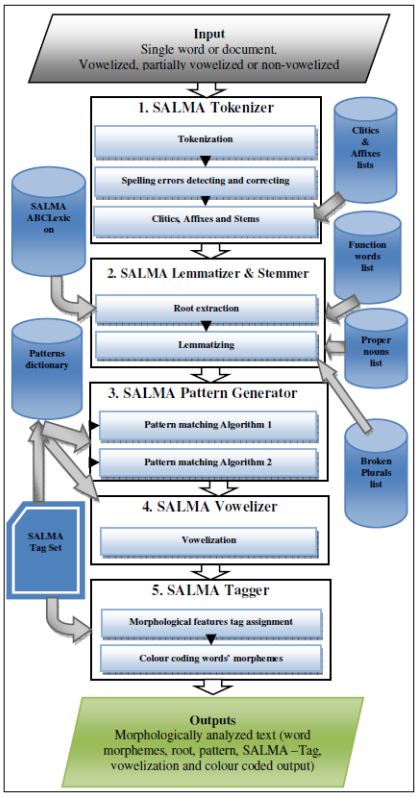


Fig. 5.2 : Algorithme Salma Pos (Sawalha, E. Atwell et Abushariah, 2013)

L'algorithme commence par effectuer une segmentation en cinq parties (proclitique, préfixe, radical, suffixe, enclitique) ; à l'aide d'un lexique, ces éléments sont étiquetés puis les étiquettes sont désambiguïsées ou corrigées grâce à des règles linguistiques sur les mots voisins du mot traité. Cette approche est lourde et nécessite beaucoup de traitements ad-hoc.

5.2.4 TreeTagger

TreeTagger est un étiqueteur développé par Schmid (1994) à l'Université de Stuttgart, produisant, réputé à l'époque pour sa robustesse, sa rapidité, et sa capacité à être adapté à différentes langues via un entraînement supervisé utilisant un arbre de décision. Il a été adapté à l'arabe par Ghoul (2013).

La méthode consiste à produire une première annotation avec ASVM, un ancêtre des outils Amira (sect. 4.4.1) entraîné sur PATB (sect. 2.5.2), puis de corriger à la main le résultat. Le corpus corrigé sert à entraîner des arbres de décision qui prennent en compte le contexte (les deux mots voisins à gauche et à droite) pour

prédire le tag à partir du contexte. Selon les auteurs, la précision atteinte était satisfaisante sur un petit corpus. Quelques années plus tard, Zeroual et Lakhouaja (2016) proposent une adaptation semblable, dans le cadre du corpus OSIAN (sect. 2.4.3).

5.2.5 MADAMIRA POS

Nous avons déjà vu le segmenteur de Madamira (section 4.4.1) ; cette bibliothèque dispose également d'un étiqueteur basé sur un modèle probabiliste de Maximum Entropy (Ratnaparkhi, 1996) (alg. 29), entraîné sur le corpus PATB. La performance est évaluée à travers un test à l'aveugle sur un ensemble de données, donnant 95,9% d'étiquettes correctes.

Algorithm 29 : Madamira POS Tagging

foreach *mot* $w_i \in S$ **do**
  Segmenter w_i en clitiques et radical : $w_i \rightarrow (c_1, c_2, \ldots, r, \ldots, c_k)$;
  Générer toutes les analyses morphologiques possibles : $A_i = \{a_{i1}, a_{i2}, \ldots, a_{im}\}$;
  foreach *analyse* $a_{ij} \in A_i$ **do**
    Extraire les caractéristiques linguistiques du contexte : $F_{ij} = \text{Features}(a_{ij}, \text{Contexte})$;
    Calculer la probabilité : $P(a_{ij} \mid F_{ij})$ via un classificateur supervisé (MaxEnt) ;
  Sélectionner l'analyse optimale : $a_i^* = \arg\max_{a_{ij} \in A_i} P(a_{ij} \mid F_{ij})$;
  Extraire le tag POS : $t_i = \text{POS}(a_i^*)$;
return $T = (t_1, t_2, \ldots, t_n)$

5.2.6 Farasa POS tagger

Abdelali, K. Darwish et al. (2016) ont introduit dans Farasa (sect. 4.4.2), système concurrent de Madamira mais focalisé sur la rapidité d'exécution, un étiqueteur qui repose sur un classificateur supervisé basé sur SVM multi-classe. Le POS tagger fonctionne comme suit :

1. Segmentation de chaque mot en clitiques et radical.
2. Extraction de caractéristiques du radical
   - morphologiques (affixes, racine) ;

- lexicales selon des dictionnaires d'entités nommées et de mots fréquents ;
- contextuelles : caractéristiques des mots voisins, ponctuation.

3. Classification supervisée pour attribuer une étiquette morphosyntaxique.
4. Postraitement à base de règles.

5.2.7 Etiqueteur par colonie d'abeilles

Alhasan et Al-Taani (2018) ont proposé un étiqueteur basé sur l'algorithme d'optimisation par colonie d'abeilles (Amador et Castillo, 2017), avec un ensemble de ressources linguistiques (dictionnaires). Une implémentation MATLAB générique de cet algorithme est disponible sous licence MIT[1], donc open-source.

L'étiquetage comprend trois phases :

1. Le texte est découpé en séquences de mots.
2. Le problème est modélisé comme un graphe, où chaque nœud représente une étiquette et chaque arête une transition possible entre les étiquettes.
3. Les abeilles explorent les chemins dans ce graphe pour trouver la séquence d'étiquettes optimale.

   → Une fonction objectif est utilisée pour évaluer les chemins.

Cette approche a été évaluée à l'aide du corpus KALIMAT. Les résultats expérimentaux ont montré que l'approche proposée atteignait 98,2% d'exactitude, comparée à une méthode basée sur des règles donnant 94,6%, un HMM donnant 97,4% et une méthode hybride donnant 98%. De plus, elle a identifié toutes les balises présentes dans le corpus, alors que les autres méthodes mentionnées n'ont pu identifier que trois balises.

5.2.8 LSTM POS Tagger

Alharbi et al. (2019) proposent un étiqueteur à base de biLSTM (sect. 4.6.1), mais pour l'arabe dialectal des pays du Golfe. L'apprentissage comprend trois principales étapes :

1. un prétraitement pour encoder les mots par embedding et les étiquettes par *One-hot encoding* ;

1. BCO est disponible sur https://github.com/victorkich/Bee-Colony-Algorithm.

2. les séquences d'embeddings sont transmis à une couche (bi)LSTM, puis à une couche linéaire sur laquelle un *softmax* est appliqué ;

3. Les poids sont optimisés avec Adam ou Stochastic Gradient Descent, à partir d'une fonction de perte définie par la *cross-entropy* entre la sortie voulue et le résultat du softmax.

L'exactitude sur le corpus PADT (sect. 2.5.3) a atteint entre 94 et 95%.

5.2.9 CAMeL POS tagger

CAMeL Tools[1] (Badr et al., 2022) est une suite open-source en Python développée par le CAMeL Lab à l'Université NYU d'Abu Dhabi ; elle continue d'évoluer. Elle regroupe plusieurs modules pour le traitement automatique de l'arabe, notamment : morphologie (analyse, désambiguïsation), POS tagging, diacritisation, lemmatization, reconnaissance d'entités nommées, identification de dialectes, traduction glossée. En sortie, on a un arbre JSON tel que :

```
{   "token": "  ",
    "POS": "verb",
    "lemma": "  ",
    "features":
    {   "person": "3",
        "number": "singular",
        "gender": "masculine"  }   }
```

L'étiqueteur est fondé sur des modèles probabilistes et neuronaux (incluant des BERT préentraînés). Il fonctionne sur des bases de données du moteur morphologique Almorgeana de Madamira (sect. 4.4.1). Il commence par une analyse morphologique à partir des bases de données d'Almorgeana pour produire toutes les segmentations valides, puis désambiguïse la séquence d'étiquettes à travers plusieurs modèles possibles : SVM, CFR ou un modèle neuronal, qui associent à chaque séquence un score.

Ils ont comparé les performances à celles de Madamira, pour constater des performances presque identiques.

5.2.10 Stanford POS Tagger

Le Stanford NLP Group a proposé un étiqueteur[2] générique (C. Manning et al., 2014), avec des adaptations à différentes langues, et de nombreuses versions. Pour

1. CAMeL Tools est disponible sur https://github.com/CAMeL-Lab/camel_tools.
2. Stanford NLP est disponible sur https://nlp.stanford.edu/software/tagger.shtml.

l'arabe, la principale adaptation est introduite avec la version Universal Dependencies v2, publiée en 2019, avec des modifications jusqu'en 2021.

L'étiqueteur utilise un modèle log-linéaire, un modèle de Markov à entropie maximale (une version avec CRF, sect. 4.4.3, est aussi disponible), entraîné, pour l'arabe, avec le corpus PATB (sect. 2.5.2). Pour chaque token, on extrait ses caractéristiques par segmentation, puis on calcule la probabilité conditionnelle de chaque étiquette t_i en fonction de l'étiquette précédente, du mot w_i et de ses caractéristiques (les segments) :

$$P(t_i \mid t_{i-1}, w_i, \text{segments}(w_i)) \tag{5.1}$$

Ensuite on applique soit le classificateur à entropie maximale, soit le champ aléatoire conditionnel. Enfin, on applique un algorithme de Viterbi pour déterminer la séquence d'étiquettes optimale, selon l'équation 5.2.

$$T^* = \arg\max_T \prod_{i=1}^{n} P(t_i \mid t_{i-1}, w_i, \text{segments}(w_i)) \tag{5.2}$$

Cet étiqueteur a été évalué sur des données provenant d'agences de presse, atteignant une exactitude de 96,5%. La performance tombe à 49% pour une évaluation faite sur un corpus de tweets, selon Albogamy et Ramsay (2016).

5.2.11 ALP POS tagger

Freihat, Bella, Abbas et al. (2023) ont proposé un pipeline open-source de traitement de l'arabe, Arabic Language Pipeline[1], qui inclut un étiqueteur effectuant en même temps la segmentation et la reconnaissance d'entités nommées sans prétraitement ni normalisation préalable, ce qui diminue l'ambiguïté selon Freihat, Bella, Mubarak et al. (2018).

L'algorithme est assez semblable à celui de Stanford : tokenisation et segmentation sont faites pour extraire des caractéristiques (affixes, type de token - alphabétique, numérique, etc., éventuellement racine et schème) ; des dictionnaires sont utilisés ; les caractéristiques des mots voisins dans la fenêtre de contexte sont ajoutées. Le tout est utilisé pour calculer la probabilité conditionnelle d'une étiquette t_i en fonction du mot w_i :

$$P(t_i \mid w_i) = \frac{1}{Z(w_i)} \exp\left(\sum_{j=1}^{n} \sum_{k} \lambda_k f_k(y_{j-1}, y_j, w_i, j)\right) \tag{5.3}$$

1. ALP est disponible sur http://www.arabicnlp.pro/alp/.

où Z représente le facteur de normalisation, et λ_k sont les pondérations apprises des fonctions de caractéristiques f_k.

Ensuite on applique l'algorithme de Viterbi pour optimiser les étiquettes :

$$\hat{t_i} = \arg\max_{t_i} P(t_i \mid w_i) \tag{5.4}$$

Un post-traitement est nécessaire pour corriger les séquences invalides, sans doute avec des règles.

Cet étiqueteur a été entraîné sur un corpus de deux millions de mots constitué par les auteurs à partir de divers genres (journaux, romans, pages web de conseils médicaux et réseaux sociaux). Les résultats ont montré une différence significative par rapport aux étiqueteurs entraînés sur un PATB ou un sous-ensemble. L'évaluation a été faite sur le corpus Aljazeera et sur le portail de conseil médical Altibbi. Ils ont utilisé 9990 tokens et ont effectué une évaluation et une validation manuelles des résultats. Les résultats ont révélé plusieurs types d'erreurs, grossières (mots segmentés correctement mais étiquetés incorrectement) et fines (segmentation et étiquette correctes mais sous-catégorie incorrecte). L'exactitude grossière atteignait 98,7%, tandis que l'exactitude fine atteignait 97,9%.

5.2.12 Alma

Jarrar, Akra et Hammouda (2024) ont proposé Alma[1], un étiqueteur et lemmatiseur open-source pour l'arabe, offrant à la fois une grande rapidité et une précision élevée. Alma repose sur un dictionnaire de solutions morphologiques classées par fréquence, développé sur la très grande base de données lexicographique Qabas. L'étiqueteur utilise un ensemble de 40 étiquettes non dérivées de l'anglais.

Dans un premier temps, les mots sont segmentés, puis pour chaque élément, on interroge la base Qabas, on récupère la solution la plus fréquente. Sinon, on marque l'élément comme inconnu.

L'évaluation d'Alma sur un ensemble de 339 710 mots du corpus PATB donne un score F1 de 88%, et sur 34.000 mots du corpus Salma (34k tokens), un score F1 de 90%. Alma a surpassé ces ensembles Farasa et Madamira POS tagger en termes de rapidité et d'exactitude.

1. Alma est disponible sur https://sina.birzeit.edu/alma.

5.3 Conclusion

Des étiqueteurs comme APT, SVM POS tagger ou Salma ont surtout un intérêt historique. Le LSTM POS tagger est intéressant en soi, mais il ne concerne pas l'arabe standard moderne. Camel et Madamira POS tagger sont pratiquement identiques en performance. Stanford est une référence partiellement indépendante de la langue, alors qu'Alma est extrêmement dépendant de ressources linguistiques sans lesquelles il ne peut conclure. De surcroît, Stanford est le seul dont les résultats sur Kalimat sont disponibles. L'étiqueteur par colonie d'abeille semble très intéressant, d'autant plus qu'il a été testé sur Kalimat, mais nous ne disposons ni du code ni du résultat.

Pour la suite de notre travail, nous partirons donc de Stanford, dont les résultats sur Kalimat sont supérieurs à 95% (certes moins bons que l'étiqueteur par colonie d'abeilles, mais pas de beaucoup).

Troisième partie

Système et réalisations

Chapitre 6

Corpus de référence et évaluation des segmenteurs

Sommaire

6.1 Introduction 158
6.2 Travail sur la normalisation 158
6.3 Corpus de référence initial 161
6.3.1 Kalimat 161
6.3.2 Prétraitement catégoriel 165
6.3.3 Comparaison catégories Stanford / Al-Khalil 169
6.3.4 Segmentation automatique 174
6.4 Production du corpus de référence 177
6.4.1 Analyse catégorielle 178
6.4.2 Segmentation manuelle 181
6.4.3 Description de la segmentation 183
6.4.4 Description du gold standard 186
6.5 Évaluation des segmenteurs 186
6.5.1 Analyse des résultats 188
6.6 Conclusion 194

6.1 Introduction

L'un des problèmes fondamentaux posés dans notre problématique est le faible nombre de corpus de référence libres utilisables pour la recherche et l'expérimentation en linguistique informatique (chapitre 2). Dans ce chapitre, après avoir présenté la manière dont nous devons convertir nos données pour travailler sur des données normalisées, nous allons introduire notre propre corpus de référence manuellement étiqueté et segmenté (Awdeh, Abdallah et al., 2021a), ainsi que la méthodologie que nous avons utilisée pour le construire. Puis nous montrerons son utilisation pour l'évaluation des segmenteurs vus aux chapitres 3 et 4, pour déterminer ceux qui peuvent nous être les plus utiles, ainsi que pour l'évaluation des parties du discours (POS).

6.2 Travail sur la normalisation

Tout d'abord, il faut prendre en compte le fait que nous traitons ici de texte normalisé (sans diacritiques) et que, par conséquent, l'inventaire des clitiques, en particulier, est modifié. Le tableau 1.6 des clitiques présenté dans le chapitre sur la problématique aboutit au tableau 6.1 après normalisation, avec de nombreuses confusions. Pour les enclitiques, la confusion entre /ka/ et /ki/ n'influence pas la segmentation. En revanche, la fusion, par exemple, des proclitiques /li/ de N0, /l/ de N – version contextuelle de l'article – et /la/ de V en /l/) peut changer la segmentation ; par conséquent, pour les proclitiques, on s'appuiera sur le tableau reformulé 6.2 plutôt que sur le tableau théorique initial.

Le proclitique ل se retrouve en trois endroits du tableau, dans des positions différentes. Sachant que pour le verbe, les positions médianes et finales pour les proclitiques sont fusionnées, on peut donc mettre ل en position médiane afin de l'unifier avec le proclitique médian ل du nom, qui présente une vocalisation différente mais est devenu identique par normalisation (la position du س reste indifférente). Enfin, ل en position finale pour le nom, seconde forme de l'article, est forcément précédé d'un proclitique dans l'une des positions antérieures. Les segmenteurs remplacent systématiquement ce ل par ال, ce que nous suivrons également, de manière à avoir un seul ل en position médiane. Pour pouvoir effectuer ce remplacement, il faudra trouver la catégorie du mot.

Tab. 6.1 : Clitiques normalisés

	Proclitiques			**Enclitiques**
	Procl.debut	**Procl.**	**Procl.fin**	
V		ل ما	س	ني
N		ك	ال	
NS/0		ب ل		ي
P		ما		
VNP	و ف	أ لا		نا ك كما كم ه ها هما هم هن

Ceci permet de définir le tableau simplifié 6.2 des proclitiques normalisés, avec 2 proclitiques initiaux, 6 proclitiques médians, dont أ peut être écrit ا selon les types de normalisation (d'où 7 formes possibles), et 2 proclitiques finaux. Le tableau 6.3 des enclitiques normalisés lui reprend simplement les données initiales.

Tab. 6.2 : Proclitiques après normalisation

	Procl.deb	**Procl.**	**Procl.fin**
V			س
VP		ما	
N		ك	ال
NS/0		ب	
VNP	و ف	أ ل لا	

Tab. 6.3 : Enclitiques après normalisation

	Enclitiques
V NS/0	ني ي
VNP	نا ك كما كم ه ها هما هم هن

Le tableau 6.4 reprend pratiquement le tableau 1.7 du chapitre de la problématique ; les quatre préfixes sont déjà normalisés dans la mesure où la voyelle qu'ils portent dépend du radical, comme on l'a vu, et le seul élément notable c'est le possible remplacement de أ par ا, ce qui fait qu'en fait nous avons cinq formes. Trois préfixes ne sont possibles qu'avec VI, et un préfixe est possible avec V0 ou VI.

Tab. 6.4 : Préfixes flexionnels

	Préfixe
VI	ن ت ي
V0/VI	أ

Il y a une confusion possible du préfixe أ avec le proclitique médian de même forme, d'autant plus qu'on peut les trouver tous les deux avec les verbes.

Le tableau des suffixes 6.5 reprend le tableau 1.8, que nous avions déjà dû normaliser pour simplifier l'exposé dans la problématique.

Il y a des confusions possibles entre les enclitiques et les suffixes, par exemple نا, qui peuvent être les deux, ي également, ainsi que ى pour les normalisations qui les confondent ; une séquence suffixe ان + enclitique ي peut se confondre avec le suffixe ا + enclitique ني.

Bien entendu, la plus grande quantité de confusion réside dans le radical, car étant lexical, il est très peu limité dans sa composition en consonnes et en séquences, contrairement aux clitiques et aux affixes.

Tab. 6.5 : Suffixes flexionnels

Suffixe	**Catégories**
ا	N, V
ان	N, V
ين	N, V
و	N, V
ون	N, VI
ات	NA
ة	NA
تين	NA
ى	NA
ن	V
وا	VI, V0
ي	VI, V0
ت	VA
نا	VA
تا	VA
تما	VA
تم	VA
تن	VA

6.3 Corpus de référence initial

Pour constituer notre corpus de référence initial, nous sommes partis du corpus *Kalimat* (voir la section 2.4.1), car c'est le plus grand corpus libre en arabe standard moderne qui soit à la fois étiqueté pour les parties du discours et segmenté. L'étiquetage et la segmentation ont été générés automatiquement, donc notre objectif est de parvenir à en faire un corpus de référence contrôlé.

Dans la suite de cette section, nous allons détailler les caractéristiques de ce corpus, ainsi que les problèmes qu'il pose, puis la méthode que nous avons utilisée, dans un premier temps, pour produire automatiquement une segmentation que nous avons, dans un deuxième temps, corrigée à la main. Nous présentons à la fin de cette section le corpus résultant et ses caractéristiques.

6.3.1 Kalimat

Kalimat contient 20.291 documents, qui se répartissent en six catégories : culture, économie, actualités locales, actualités internationales, religion et sport, pris dans le quotidien omanais Al-Watan.

Le corpus Kalimat, une fois téléchargé, ne contient que 10 578 624 tokens, dont

un peu moins d'une dizaine de millions de mots, au lieu des dix-huit millions de mots indiqués dans l'article de El-Haj et Koulali (2013) ; nous n'avons pu savoir pourquoi. Cette divergence quantitative est notable, bien que le nombre total d'articles soit exact, ainsi que la répartition de ces articles dans les catégories : la structure documentaire est bien conservée. L'écart porte donc uniquement sur le volume ; cette correction est essentielle pour toute étude quantitative ou linguistique fondée sur Kalimat. La répartition exacte est indiquée dans le tableau 6.6 (où ne sont comptabilisés que les mots, en enlevant les ponctuations et les nombres – mots entièrement numériques), où l'on constate que certaines catégories ont été sous-estimées (surtout Religion) et d'autres surestimées (surtout Sports et Économie). La façon de comptabiliser les tokens ne peut à elle seule expliquer cet écart.

Tab. 6.6 : Collection de données

Catégories	Articles	Mots selon les auteurs	Nombre réel de mots
Culture	2 782	1 359 210	1 403 974
Economy	3 468	3 122 565	1 468 270
International News	2 035	855 945	862 328
Local news	3 596	1 460 462	1 560 833
Religion	3 860	1 555 635	3 117 270
Sports	4 550	9 813 366	1 436 754
Total	20 291	18 167 183	9 849 429

Le tableau 6.7 présente les statistiques globales du corpus, et le tableau 6.10 présente la liste des dix mots les plus fréquents. Les mots en arabe représentent plus de 93% des occurrences ($\sim$ 95% si l'on y ajoute les nombres) et $\sim$ 97% des tokens uniques. Les hapax représentent 39% des mots uniques, un taux un peu bas qui montre une certaine redondance, plutôt caractéristique des textes retranscrits de l'oral (textes journalistiques).

Tab. 6.7 : Statistiques de Kalimat

Tab. 6.8 : Occurrences

Mots	9 849 429
Nombres	159 251
Ponctuations	569 944
Total tokens	10 578 624

Tab. 6.9 : Tokens uniques

Mots uniques	261 372
dont hapax (mots à une seule occurrence)	97 313
Tokens uniques	270 537

Tab. 6.10 : Fréquence des mots les plus courants

الله	59517
عليه	90 898
كان	24402
قال	20560
عام	14108
صلى	12711
وسلم	12521
العام	12374
محمد	11936
العمل	11817
Total	117 023

La partie qui nous intéresse ici contient deux ensembles de données : les fichiers HTML qui contiennent les segmentations et ceux qui contiennent les parties du discours étiquetées. Comme nous l'avons indiqué dans la section 2.4.1, les segmentations sont produites par l'analyseur morphologique Al-Khalil (section 3.2.17) puis sélectionnées par un algorithme de Viterbi (section 4.3.2), et l'étiquetage a été effectué par le Stanford POS Tagger (section 5.2.10).

La figure 6.1 illustre la structure des fichiers de segmentation :



Fig. 6.1 : Structure de Kalimat corpus

Prenons, par exemple, la segmentation suivante :

تنطلق: تُنْطَلَقْ: ت: تاء الغائبة: تنطلق: فعل مضارع مبني للمجهول: تُنْفَعَلْ:طلق:#:

Les éléments de cette segmentation sont, de droite à gauche :

1. Mot : le mot à analyser (تنطلق) ;
2. Mot voyellé : le mot avec restitution des diacritiques (ici تُنْطَلَقْ) ;
3. Préfixe : préfixes et proclitiques concaténés ou séparés (ici ت) ;
4. Minimal : le mot minimal (تنطلق) ;
5. Catégorie : la catégorie grammaticale et sémantique du mot (ex. nom de matière, nom adjectif...) ; (ici فعل مضارع مبني للمجهول, verbe inaccompli passif) ;
6. Schème : le schème du radical (incluant les affixes mais non les clitiques) (ici تُنْفَعَلْ) ;
7. Racine : la racine du mot (ici طلق) ;
8. Suffixe : suffixes et enclitiques concaténés ou séparés (ici # indique qu'il n'y a pas de suffixe).

On observe que le radical n'est pas présent dans le résultat, seulement le mot minimal incluant les affixes.

La segmentation produite par Al-Khalil pose de nombreux problèmes ; ainsi, 39% des catégories sont non générées, 15% des segmentations ne sont pas faites (exemple figure 6.2), le découpage en clitiques et affixes est inconsistant (en effet, parfois le « préfixe » est la concaténation des proclitiques et du préfixe, parfois ils sont séparés, et de même pour le « suffixe », concaténé avec les enclitiques ou non), sans compter les erreurs de segmentation, assez nombreuses à première vue.

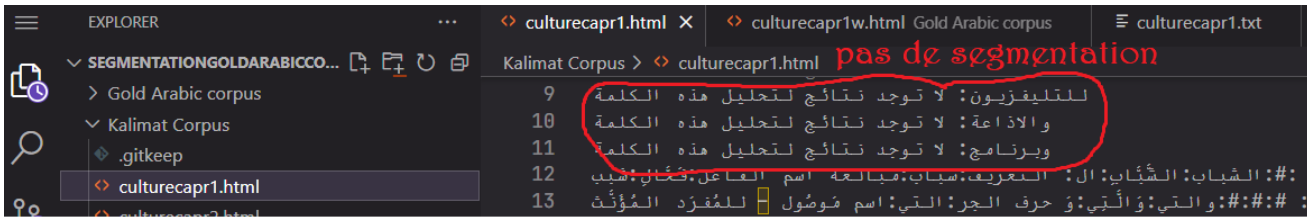


Fig. 6.2 : Absence de segmentation dans Kalimat

Les fichiers d'étiquettes POS ont la forme suivante :

تنطلق/ VBP اليوم/ DTNN الدورة/ DTNN البرامجية/ DTJJ الجديدة/ DTJJ للتليفزيون/ NNS والاذاعة/ NN وبرنامج/ NN الشباب/DTNN والتي/ NNS تستمر/ VBP طوال/ NN اشهر/ NN ابريل/ NN ومايو/ NNP ويونيو/ NNP وتحمل/ NNP في/ IN طياتها/ NN العديد/ DTNN من/ IN البرامج/ DTNN الجديدة/ DTJJ والفقرات/ NNS الشيقة/ DTJJ
الدكتور/ DTNN تيسير/ NN اخصائي/ NN الفنون/ DTNN التشكيليةDTJJ/ والتي/ NNS استمرت/ VBD

6.3.2 Prétraitement catégoriel

La première étape par laquelle nous devons passer, c'est la production, à partir de ce corpus, d'un ensemble de données standardisé en vue d'une segmentation contrôlable, et tout d'abord, la normalisation des données catégorielles ici produites par Al-Khalil et Stanford. Rappelons que notre but n'est pas d'être le plus détaillé possible concernant l'étiquetage des parties du discours ou les caractéristiques morphologiques, mais de conserver des informations morphologiques et catégorielles qui soient pertinentes pour la segmentation, que ce soit pour les prédire ou pour les utiliser.

Il découle de notre problématique (voir page 37) que nous avons besoin des catégories suivantes pour assurer une meilleure segmentation :

- N (noms)
  - NS (noms substantifs, y compris infinitifs)
  - NA (noms adjectifs, y compris participes)
- V (verbes)
  - VO (impératif)
  - VI (inaccompli)
  - VA (accompli)
- P (particules)
- IS (mots isolés, adverbes)

Ces catégories constituent notre schéma catégoriel, notre jeu d'étiquettes, qui est directement inspiré par les contraintes de la langue. Nous avons écarté tout ce qui concerne les affixes de dérivation, qui sont difficilement utilisables pour le type de segmentation que nous cherchons à mettre en œuvre (sans décomposition du radical). La catégorie IS n'est pas distinguable dans les étiquettes fournies par Al-Khalil et Stanford, elle se confond avec P.

Dans les tableaux suivants, nous appelons POS1 les catégories du premier niveau, POS2 les catégories du second niveau. Les deux procédures citées permettent d'introduire un troisième niveau, que nous appellerons POS3. Nous allons examiner de plus près les catégories produites par ces procédures.

Le tableau 6.11 montre ce qui peut être pertinent dans les catégories proposées par Al-Khalil, ainsi que les statistiques de chaque catégorie. L'aspect sémantique (nom de matière, nom d'agent, nom d'action) n'a aucune incidence directe sur la segmentation. Il peut y avoir une influence très indirecte (un nom de temps s'emploiera plus rarement avec des pronoms possessifs), mais les catégories sémantiques ont des définitions trop variables pour être utilisées.

L'aspect nominal ou verbal est crucial ; la distinction entre substantif et adjectif a une importance pour la gestion du suffixe féminin (ne fait pas partie du radical pour l'adjectif, en revanche, en fait partie pour le nom). L'aspect accompli / inaccompli gouverne la position des affixes pour le verbe (préfixes pour l'inaccompli, suffixes pour l'accompli). L'actif ou le passif ont une interaction avec les diacritiques et devraient donc permettre de les retrouver, ce qui peut améliorer la segmentation. Cependant, mais comme nos données ne portent pas de diacritiques, l'apport est faible ; l'impératif n'a pas (ou très peu) d'affixes, donc il est également pertinent. En revanche, l'infinitif est un nom (substantif) comme un autre, puisqu'on ne prend pas en compte les affixes de dérivation.

Tab. 6.11 : Conversion catégories Al-Khalil

Al-Khalil	**sens**	**nombre**	**POS1**	**2**	**3**
اسم الجلالة	nom divin	67 721	N	NS	
اسم آلة	nom d'outil	359 139	N	NS	
اسم زمان أو مكان	nom de temps/lieu	271 493	N	NS	
اسم شَرْط	nom de condition	267 316	N	NS	
اسم علم	nom propre	105 990	N	NS	
اسم فاعل	nom d'agent	440 140	N	NS	
اسم مفعول	nom d'objet	206 928	N	NS	
اسم تفضيل	nom de supériorité	50 157	N	NS	
اسم صَوت بِمعنى الفِعل	nom du son	188	N	NS	
مصدر مرة	nom d'action ponctuelle	324 820	N	NS	
مصدر أصلي	infinitif de base	906 978	N	NS	
مصدر صناعي	nom verbal artificiel	210 411	N	NS	
مصدر هيئة	nom verbal/manière	263 826	N	NS	
مبالغة اسم الفاعل	nom d'agent intensif	331 888	N	NS	
Total	substantifs	**3 806 995**	N	NS	
اسم جامد	nom adjectif	336 312	N	NA	
صفة مشبهة	adjectif verbal	121 152	N	NA	
نسبة	adjectif de relation	140 023	N	NA	
نسبة إلى اسم علم	adjectif dérivé	3 265	N	NA	
Total	adjectifs	**600 752**	N	NA	
Total	noms	**4 407 747**	N		
حرف جر	préposition	411 654	P		
حَرف جرّ أو ظَرف	préposition/adverbe	147 134	P		
حَرف جرّ-مصدري	préposition/adverbe	70 781	P		
ظَرف (للزَّمان والمَكان)	adverbe temps/lieu	41 045	P		
ظَرف زَمان	adverbe de temps	19 381	P		
ظَرف مَكان	adverbe lieu	35 943	P		

continué à la page suivante

Tab. 6.11 : Conversion catégories Al-Khalil

Al-Khalil	**sens**	**nombre**	**POS1**	**2**	**3**
اسم استِفهام – عن الحال	pronom interrog./état	4 031	P		
اسم استِفهام – للعاقِل	pronom interrogotif	15 116	P		
اسم إشارة – للمُفرَد المُذَكَّر	pron. démonst./si. masc.	61 460	P		
اسم إشارة – للمُفرَد المُؤَنَّث	pron. démonst./si. fém.	49 721	P		
اسم مَوصُول – لغير العاقِل	pron. relatif/chose	94 932	P		
اسم مَوصُول – للجمع المُذَكَّر	pron. relatif/pl. masc.	10 569	P		
اسم مَوصُول – للمُفرَد المُذَكَّر	pron. relatif/si. masc.	44 508	P		
اسم مَوصُول – للمُفرَد المُؤَنَّث	pron. relatif/si. fém.	68 266	P		
أداة استِثناء	particule d'exclusion	18 238	P		
أداة نَفي أو نهي أو حرف زائد	outil de négation	64 867	P		
ضَمِير الغائِب – للجمع المُذَكَّر	pro. pers. sujet/pl. masc.	13 629	P		
ضَمِير الغائِب – للجمع المُؤَنَّث	pro. pers. sujet/pl. fém.	386	P		
ضَمِير الغائِب – للمُفرَد المُذَكَّر	pro. pers. sujet/si. masc.	37 057	P		
ضَمِير الغائِب – للمُفرَد المُؤَنَّث	pro. pers. sujet/si. fém.	21 511	P		
جار ومجرور	préposition	58 088	P		
حَرف تَحقِيق وتَقرِيب	particule de certitude/proximité	44 308	P		
حَرف توكيد ناسِخ	particule d'emphase/annulation	118 783	P		
حَرف عَطف	conjonction de coordination	36 271	P		
حَرف نداء - للبعيد	particule d'appel	5 601	P		
حرف نصب وجرّ وعَطف وابتِداء	particule de subordinat./prépos.	18 552	P		
Total	particules	**1 511 832**	P		
فعل أمر	verbe impératif	1 410 178	V	VO	
فعل أمر مؤكد	verbe à l'impératif avec emphase	93 953	V	VO	
فعل ماض مبني للمجهول	verbe accompli passif	165 622	V	VA	VAP
فعل ماض مبني للمعلوم	verbe accompli actif	226 565	V	VA	VAA
فعل مضارع مبني للمجهول	verbe inaccompli passif	441 480	V	VI	VIP
فعل مضارع مبني للمعلوم	verbe inaccompli actif	124 982	V	VI	VIA
Total	verbes	**2 462 780**	V		
Unknown	inconnu	1 467 070	U		
Total final		**9 849 429**			

On retombe donc sur notre schéma catégoriel, auquel on peut ajouter en POS3 la distinction actif/passif.

Les étiquettes produites par Stanford portent également des informations pertinentes pour la segmentation, comme la catégorie morphosyntaxique (nom, verbe conjugué, particule). Certaines informations morphologiques secondaires (POS3) sont pertinentes : le défini est lié à la présence de l'article (clitique), le pluriel à la présence possible de suffixes spécifiques. D'autres informations morphosémantiques ne sont pas pertinentes, ou alors très indirectement. D'où la conversion

présentée, avec les statistiques de chaque catégorie, dans le tableau 6.12.

Tab. 6.12 : Conversion catégories Stanford

Stanford	**sens**	**nombre**	**POS1**	**2**	**3**
NOUN_QUANT	quantificateur (tout, etc.)	59 537	N	NS	
NN	nom	3 009 096	N	NS	
NNP	nom propre	1 274 290	N	NS	
NNS	nom pluriel	200 621	N	NS	P
DTNN	nom + article	1 386 383	N	NS	D
DTNNP	nom propre + article	127 463	N	NS	D
DTNNS	nom pluriel + article	196 426	N	NS	DP
DTNNPS	nom propre pluriel + article	126	N	NS	DP
CD	nom de nombre (cardinal)	159 251	N	NS	
Total	NS	**6 413 193**			
ADJ_NUM	adjectif numéral	47 772	N	NA	
JJ	adjectif	426 826	N	NA	
JJR	adjectif comparatif	13 940	N	NA	
DTJJ	adjectif + article	570 065	N	NA	D
DTJJR	adjectif comparatif + article	7 336	N	NA	D
VBG	gérondif ou participe présent	4 887	N	NA	
VN	participe passé	39 811	N	NA	
VBN	participe passé	41 436	N	NA	
Total	NA	**1 152 073**			
Total	N	**7 565 266**			
DT	déictique (2 fois l'article)	123 582	P		
IN	préposition ou conj. de subord.	955 500	P		
PRP	pronom personnel	34 974	P		
CC	conjonction de coordination	74 445	P		
RB	adverbe	24 462	P		
RP	particule	100 502	P		
UH	interjection	910	P		
WP	pronom interrogatif	156 956	P		
WRB	adverbe interrogatif	29 611	P		
PRP	pronom possessif	34 974	p		
PRP$	pronom possessif	3 320	p		
Total	P	**1 504 262**			
VBD	verbe accompli	487 860	V	VA	
VBP	verbe inacc. sg autre que 3è p.	448 873	V	VI	
VB	forme de base du verbe	2 419	V	VO	
Total	V	**939 152**			
PUNC	ponctuation	569 944	PUNC		
Total final		**10 578 624**			

Les prépositions sont étiquetées IN quand elles sont isolées, comme NN ou NNS quand elles ont des enclitiques (NNS quand l'enclitique est au pluriel), comme JJ quand elles n'ont que des proclitiques, ce qui n'est pas très cohérent.

À partir de ces catégories, nous déduisons notre schéma catégoriel ; de plus, en POS3, la catégorie du défini et la catégorie pluriel (pour les noms) ; la sous-catégorisation POS3, pour notre corpus, intégrera donc D et P (défini et pluriel) pour les noms, transmise par Stanford, et A et P (actif et passif) pour les verbes, transmise par Al-Khalil. Les tables de conversion, accessibles sur notre dépôt gitlab, ont la forme :

Tag (AlKhalil ∨ Stanford)	pos1	pos2	pos3

Dans le jeu de tags que nous avons défini, la catégorie POS3 diffère complètement dans les deux systèmes ; si elle pourra être utilisée par la suite, pour améliorer l'entraînement d'un modèle, elle ne peut donc pas servir à la comparaison des deux systèmes.

6.3.3 Comparaison catégories Stanford / Al-Khalil

Ce travail préliminaire accompli, nous pouvons faire différentes observations. Le nombre total de tokens diverge parce qu'Al-Khalil ne prend pas en compte la ponctuation et les nombres en chiffres. Stanford prend en compte la ponctuation, mais les nombres en chiffres sont confondus avec les nombres en lettres et traités comme des noms (ce qui peut se justifier car on observe qu'ils prennent parfois des clitiques). Al-Khalil ne parvient pas à étiqueter 15% des mots qu'il prend en compte.

Concernant la qualité des annotations, les auteurs de Kalimat indiquent que Stanford POS a un taux de réussite de 96,5%. Même si nous ne savons pas par rapport à quoi ce taux de réussite a été calculé, en l'absence d'un gold standard, nos observations confirment qu'Al-Khalil présente de nombreuses erreurs d'annotation, bien plus que Stanford ; mais les erreurs de Stanford sont suffisamment nombreuses pour qu'on ne puisse pas éliminer d'office Al-Khalil.

Si on regroupe les mots par catégories (POS1 et POS2), on obtient le tableau 6.13, qui montre qu'Al-Khalil détecte beaucoup plus de verbes que Stanford, la plupart à l'impératif (forme non fléchie du verbe), et beaucoup moins de noms. Bon nombre de ces impératifs sont probablement, en fait, des noms appartenant à une racine fournissant également un verbe. Pour le reste, la proportion accompli/inaccompli pour les verbes fléchis est très comparable ; les particules aussi.

Tab. 6.13 : Statistiques sur les POS

	Stanford	**Alkhalil**
N NS	6 413 193	3 806 995
N NA	1 152 073	600 752
N	7 565 266	4 407 747
P	1 504 262	1 511 832
V VA	487 860	392 187
V VI	448 873	566 462
V VO	–	1 504 131
V VB	2 419	–
V	939 152	2 462 780
PUNC	569 944	-
U	–	1 467 070
Total	**10 578 624**	**9 849 429**

Le tableau 6.14 présente les convergences et divergences entre Stanford et Al-Khalil. Nous avons commencé par supprimer les nombres et les ponctuations pour avoir le même nombre de mots dans les deux systèmes.

Tab. 6.14 : Comparaison entre Stanford et Alkhalil (tous les mots)

Alkhalil \ Stanford	N	NS	NA	V	VA	VI	P	Total
N	**4 108 241**	3 358 882	749 359	16 319	13 256	303	296 911	
NS	3 504 119	**2 908 975**	595 144	15 352	12 592	2 721	296 638	
NA	604 122	449 907	**154 215**	967	664	3 024	273	
V	1 674 934	1 547 321	127 613	**779 097**	357 318	420 104	15 928	
VA	228 331	213 339	14 992	162 505	**160 156**	2 336	1 395	
VI	210 820	180 734	30 086	355 744	24 884	**330 851**	24	
P	528 354	509 199	19 155	73 196	68 015	4 818	**910 790**	
VO	1 235 783	1 153 248	82 535	260 848	172 278	86 917	14 509	
U	1 253 737	997 791	255 946	70 540	4 9271	20 927	280 633	
Total	7 565 266	6 413 193	1 152 073	939 152	487 860	448 873	504 262	

On ne peut pas directement comparer les catégories des mots uniques, étant donné qu'ils peuvent appartenir à plusieurs catégories et que ce ne sont naturellement pas forcément les mêmes dans chacune. Par conséquent, nous comparerons dans le tableau 6.15 le nombre de (mot + catégorie) uniques, selon nos deux niveaux de catégorisation POS1 ou POS2. Le total des mots uniques par sous-catégorie est forcément supérieur au nombre de mots uniques dans la catégorie, puisque certains mots se retrouvent dans les deux sous-catégories. Nous avons mis VB de Stanford et V0 de Al-Khalil sur la même ligne, ce qui ne veut pas dire qu'ils sont comparables.

Ces chiffres confirment la plus grande prédominance des noms dans Stanford. On y observe aussi une sous-représentation dans Stanford des particules, alors qu'en nombre d'occurrences les résultats d'Al-Khalil et Stanford étaient très similaires. Stanford a sans doute une définition plus stricte, qui le conduit à considérer comme particules uniquement des éléments très fréquents. S'il manque des particules chez Stanford, elles se retrouvent probablement dans les noms, avec lesquelles elles ont plus de parenté morphologique (beaucoup de particules sont d'anciens noms). En revanche, s'il y a trop de particules (et peu fréquentes) chez Al-Khalil, ce sont probablement des noms mal catégorisés.

Tab. 6.15 : Mots uniques pour chaque POS

POS1	POS2	**Stanford**	**Alkhalil**
N		248 082	120 353
	NS	223 567	104 450
	NA	57 498	20 775
	Total	**281 065**	**125 225**
P		149	1 541
V		35 190	79 301
	VA	17 778	13 333
	VI	18 087	31 763
	VB	149	–
	V0	–	34 205
	Total	**36 014**	**79 301**
PUNC		8	-
U		-	69 577
Total		**283 429**	**261 372**

Le tableau 6.16 présente le nombre de mots uniques dont la catégorie POS1 varie, en fonction de chaque étiqueteur (nous n'avons pas pris en compte la possibilité de multiple ambiguïté). Stanford présente essentiellement des cas d'ambiguïté entre noms et verbes, les autres échanges étant mineurs ; le total de ces cas correspond à 4,94% des mots uniques au maximum (en enlevant d'éventuelles multiples ambiguïtés). En revanche, la situation est beaucoup moins claire pour Al-Khalil, qui présente des ambiguïtés un peu partout, pour un total de 18,4% au maximum. On voit donc bien ici que Stanford semble plus cohérent qu'Al-Khalil.

Tab. 6.16 : Ambiguïtés de POS1

	Stanford	**Alkhalil**
N, V	12 831	14 148
N, P	58	6 144
P, V	11	6 733
N, U	-	9 731
V, U	-	7 471
P, U	-	3 756
Total	**12 900**	**47 983**

Le tableau 6.17 présente les ambiguïtés intracatégorielles. Ici non plus nous n'avons pas calculé la possibilité d'une multiple ambiguïté. On y constate que la plus grande ambiguïté est entre substantifs et adjectifs ; pour Stanford, c'est la principale cause d'ambiguïté intracatégorielle ; Pour Al-Khalil, les ambiguïtés sont plus fortes que chez Stanford dans les autres catégories, et le nombre d'ambiguïtés à l'intérieur des noms est beaucoup plus faible. Il n'est pas sûr que ce soit un bon point pour Al-Khalil, car l'ambiguïté substantif/adjectif est beaucoup plus vraisemblable que l'ambiguité entre un verbe à l'accompli et un verbe à l'impératif. L'ambiguïté entre inaccompli et impératif est en revanche plus vraisemblable.

Tab. 6.17 : Ambiguïtés de POS2

	Stanford	**Alkhalil**
NA, NS	32 983	4 872
VA, VI	756	1 381
VA, V0	–	2 292
VI, V0	–	2 767
VA, VB	65	–
VI, VB	13	–

En conclusion de cette section, l'utilisation de notre jeu d'étiquettes donne des indices indirects nous permettant d'estimer que l'étiquetage par Stanford est plus cohérent et plus vraisemblable que l'étiquetage par Al-Khalil.

6.3.4 Segmentation automatique

Ayant intégré les indications des parties du discours provenant des étiqueteurs Al-Khalil et Stanford, il nous reste maintenant à préparer la segmentation de ce corpus.

Pour cela, nous disposerons de plusieurs segmentations automatiques :

- celle, intégrée dans le corpus initial, produite par le segmenteur Al-Khalil (section 3.2.17) ; il faudrait prévoir une standardisation du produit de Al-Khalil, car, comme nous l'avons vu, il ne donne pas le radical mais le mot minimal, contrairement aux autres analyseurs ;
- celle de mon propre pseudo-segmenteur se fondant sur la catégorie POS, d'après Stanford ;
- celle d'autres segmenteurs vus dans les chapitres sur la segmentation ; testés dans un premier temps sur le *gold standard*, pour sélectionner les meilleurs d'entre eux.

Mon pseudo-segmenteur[1] (algorithme 30) n'est pas un vrai segmenteur, car un segmenteur ne suppose pas connaître la catégorie du mot qu'il segmente ; c'est simplement la combinaison de deux procédures de segmentation disjointes, une pour les noms et une pour les verbes. L'objectif est de fournir une aide supplémentaire au contrôle de la segmentation. Il commence, comme presque tous les segmenteurs par règles, par une normalisation. Si la catégorie du mot n'est ni un nom ni un verbe (c'est donc une particule), on applique la fonction qui sépare les clitiques (une particule n'a pas d'affixes). Les seuls clitiques possibles pour une particule sont N/0, P et VNP. Si c'est un nom, on utilise les clitiques N, N/0 et VNP ; si c'est un verbe, les clitiques V et VNP. Pour les proclitiques, l'ordre qui doit être suivi est Procl.debut, puis Procl. puis Procl.fin.

La donnée principale est le tableau 6.18, qui permet de retrouver les segments correspondant à une catégorie grammaticale. Ici la catégorie "rien", désignant un radical vide, est remplacée par P, car c'est la catégorie du mot entier.

Si on ne connaît pas POS2, le tableau 6.19 est utilisé à la place. On remarquera que le segmenteur n'est plus capable de découper correctement {ة, تين, ى}, suffixes spécifiquement adjectivaux, ni de découper correctement les verbes. C'est encore pire si le segmenteur (ou le texte) a subi une normalisation radicale (fusion de أ avec ا et de ي avec ى). En fait s'il n'a que POS1, il a seulement la capacité à détecter correctement les clitiques.

1. Le code d'AwdehSegmenter en java est accessible sur https://gitlab.com/hussein.awdeh/segmentationgoldarabiccorpus/-/blob/main/Tools/AwdehSegmenter.java

Tab. 6.18 : Distribution des segments par catégorie

vec / cat	Proc1	Proc2	Proc3	Pref	Suffixe	Enclitique
N		ك	ال			
NA					ى ,تين ,ة ,ات	
NS/P		ب				ي
N/V					و ,ين ,ان ,ا	
N/VI					ون	
V			س		ن	ني
VA					,تما ,تا ,نا ,ت	
					تن ,تم	
VI				ي ,ت ,ن		
VI/V0				أ	ي ,وا	
V/P		ما				
VNP	ف ,و	لا,ل ,أ				ه ,كم ,كما ,ك,نا
						هن ,هم ,هما ,ها

Tab. 6.19 : Distribution des segments par catégorie sans POS2

vec / cat	Proc1	Proc2	Proc3	Pref	Suffixe	Enclitique
N		ك	ال		ى ,تين ,ة ,ات	
N/P		ب				ي
N/V					ون ,و ,ين ,ان ,ا	
V			س	ي ,ت ,ن ,أ	ن ,وا ,ي ,ت ,نا ,تا	ني
					تن ,تم ,تما	
V/P		ما				
VNP	ف ,و	لا,ل ,أ				ه ,كم ,كما ,ك,نا
						هن ,هم ,هما ,ها

Ce programme peut accepter plusieurs types de normalisation :

1. simple, avec suppression des diacritiques vocaliques ;
2. avancée, avec suppression du signe de redoublement (chadda) et réécriture de quelques lettres (ex. remplacer ئ par la séquence ىء) ;
3. radicale, avec fusion de lettres .(أ – ا, ي – ى, ه – ة) ;
4. adhoc, personnalisée au travers d'un fichier de correspondances.

Algorithm 30 : Segmentation dirigée par la catégorie

Input : mot w, catégorie POS1 POS2 POS3, type de normalisation
Output : $\vec{V}$ = [Pro1, Pro2, Pro3, Pref, Rad, Suf, Encl] ; $\vec{V}$ = ""
Data : T (table des segments par catégorie)

```
Function Segment(w) :
    Normalisation
    if longueur(w) ≤ 2 then V.Stem ← w ; return V ;
    Détection de Procl.1
        if début(w) ∈ Procl.1 then
            V.Procl1 ← début(w) ; w ← w-début(w)
        if longueur(w) ≤ 2 then V.Stem ← w ; return V ;
        if C = verb then return Verbproc(w) ;
        else return Nomproc(w) ;
    Verbproc
        if longueur(w) ≥ 4 then
            if début(w) ∈ Procl.VNP then
                V.Procl ← début(w) ; w ← w-début(w)
            else if début(w) ∈ Procl.finV then
                V.Procl ← début(w) ; w ← w-début(w)
            else if fin(w) ∈ EncliticsV then
                V.Enclitics ← fin(w) ; w ← w-fin(w)
            else if fin(w) ∈ Suff.V then
                V.Suffix ← fin(w) ; w ← w-fin(w)
        else
            V.Stem getsw ; return V ;
        if longueur(w) ≥ 4 then return Verbproc(w) ;
    Nomproc
        if longueur(w) ≥ 3 then
            if début(w) ∈ Procl. then
                V.Procl ← début(w) ; w ← w-début(w)
            else if début(w) ∈ Procl.finN then
                V.Procl ← début(w) ; w ← w-début(w)
            else if fin(w) ∈ EncliticsN then
                V.Enclitics ← fin(w) ; w ← w-fin(w)
            else if fin(w) ∈ Suff.N then
                V.Suffix ← fin(w) ; w ← w-fin(w)
        else V.Stem getsw ; return V ;
        if longueur(w) ≥ 4 then return Nomproc(w) ;
    return V ;
```

6.4 Production du corpus de référence

Pour constituer le *gold standard*, destiné en particulier à sélectionner les meilleurs segmenteurs, j'ai sélectionné dans un unique fichier 51 articles de la catégorie culture (catégorie a priori la moins spécialisée) de 30 438 tokens. Les articles sont concaténés dans l'ordre initial, indiqué dans le nom du fichier correspondant. À la fin de chaque article un indicateur de fin d'article a été introduit, avec la catégorie EOF. Chaque token est identifié par sa position dans le fichier résultant. Le résultat final est détaillé dans le tableau 6.20.

Tab. 6.20 : Statistiques du corpus de référence

(a) Occurrences

Mots	28 537
Nombres	210
Ponctuations	1 691
Fins d'article	51
Total tokens	30 489

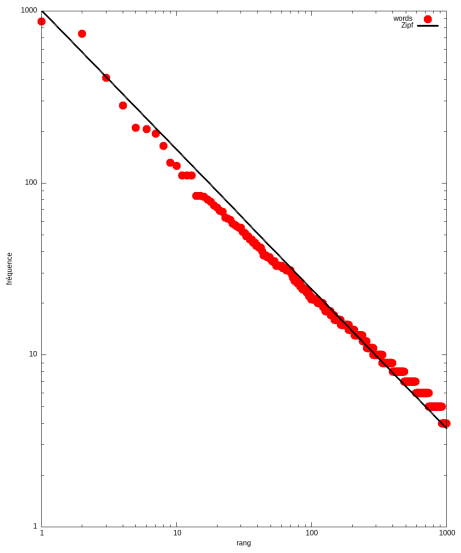
(b) Tokens uniques

Mots uniques	11 427
dont hapax (une seule occurrence)	7 942
Tokens uniques	11 539

On observe que

- les mots en arabe représentent plus de 93% des occurrences, comme dans le corpus général, et de 99% des tokens uniques ;
- les mots uniques montrent un taux de lexicalisation unique de $\sim$ 40%, ce qui est cohérent pour un corpus standard de petite taille, à comparer au taux de $\sim$ 3% du corpus général ;
- les hapax représentent $\sim$ 69% du vocabulaire, ce qui montre que les textes de la catégorie culture appartiennent à un genre plus littéraire (plus écrit) que la moyenne des documents du corpus, avec une plus grande diversité lexicale.

Ce corpus est donc linguistiquement centré, sans bruit structurel excessif, avec une grande variabilité lexicale et un traitement correct des unités fonctionnelles. La fréquence élevée des hapax indique qu'il pourra être exploitée efficacement dans des tâches d'évaluation de la couverture lexicale. La figure 6.3 montre les 10 mots les plus courants de ce corpus, qui sont des mots grammaticaux, et la figure voisine montre sa courbe de Zipf en échelle logarithmique ; la pente est de 0,81 et la constante de Zipf est de 10. Ces données reflètent une distribution cohérente avec un corpus représentatif ; en effet, ces propriétés (courbe de Zipf et mots grammaticaux en tête des fréquences) n'apparaissent pas avec des corpus



في	872
من	739
على	410
ان	282
الى	209
التي	206
عن	194
الذي	164
مع	131
هذا	126

Fig. 6.3 & Tab. 6.21 : Courbe Zipf du Gold Corpus et mots fréquents

trop petits ou trop peu variés. La pente un peu faible s'observe souvent pour les corpus en arabe (pour des corpus du même type en français ou en anglais elle est proche de 0,9).

Il y a quatre mots qui sont en réalité une séquence de deux mots collés par erreur (sur des caractères qui ne se lient pas aux caractères voisins). Nous avons introduit une catégorie ERR, accompagnée des numéros de ligne où figurent les mots séparés. Ces numéros sont enregistrés avec des identifiants dans une plage au-delà de la plage des tokens (ici 50 000). Quatre de ces mots sont uniques. Dans notre *gold corpus*, il y a donc en fait 30 497 tokens, 28 542 mots et 11 431 mots uniques.

6.4.1 Analyse catégorielle

Commençons par indiquer les catégories utilisées avec leur nombre, tableau 6.22. Nous avons ajouté, à nos catégories initiales, EOF, ERR, PUNC, NUM et AB. Nous avons déjà évoqué EOF (fins d'articles) et ERR (erreurs de mots collés). PUNC désigne la ponctuation. NUM désigne les nombres qui peuvent être considérés comme des noms et peuvent être suivis ou précédés de clitiques (non collés en raison des particularités de l'écriture arabe). AB désigne des abbréviations, c'est à dire une lettre unique, qui a une fonction nominale. Du coup, le total des noms devrait inclure NUM et AB.

On observe une prédominance des noms sur les verbes, les substantifs étant les plus fréquents. Dans les verbes, l'inaccompli est en tête, et l'impératif une rareté.

Tab. 6.22 : Catégories finales des tokens

N	noms	17 338
NS	substantifs	13 668
NA	adjectifs	3 670
AB	abbréviations	34
NUM	nombres	210
V	verbes	3 479
VA	accomplis	1 375
VI	inaccomplis	2 102
V0	impératifs	2
P	particules	7 682
PUNC	ponctuations	1 691
EOF	fins d'article	51
ERR	erreurs	4
ADD	corrections des erreurs	8
Total		30 539

Ces éléments renforcent la crédibilité de Stanford par rapport à Al-Khalil.

Nous allons maintenant voir les matrices de confusion des deux étiqueteurs. Nous commencerons par éliminer les catégories :

- EOF, qui n'existe pas dans le corpus original ;
- PUNC, éliminée par Al-Khalil, et qui de toutes façons est détectée automatiquement à 100% ;
- ERR, qui n'a pas de correspondant chez les autres ;
- ADD, idem.

Pour NUM, nous le gardons pour Stanford mais pas pour Al-Khalil. Nous fusionnons AB et NUM dans la sous-catégorie NS, puisqu'ils en sont des cas particuliers (et non distingués par aucun étiqueteur).

Le tableau 6.23 présente la matrice de confusion de Stanford pour les grandes catégories. La meilleure performance est pour les noms, avec une F-mesure de 89,5%. Pour les autres catégories, le rappel est supérieur à la précision. La F-mesure est 80% pour les verbes et pour les particules. Le rappel pour les particules est particulièrement bon (ce qui signifie que si Stanford prédit une particule, on peut être quasi-certain que c'en est une).

Le tableau 6.24 indique la matrice de confusion des sous-catégories. Le rappel et la précision de V0 sont nuls. La meilleure sous-catégorie pour la précision est celle

Tab. 6.23 : Matrice de confusion de Stanford, POS1

Gold \ Stanford	N	V	P	Total	Précision
N	17.405	168	9	17.582	99%
V	1.093	2.386		3.479	69,6%
P	2.833	223	4.626	7.682	60,2%
Total	21.331	2.777	4.635		
Rappel	81,6%	85,9%	99,8%		

des substantifs (F-mesure 90%), suivie des adjectifs (F-mesure également 90%) ; pour le rappel, ce sont les verbes à l'inaccompli (F-mesure 80%), suivi par les adjectifs à nouveau. La précision moyenne (macro) est 76,3%, le rappel moyen est 89,1%, et la F-mesure moyenne est 80%.

Tab. 6.24 : Matrice de confusion de Stanford, POS2

Gold \ Stanford	NS	NA	VA	VI	VO	Total	Précision
NS	13.623	123	132	25	–	13.912	97,9%
NA	490	3.169	7	4	–	3.670	86,3%
VA	455	54	851	15	–	1.375	61,9%
VI	501	81	30	1.487	3	2.102	70,7%
V0	2	–	–	–	–	2	0%
Total	17.850	3.481	1.243	1.531	3		
Rappel	76,3%	91%	68,5%	97,1%	0%		

Dans le détail, on observe que quand Stanford étiquette mal un verbe ou une particule, il les classe en général comme substantifs. Inversement, quand il attribue à tort l'étiquette de nom, c'est le plus souvent une particule. Le reste des erreurs est plus ou moins réparti.

Le tableau 6.25 présente la matrice de confusion d'Al-Khalil pour les grandes catégories. La catégorie U chez Al-Khalil indique les mots non étiquetés. Par rapport à Stanford, la précision est moins bonne pour toutes les catégories sauf les verbes ; le rappel est meilleur pour les noms, et pratiquement équivalent pour les particules. La F-mesure donne 80% pour les noms, 60% pour les verbes (avec un rappel particulièrement mauvais), 70% pour les particules. La précision moyenne (macro) est 70,2%, le rappel moyen est 77,3%, et la F-mesure moyenne est 70%.

En dépit de la quantité de mots inconnus, quand Al-Khalil prédit une particule, on est pratiquement sûr que c'en est une : même si le taux d'erreur est supérieur de 1,5% à celui de Stanford.

Le tableau 6.26 indique la matrice de confusion des sous-catégories. La plus

Tab. 6.25 : Matrice de confusion d'Al-Khalil, POS1

Gold \ AlKhalil	N	V	P	U	Total	Précision
N	11.202	2.887	48	3.235	17.372	64,5%
V	68	3.186	25	200	3.479	91,6%
P	1.168	985	4.176	1.353	7.682	54,4%
Total	12.438	7.058	4.249	4.788		
Rappel	90,1%	45,1%	98,3%			

mauvaise catégorie pour la précision et le rappel est celle des adjectifs. La détection de V0 est également très mauvaise, en raison d'un rappel très faible (F-mesure 0%). La catégorie la mieux détectée, en précision et en rappel, est celle des verbes à l'inaccompli (F-mesure 80%, comme pour Stanford).

Tab. 6.26 : Matrice de confusion d'Al-Khalil, POS2

Gold \ AlKhalil	NS	NA	VA	VI	VO	U	Total	Précision
NS	7.496	1.256	309	177	2.082	2.337	13.702	54,7%
NA	1.867	583	38	8	273	898	3.670	15,9%
VA	39	1	626	51	532	113	1.375	45,5%
VI	24	4	14	1.512	449	87	2.102	71,9%
V0	–	–	–	–	2	0	2	100%
Total	9.426	1.844	987	1.748	3.338			
Rappel	79,5%	31,6%	63,4%	86,5%	0,1%			

En résumé, cette analyse montre que Stanford a dans l'ensemble de meilleures performances, ce que montre la différence en F-mesure moyenne (10% de plus) avec un pic pour les substantifs et les particules ; Al-Khalil est meilleur en précision pour la catégorie verbale et ses sous-catégories, mais avec un rappel moins bon. Certaines tendances sont communes, en particulier la meilleure détection des particules et des verbes à l'inaccompli.

6.4.2 Segmentation manuelle

Après avoir appliqué, avec l'aide de mon équipe, les différents segmenteurs sur ce corpus, j'ai élaboré un outil en java qui affiche les différentes analyses possibles fournies par les différents segmenteurs et mon pseudo-segmenteur. Il ne reste plus qu'à décider laquelle est la bonne ou d'en ajouter une autre. Cet outil (figure 6.4) qui facilite la comparaison et la correction montre également le contexte gauche et droit de chaque segmentation afin de pouvoir le vérifier (la taille du contexte

est paramétrable). Ainsi, il sera facile pour un chercheur d'accroître la taille de la partie manuellement contrôlée du corpus Kalimat.



Fig. 6.4 : Outil d'aide à la correction manuelle

Les champs de cette interface sont :

- **Choose** : Choix du corpus à segmenter.
- **Show All** : Checkbox qui dit si on veut traiter tous les mots ou seulement les mots non segmentés.
- **Total words** : Nombre de mots restant à traiter.
- **Previous / Next** : naviguer dans les mots
- **Complete** : aller directement au premier mot non segmenté.
- **Write** : enregistrer les mots corrigés.
- **Count** : rayon de la fenêtre à afficher autour du mot.
- **After / Word / Before** : affiche le mot dans son contexte.
- L'identifiant du mot.
- La boite de droite affiche les différentes segmentations disponibles.
- **Segment** : segmentation choisie, à corriger manuellement.
- **New** : pour corriger la segmentation.
- **POS Tags** : par défaut, POS1, POS2 selon Stanford, à corriger si faux.
  - Pos1 : présente les valeurs {N, V, P, U}.
  - Pos2 : pour Pos1 = N, Pos2 présente {NA, NS}, et pour Pos1 = V, Pos2 présente {VA, VI ou rien}.

Une fois la phase manuelle terminée, nous obtenons pour chaque occurrence d'un mot un vecteur de caractéristiques de la forme :

procl.debut	procl.	procl.fin	préf.	radical	suff.	encl.

S'ajoutent à ces caractéristiques POS1 et POS2, faisant au total un vecteur de 9 colonnes.

6.4.3 Description de la segmentation

En intégrant les 8 segmentations supplémentaire générées par les erreurs (ERR), dont 4 sont uniques, le nombre total de segmentations est 28.700, et le nombre de segmentations différentes (segmentations uniques) de notre corpus est de 11 444.

Aucun mot n'est découpé en plus de cinq parties (sur les huit théoriquement possibles). Le tableau 6.27 présente les statistiques de segmentation. Près de 60 % des mots sont morphologiquement segmentés, ce qui est tout à fait représentatif. Plus de 84% des mots contiennent une ou deux parties, et le découpage en deux parties est le plus fréquent. Les formes à quatre ou cinq parties, bien que minoritaires (1,56%) sont linguistiquement cruciales.

Tab. 6.27 : Répartitions des segmentations par nombre de segments

Parties	**Nombre**	%
Une	11 678	40.92%
Deux	12 390	43.42%
Trois	4 019	14.04%
Quatre	442	1.54%
Cinq	6	0.02%
Total	28 535	

Des exemples des partitions les plus fréquentes avec au moins deux parties figurent dans le tableau 6.28.

Tab. 6.28 : Type de segmentations les plus fréquentes

Parties	**Mot**	**Procl1**	**Proc**	**Proc.fin**	**Préf.**	**Radical**	**Suff**	**Encl.**	**Occs**
Cinq	ويفداك	و			ي	فد	ا	ك	3
Quatre	والاستفتاحية	و		ال		استفتاحي	ة		163
Trois	الحالية			ال		حالي	ة		1499
Deux	الفكري			ال		فكري			5108

Le tableau 6.29 indique la répartition des clitiques et affixes. Sur la totalité des mots du gold corpus (y compris les mots non segmentés),

- 44% (13 418) contiennent des proclitiques ; cela reflète l'importance des éléments grammaticaux comme les conjonctions (و, ف), les prépositions (ب, ل, ك) et l'article défini (ال) ;
- 23.4% (7 144) contiennent des suffixes, qui représentent généralement les marques de personne, nombre et genre (ـون, ـات, ـة...), soulignant l'importance de la flexion dans la langue ;
- 8% (2 434) contiennent des enclitiques (possessifs et pronoms objets) ;
- 6.3% (1 936) contiennent des préfixes, ce qui est relativement faible.

On observe une fréquence croisée : en début de mot, les clitiques sont beaucoup plus fréquents que les affixes (44% – 6,3%) ; en fin de mot c'est l'inverse, les affixes sont les plus fréquents, et les clitiques moins fréquents (23,4% – 8%). Notre corpus présente tous les clitiques existants et une très large majorité des affixes flexionnels existants, ce qui montre qu'il a une bonne couverture linguistique pour notre objectif.

Tab. 6.29 : Distribution des clitiques et affixes

(a) Proclitiques

	Occs
ال	7896
و	2995
ب	975
ل	946
ف	229
ما	131
ك	82
س	69
أ	54
ا	32
م	9
Total	13 418

(b) Suffixes

	Occs
ة	2941
ي	1292
ات	783
ا	659
ت	630
ين	307
ون	182
ما	116
وا	107
ان	45
ى	42
و	40
Total	7 144

(c) Enclitiques

	Occs
ه	934
ها	800
هم	214
نا	153
ي	99
ك	97
كم	56
ني	32
هما	30
تم	7
هن	7
ن	4
كما	1
Total	2 434

(d) Préfixes

	Occs
ي	1130
ت	689
ن	117
Total	1 936

Concernant le nombre de segmentations multiples (mots ayant plusieurs segmentations), il faut déjà observer que les hapax ne peuvent avoir qu'une segmentation, ce qui ne laisse que 3 489 mots pouvant avoir plus d'une segmentation. Le

corpus présente 13 formes ayant plusieurs segmentations sur 28 535 formes, c'est-à-dire 0,05%, représentant 180 segmentations sur un total de 28 700 segmentations, soit 0,63%. Cela équivaut, en moyenne, à deux formes à désambiguïser par page dans un livre standard.

Cette faible proportion permet de comprendre pourquoi les segmenteurs non contextuels continuent à être populaires. Pourtant, une segmentation fiable doit absolument traiter le cas des mots à plusieurs segmentations. Le tableau 6.30 illustre les types d'ambiguïté présents dans notre corpus (un tableau complet figure en annexe B).

Tab. 6.30 : Segmentations multiples d'un même mot

Mot	Segment.	Contexte	Cat.	Glose
الحاجة	ال حاجة	أن الحاجة إليها	P3+NS	[que] le besoin [(qu'on a) d'elle]
	ال حاج ة	أن الحاجة عليّه	P3+NA+s	[que] la vieille [Aliyya]
فقد	فقد	فقد أبويه	VA	il a perdu [ses parents]
	ف قد	فقد سبقتها	P2+P	et déjà [elle a accepté]
ولديه	ولد ي ه	مع ولديه	NS+s+E	[avec] ses enfants
	و لدي ه	ولديه ميول	P1+P+E	et avec lui [(sont) des penchants = il a des penchants]
أحدث	أحدث	أحدث أعماله	NA	[ses œuvres] les plus récentes
	أ حدث	أحدث عدد صفحاته	p+VI	je mets à jour [le nombre de ses pages]

Une bonne partie de ces doubles segmentations (quatre cas sur les treize) provient du statut du suffixe féminin, selon que le mot est un substantif (NS) ou un adjectif (NA), comme dans le premier exemple du tableau. Pour le reste, quatre cas concernent des combinaisons proclitique + particule, majoritairement confondues avec un nom (plutôt substantifs) ou avec un verbe à l'accompli, et une fois avec particule + enclitique. Parmi les derniers cas, cinq sont des confusions entre nom et verbe, et les cas restants sont des confusions entre le verbe à l'accompli avec un préfixe dérivationnel, non segmenté par choix théorique, et à l'inaccompli avec un préfixe flexionnel, ou entre particules et noms. On constate donc que les confusions possibles ne sont pas distribuées aléatoirement (peu de confusions entre noms ou entre particules), et qu'il serait donc possible d'avoir une probabilité plus ou moins grande de l'existence d'autres segmentations possibles à partir d'une première segmentation.

Pour finir, nous avons repéré quelques anomalies qui nous amènent à prendre des précautions supplémentaires. La plus importante est qu'en présence d'un nombre en chiffres, les clitiques sont forcément isolés du mot minimal par l'écriture ; ex. « 64 سنة ه » ; ici, on voit à la fois l'importance de considérer ces nombres comme des mots, mais isolants par nature. Il en serait de même pour des mots d'un autre

script que l'arabe. Par ailleurs, même si c'est rare, il arrive que certains enclitiques existent sous forme non clitique, ex. بل هما, ce qui fait qu'on ne peut pas toujours détecter cette anomalie, sauf à prendre en compte le script ou le caractère numérique.

6.4.4 Description du gold standard

Notre gold standard est disponible sur https://gitlab.com/hussein.awdeh/segmentationgoldarabiccorpus/ dans le répertoire *Gold Corpus*. On y trouve :

- Original files : contient les fichiers *txt* d'origine avec la segmentation AlKhalil + Viterbi et la catégorisation Stanford (un mot par ligne).
- *All tokens.txt* : la liste de tous les tokens dans l'ordre des documents originaux, un mot par ligne, avec ajout du mot *ENDFILE* pour marquer la fin des documents.
- *Unique tokens.txt* : la liste des tokens uniques, les mots en arabe d'abord, et à la fin le reste (ponctuations, nombres, caractères non arabes).
- Automatic segmentations : les fichiers csv représentant le résultat de la segmentation de chaque mot arabe par les segmenteurs Madamira, Farasa, Stanford et AlKhalil.
- Gold segmentation : le fichier csv représentant la segmentation et les catégories manuellement établies par nos soins.

Le répertoire *Tools* contient des outils qui peuvent servir à la fois pour le gold standard et pour le silver standard, en particulier un programme écrit par Mai Khalil qui permet de répartir les segments dans le vecteur en fonction de la catégorie, le programme que j'ai écrit pour aider à la segmentation / catégorisation manuelle, pour les chercheurs qui souhaiteraient prolonger mon travail, et les tables de conversion qui permettent de convertir les tags de Stanford ou AlKhalil dans notre système de tags.

6.5 Évaluation des segmenteurs

À l'aide du corpus de référence défini ici, nous allons maintenant évaluer l'ensemble des segmenteurs de l'état de l'art (Awdeh, Saadiyeh et al., 2025), qu'ils soient open-source ou disponibles gratuitement pour les chercheurs, par rapport à la tâche de segmentation elle-même, et non plus, comme dans la plupart des

évaluations données dans la littérature, par rapport à une tâche externe comme la recherche d'information. Par conséquent, on ne sera pas surpris si les résultats et le classement des algorithmes diffèrent sensiblement des résultats obtenus dans la littérature.

Deux problèmes se posent d'emblée. D'une part, il n'y a aucun standard pour les entrées des segmenteurs, ce qui nous a amenés à préparer presque autant de versions de notre corpus de référence que de segmenteurs. D'autre part, les segmenteurs ont des sorties très variables en fonction du type de segmentation visé (lemmatiseur, segmenteur de clitiques, racinisateur, etc.). Pour résoudre ce deuxième problème, nous avons distingué deux procédures.

- une pour établir la qualité de détection du radical, seul point commun à tous les segmenteurs, que nous avons appliquée pour ISRI, Khoja, Tashaphyne, Assem, Safar, Motaz, Farasa, Stanford, Madamira, Light10, Stanza et Al-Khalil,
- et une pour établir la qualité d'une segmentation complète, pour laquelle nous n'avons que les segmenteurs Madamira, Farasa, Al Khalil (avec Viterbi) et Stanford. Pour Al-Khalil avec Viterbi, nous n'avons que les résultats de la segmentation, le code n'est pas disponible.

Les segmenteurs étant en Java ou en Python, leur exécution ne pose pas de problème. Une fois le corpus adapté à l'entrée pour un segmenteur, la sortie est convertie en un format commun qui puisse être comparé. La figure 6.5 récapitule le processus :

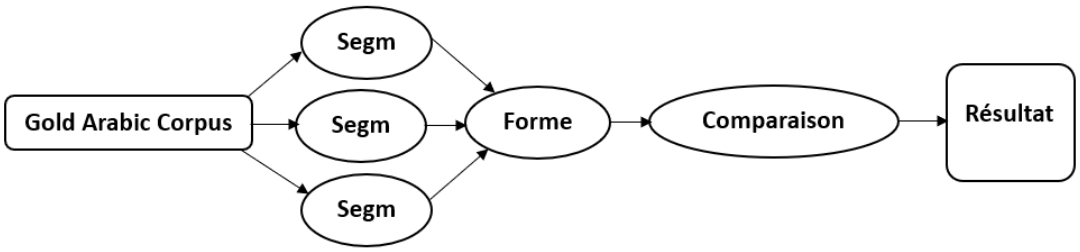


Fig. 6.5 : Processus d'évaluation

Nous indiquons ici brièvement les éléments nécessaires par les pré- et post-traitements :

- Pour ISRI, Stanza, Tashaphyne, Assem, entrée envoyée par API Python mot par mot à partir de notre fichier et récupération du mot à insérer dans le fichier output.

- Pour Farasa, entrée envoyée par API Java mot par mot à partir de notre fichier et récupération du mot à insérer dans le fichier output.
- Pour Safar, Motaz, Light10, transmission du fichier à la plateforme Safar et récupération d'un fichier XML des résultats.
- Pour Khoja, Stanford NLP, Al Khalil, Madamira, fichier d'entrée converti au format demandé par chacun (XML, texte) transmis par ligne de commande Java et récupération d'un fichier de sortie au format texte ou XML suivant les cas.

Notre procédure de validation est la suivante :

1. Convertir le Gold Corpus en un fichier adapté ;
2. Récupérer les sorties et les convertir dans un format commun ;
3. Comparer le radical avec celui du Gold Corpus ou comparer la segmentation complète avec celle du Gold Corpus ;
4. Calculer le résultat comme le nombre de cas justes sur le nombre total de cas (on ne tient pas compte des ressemblances partielles, c'est tout ou rien).

Safar forme un cas particulier : il présente plusieurs résultats. Nous avons choisi de valider son résultat si la bonne solution figurait dans la liste de ses propositions, mais ça introduit un biais important dont il faudra tenir compte.

Enfin, dernier élément à prendre en compte, nous avons un peu élargi ici notre définition des suffixes, pour intégrer quelques suffixes dérivationnels très fréquents (ي pour les noms, ت pour les substantifs, ها pour les particules) et que les segmenteurs en lice découpent toujours. En effet, sans cela, les résultats globaux sont beaucoup plus faibles et il est difficile d'effectuer une comparaison honnête.

6.5.1 Analyse des résultats

Concernant la détection du radical, nous avons testé douze des segmenteurs décrits dans l'état de l'art. Les résultats sont présentés dans le tableau 6.31, sur deux évaluations : le nombre de détections correctes divisé soit par le nombre de segmentations uniques (11 444), soit par le nombre total de segmentations (28 700). La différence entre les deux est due à la fréquence relative des radicaux. Ainsi, si les radicaux A et B apparaissent respectivement 100 et 20 fois, détecter A donne une couverture plus grande que trouver B. Par conséquent ce tableau montre que

la plupart des segmenteurs trouvent les radicaux les plus fréquents. La seule exception est Khoja, qui ne trouve pas en général les radicaux les plus fréquents. Ensuite vient Light10, which choisit des radicaux de moyenne fréquence. À l'opposé, Stanza, Stanford, Al-Khalil Stemmer, Farasa et Madamira accroissent leur couverture en occurrences entre 10% et 13.5%, ce qui signifie qu'ils sélectionnent des radicaux fréquents de manière systématique.

Tab. 6.31 : Détection du radical

	Unique	**Occs**	**Durée**	**Langage**
Farasa	74,4%	85%	9”	Java
Safar	73,1%	79,8%	14”	Java
Tashaphyne	64,8%	69,1%	30,1”	Python
Madamira	56,1%	65,8%	1’13”	Java
Motaz	53,7%	60,4%	3”	Java
Assem	53%	58,9%	7,3”	Java
Al Khalil Stemmer	50,3%	62,2%	17”	Java
Light10	48,8%	50,0%	1”	Java
Stanza	44,3%	57,9%	23’54,7”	Python
Stanford	34,6%	46,2%	48”	Java
ISRI	25,7%	39,5%	3,1”	Python
Khoja	24,6%	22,2%	1’45”	Java

En prenant le premier indicateur, il y a quatre niveaux : le plus bas, autour de 25% (± .5%), à 34.6%, moyen (autour de 49% ± 5%), et le plus haut autour de70%(± 5%). Au plus bas niveau, nous trouvons les racinisateurs, ce qui n'est pas surprenant car leur rôle premier n'est pas d'extraire des radicaux. Au passage, ISRI, sans dictionnaire de racines, obtient des performances légèrement meilleures que Khoja, avec dictionnaire, conformément aux affirmations des auteurs de ISRI. Le segmenteur de Stanford est assez mauvais ici.

La plupart des extracteurs de radicaux les plus connus et les plus utilisés sont au niveau moyen, dans lequel la probabilité d'une erreur est proche de 50%. L'extracteur neuronal Stanza, qui n'a aucune connaissance de la langue est le plus bas de ce groupe, bien que son choix de radicaux lui donne une couverture en occurrence pas très éloignée du plus haut du groupe. Light1 est dans ce groupe et sa version améliorée par Motaz le dépasse, comme Assem et l'extracteur d'Al-Khalil. Madamira est le meilleur du groupe.

Au plus haut niveau, il faut se rappeler que notre manière de calculer a favorisé Safar, lemmatiseur markovien ; même s'il a le bon radical, il peut aussi en avoir d'autres et dans ce cas, il faudra encore trouver lequel est le bon. Si on divisait

son succès par le nombre de propositions qu'il fait, ses performances tombent au niveau moyen.

Farasa est indiscutablement le meilleur, suivi par Tashaphyne. La durée est le temps mis par chaque segmenteur pour donner les résultats sur tous les mots. Nous donnons cette information à titre indicatif, car les segmenteurs utilisent plusieurs langages et ne sont pas vraiment comparables en terme de travail effectué ; certains segmenteurs donnent plein d'autres informations à côté du radical, et d'autres juste le radical. C'est toutefois intéressant que Stanza soit le segmenteur le plus gourmand en temps calcul.

Les douze segmenteurs s'accordent sur le même radical dans 5,2% des cas, ce qui est très peu et montre qu'une stratégie avec un ensemble serait vouée à l'échec.

Les erreurs sont principalement dues à une insuffisance de découpage (*understemming*), et principalement à des clitiques non enlevés. Assem, ISRI et Motaz échouent souvent à enlever des proclitiques (prépositions ل ,ف, article ال, conjonction و). Tashaphyne n'enlève pas certains clitiques comme هم (enclitique) et ال. Les meilleurs segmenteurs font très rarement des erreurs de découpage insuffisant (comme dans certains cas de و non enlevé par Safar). Les erreurs de sur-découpage (*overstemming*) sont essentiellement l'enlèvement du proclitique و dans des mots où il appartient au radical (Farasa, Safar, Light10, Assem), et, plus difficile à observer, ة, qui est en général un suffixe, ne devrait pas être enlevé quand la forme sans lui n'existe pas, avec les substantifs (ex. مرة).

Le tableau ci-dessous 6.32 présente un résumé des principales erreurs des segmenteurs testés ; elles portent beaucoup plus fréquemment sur des clitiques que sur des affixes.

Tab. 6.32 : Erreurs les plus courantes

Type d'erreur	**Segmenteur**
Confond أ et ا : ex. le radical de أو est او	ISRI , Motaz, Light10, Madamira
N'enlève pas le proclitique و du mot : ex. le radical de وكتاب est وكتاب	ISRI, Motaz, Safar, Light10
Enlève و, ة quand ils font partie du mot : ex. مرة est découpé	Assem, Farasa, Safar, Light10
N'enlève pas l'article ال du mot	Assem, Tashaphyne, Safar
Échoue à trouver le radical de mots comme كان	Tashaphyne
N'enlève pas les proclitiques ل, ف et ب	Assem, ISRI, Motaz
N'enlève pas l'enclitique هم du mot	Tashaphyne

Certaines erreurs, comme la première du tableau, sont des problèmes liés à l'orthographe des mots. Nous avons donc effectué une normalisation orthographique

(voir section 1.3.2) des données de sortie aussi bien que des données d'entrée, et les statistiques indiquées ci-dessus sont calculées après cette procédure.

Les résultats concernant la qualité de la segmentation complète figurent dans le tableau 6.33. L'exactitude est, comme précédemment, calculée comme le nombre de segmentations exactes sur le nombre total de segmentations. Madamira est ici aussi largement en tête.

Tab. 6.33 : Segmentation complète

Segmenteur	**Exactitude** %
Farasa	75,17%
Madamira	53%
Stanford	48,5%

Pour cette partie, nous avons utilisé une segmentation grammaticale stricte (sans aucun affixe lexical). Étant donné que Kalimat propose une version d'Al-Khalil désambiguïsée en contexte à l'aide d'un algorithme de Viterbi (Boudchiche et al., 2017), il semblait pertinent de le comparer aux trois autres segmenteurs qui offrent une segmentation complète : Farasa, Stanford et Madamira. Cependant, nous avons rencontré de nombreuses difficultés qui ont conduit à son exclusion du reste de l'étude :

- La racine n'est pas fournie, seulement le mot minimal ; cela peut être corrigé facilement en soustrayant les clitiques et les affixes.
- La segmentation est absente pour 15 % des mots.
- Les segmentations erronées sont très nombreuses.
- Les deux-points utilisés pour séparer les segments peuvent apparaître ailleurs.
- En particulier, un deux-points peut séparer le clitique/l'affixe et son analyse (par exemple التعريف : ال) ou non (par exemple حرف جار و), même avec le même clitique ou affixe.
- Les combinaisons (par exemple deux clitiques et un préfixe) sont présentées de manière incohérente, soit comme une seule chaîne, soit divisées en plusieurs parties.

Le tableau 6.33 présente les résultats de la segmentation complète ; ici, l'exactitude est calculée comme le nombre de segmentations exactes divisé par le nombre total de segmentations. Ces résultats confirment que Farasa est le meilleur choix,

largement supérieur aux autres, Madamira et Stanford. Stanford obtient de bien meilleurs résultats ici que dans le test de racinisation, ce qui indique qu'il est davantage centré sur les affixes grammaticaux ; les autres incluent des affixes dérivationnels dans leur segmentation, ce qui entraîne une performance plus faible sur la racine.

Pour le traitement des 13 formes ambigües de notre corpus, Farasa est un segmentateur non contextuel, il ne peut donc pas les résoudre. Madamira est le seul outil à avoir correctement désambiguïsé l'une des 13 formes. Ces résultats ont été confirmés sur un second lot de textes contenant 22 formes potentiellement ambiguës dans des contextes permettant clairement de choisir une segmentation plutôt qu'une autre, où Stanford n'a pas réussi à désambiguïser et Madamira n'en a traité correctement que très peu.

Le tableau 6.34 présente la proportion de segmentations correctes selon les catégories morphosyntaxiques. Les particules, qui ne comportent aucun affixe, sont les mieux segmentées (dans ce cas, Madamira surpasse Farasa) ; viennent ensuite les noms, pour lesquels Farasa réalise deux fois plus de segmentations correctes que les autres outils. Les verbes, qui possèdent le plus d'affixes, sont plus difficiles à segmenter : même Farasa n'en segmente correctement qu'un sur deux.

Tab. 6.34 : % par catégorie grammaticale

	Farasa	**Madamira**	**Stanford**
P	93,6	92,6	79,2
N	74,8	41,3	39,9
V	36	24	23,3

Le tableau 6.35 présente les probabilités de détection correcte de chaque segment de la segmentation pour les trois segmenteurs complets. On constate que d'une manière générale, les segmenteurs sont plus performants pour détecter les clitiques que les affixes ou le radical. Une exception concerne le troisième proclitique, où Madamira et Stanford échouent souvent en raison d'une mauvaise interprétation de la présence ou non de ال (le déterminant). Farasa se montre supérieur dans toutes les parties ; il détecte mieux les préfixes que les suffixes.

Lorsque le radical est correctement identifié, Farasa produit une segmentation incorrecte dans moins de 1 % des cas, principalement à cause d'un mauvais découpage des enclitiques et des suffixes. Pour Madamira, ces erreurs représentent 1,3 %, dues à des combinaisons incorrectes entre proclitique médian, suffixe et enclitique. Pour Stanford, elles représentent 0,6 %, principalement à cause d'erreurs de segmentation sur les suffixes et enclitiques, notamment dans les proclitiques 2 et 3. Dans tous ces cas, le premier proclitique et le préfixe sont toujours correctement

détectés.

L'ensemble des trois positions de proclitiques est correctement identifié par Farasa dans 98,6 % des cas, contre 70,9 % pour Madamira et 66,4 % pour Stanford, ce qui montre que la difficulté à segmenter correctement ال joue un rôle important, ce qui n'est pas surprenant au vu de sa fréquence dans les segmentations en moins de cinq parties (cf tableau 6.28 des partitions les plus fréquentes).

Tab. 6.35 : % des parties correctes

	Farasa	**Madamira**	**Stanford**
procl1	99,6	99,5	96,3
procl2	99,2	98,9	97,7
procl3	99,9	72,6	72,6
prefix	93,3	93,2	93,2
stem	78,4	53,3	48,4
suffix	87,1	84,3	84,4
enclitic	98,1	98	97,6

Madamira et Stanford ont tendance à sous-découper, c'est à dire à présenter un nombre de parties inférieur au vrai nombre ; pour Madamira c'est dans une très large majorité des cas (plus de 99%) ; pour Stanford, c'est dans 4 cas sur 5. En revanche Farasa présente une répartition plus équilibrée entre sur-découpage (11,7% des cas) et sous-découpage (8,5%). Les tableaux 6.36 montrent la répartition en catégorie de ces cas :

Tab. 6.36 : Nombre de parties

(a) Sur-découpage

	N	V	P
Farasa	**18,7**%	0,3%	1,3%
Madamira	0,4%	0,4%	0,7%
Stanford	1,2%	0,9%	**15,5**%

(b) Sous-découpage

	N	V	P
Farasa	1%	**58**%	2,6%
Madamira	**56**%	**75**%	2,6%
Stanford	**57,2**%	**75,3**%	2,1%

Il est à noter que la plus grande quantité de sous-découpages est pour les verbes, où même Farasa sous-découpe. Cela signifie que le problème de non segmentation correcte des verbes par ces segmenteurs est dû à la non détection d'un clitique ou d'un affixe ; et il s'agit plus probablement d'un affixe, étant donné que le problème devrait arriver aussi pour les noms s'il s'agissait de clitiques.

Concernant la possibilité d'utiliser les segmenteurs de manière complémentaire, nos relevés montrent que le principal point qui peut fonctionner pour les catégories

est la qualité de détection des particules par Madamira, car pour les verbes ni Madamira ni Stanford ne pourrait faire mieux que Farasa. On peut aussi détecter les cas de surdécoupage par Farasa. Une meilleure performance en matière de résolution de l'ambiguïté peut difficilement reposer sur les deux autres uniquement.

6.6 Conclusion

Dans ce chapitre, nous avons abordé le problème d'élaborer un corpus de référence de taille respectable à partir d'un petit corpus manuellement annoté et des résultats des segmenteurs de l'état de l'art. Pour ce faire, nous avons défini un certain nombre d'outils concernant la définition des clitiques et affixes, ainsi qu'un jeu d'étiquettes qui s'est avéré suffisamment large pour permettre une comparaison approfondie des autres systèmes d'étiquettes ; il absorbe des divergences secondaires, mais a permis de détecter les anomalies importantes.

Nous avons pu déterminer que la segmentation Al-Khalil est inutilisable à grande échelle, que Farasa donnait les meilleurs résultats, suivi par Madamira et Stanford, et donner des éléments permettant de connaître leurs forces et leurs faiblesses. Il apparaît en tout cas dans un premier temps qu'il n'est pas évident d'utiliser une stratégie d'ensemble.

Nous avons pu comparer les étiquettes des parties du discours selon Al-Khalil + Viterbi et Stanford, et avons pu observer que Stanford était bien meilleur pour reconnaître les noms et Al-Khalil pour reconnaître les verbes. Par conséquent, garder Stanford comme référence principale et le coupler à Al-Khalil pour améliorer sa reconnaissance des verbes pourrait être une stratégie viable.

Au chapitre suivant, nous utiliserons ces résultats pour examiner les conditions d'une segmentation automatisée d'un grand corpus de la meilleure qualité possible, en même temps que des conditions d'entraînement d'un segmenteur basé sur l'apprentissage séquentiel.

Chapitre 7

Éléments pour un segmenteur

Sommaire

7.1 Introduction 196
7.2 Données d'entrée 196
7.2.1 Prétraitement et encodage des données catégorielles . . 197
7.2.2 Prétraitement et encodage des segmentations 200
7.2.3 Encodage final 201
7.3 Corpus d'entraînement et données fournies 201
7.3.1 Gold Corpus 201
7.3.2 Tools 202
7.3.3 Silver Corpus 203
7.4 Conclusion 203

7.1 Introduction

Notre ambition initiale dans ce chapitre était de constituer un *silver corpus*, c'est à dire un corpus automatiquement annoté, mais dont les annotations sont considérées comme fiables par rapport aux autres annotations automatiques, et de l'utiliser pour entraîner un segmenteur à apprentissage séquentiel de type GRU (voir section 4.6.2) qui soit capable de prendre en compte le contexte pour produire des segmentations. Mais nos conditions ne nous ont pas permis d'arriver jusque là, et nous présentons ici notre méthodologie, les observations et des éléments de *feature engineering* qui seront nécessaires pour poursuivre ce travail.

Au chapitre précédent, nous avons pu constater qu'une stratégie d'*ensemble* groupant plusieurs segmenteurs avait très peu de chances de réussir, sauf marginalement. En revanche, la convergence globale observée, avec le fait que pratiquement tous les segmenteurs ont des points faibles semblables, assurera une certaine redondance si on transmet en parallèle toutes leurs informations à un super-segmenteur, peut-être capable de repérer, dans leurs divergences, les éléments permettant d'améliorer les résultats.

Mais surtout, nous avons pu constater qu'une stratégie d'*ensemble* groupant nos deux étiqueteurs de parties du discours avait une chance non négligeable d'améliorer leurs résultats. Même si nous avons été conduits à abandonner la segmentation d'Al-Khalil, nous avons découvert que les annotations qu'elle contient sont en revanche particulièrement pertinentes pour détecter les verbes, et c'est précisément le principal point d'achoppement de nos segmenteurs.

Ces observations conduisent aux propositions formulées dans ce chapitre.

Nous décrivons d'abord la préparation des données, puis le site sur lequel nous avons placé tout ce qui nous semble nécessaire pour l'entraînement.

7.2 Données d'entrée

Le corpus dispose des sources d'informations suivantes :

- le mot d'origine
- le mot normalisé
- les étiquettes de POS de Stanford,
- les étiquettes et la segmentation d'Al-Khalil + Viterbi,
- les segmentations produites par Madamira, Stanford et Farasa.
- la segmentation produite par mon pseudo-segmenteur
- la segmentation du gold corpus.

Nous abandonnons la segmentation d'Al-Khalil+Viterbi pour les raisons évoquées au chapitre précédent. En revanche, nous gardons ses étiquettes en raison de la bonne qualité de sa détection des verbes.

7.2.1 Prétraitement et encodage des données catégorielles

Concernant les données catégorielles, nous pouvons soit mettre les étiquettes produites par les deux systèmes (ce qui nous ferait six étiquettes POS), soit trouver une combinaison qui nous permette de ne mettre qu'une série d'étiquettes (donc trois étiquettes POS). Dans le premier cas, ce serait au segmenteur de découvrir lui-même quelles sont les étiquettes les plus fiables à partir du *gold corpus*. Mais étant donné que la taille de celui-ci n'est probablement pas assez importante pour assurer que le segmenteur trouve la meilleure combinaison, la deuxième solution est préférable.

Nous mettons de côté les ponctuations, pour lequel il est relativement trivial de trouver 100% de réussite, ainsi que les erreurs (étiquette ERR), sur lequel le taux de réussite est impossible à apprécier, et bien entendu les mots ajoutés (étiquette ADD) et les fins de fichier (étiquette EOF). Il reste 28 743 mots classés en N, P ou V.

Avec le tableau 7.1 nous voyons que Stanford est globalement meilleur sur l'ensemble des étiquettes POS1 et POS2, même si Al-Khalil est meilleur en précision pour les verbes (mais pas en rappel), comme on l'a vu au chapitre précédent (Stanford est meilleur sur les noms et les noms sont beaucoup plus fréquents).

Tab. 7.1 : Exactitude par POS

	POS1	POS2
Stanford	84,94%	90.84%
Alkhalil	64,58%	49,08%

Nous avons cherché à construire toutes les règles possibles sur POS1, en écartant toute règle dont la probabilité serait inférieure à 75%. Nous indiquons dans le tableau 7.2 le résultat ; le support désigne la proportion de cas sur lesquels la règle s'appuie.

Nous avons mis en gras les règles les plus intéressantes, d'abord en pourcentage de réussite et ensuite en fonction du support. Évidemment nous avons aussi la règle <nombre ⇒ N NS NUM>, qui s'appliquera d'office, avec une probabilité de 100%. Attention, cette règle nécessite de regarder le mot et pas seulement la catégorie. Ces quatre règles réunies réussissent à étiqueter 65% du corpus avec un taux d'erreur de 2,55%.

Tab. 7.2 : Règles sur POS1

Règle	**Proba**	**Support**
Stanford(P) ∧ Al-Khalil(P) ⇒ P	100%	35,75%
Stanford(P) ⇒ P	99,80%	60,21%
Al-Khalil(P) ⇒ P	98,28%	54,36%
Stanford(N) ∧ Al-Khalil(N) ⇒ N	96,81%	64,07%
Stanford(N) ⇒ N	80,46%	98,7%
Al-Khalil(N) ⇒ N	90,06%	64,42%
Stanford(N) ∨ Al-Khalil(N) ⇒ N	77.37%	99.33%
Stanford(V) ∧ Al-Khalil(V) ⇒ V	95,68%	63,81%
Stanford(V) ⇒ V	85,59%	68,60%

Nous prenons ensuite les mots qui n'ont pas été catégorisés à l'étape qu'on vient de voir, et nous cherchons à nouveau les règles, en intégrant les POS2 des deux étiqueteurs, qui peuvent nous aider à trouver la catégorie du mot. Le résultat figure dans le tableau 7.3. Bien que la règle sur VO aie un très petit support, il faut la garder car elle trouve la totalité des VO du corpus et rien que ceux-là. La seule question concerne la dernière règle, dont la probabilité est un peu plus basse, et le support peu important ; après nos tests, elle augmente le taux d'erreur global de 0,2%, donc nous ne la garderons pas.

Tab. 7.3 : Règles sur POS2 et 3

Règle	**Proba**	**Support**
Stanford(VI) ⇒ V	96,47%	6,72%
Stanford(V0) ⇒ V	100%	0,01%
Stanford(D) ⇒ N	98,16%	15,74%
Stanford(DP) ⇒ N	100%	1,69%
Stanford(P) ⇒ N	91,7%	1,74%

L'ensemble de ces règles, qui choisissent systématiquement Stanford pour le POS1, utilisent Al-Khalil et d'autres éléments pour séparer les prédictions de Stanford en deux lots suivant leur fiabilité. Le tableau 7.4 montre le taux d'erreur sur les deux lots, d'une part sur le total des mots, d'autre part sur le total des tokens, après avoir ajouté la règle <Stanford(Punc)⇒ Punc>, qui fonctionne avec un taux de réussite de 100%. Sur le deuxième lot, même si les prédictions de Stanford sont moins fiables, celles de Al-Khalil le sont encore moins (taux d'erreur de 75,5%).

Le tableau 7.5 des performances moyennes de chaque étiqueteur sur les sous-catégories de N et de V dans le lot sélectionné ; il montre que Stanford est ici aussi supérieur. Par conséquent, les prédictions d'Al-Khalil ne sont utiles que dans une phase de prétraitement, pour déterminer les deux lots. Le taux d'erreur dans

Tab. 7.4 : Lots de Stanford

	Lot sélectionné		Lot non sélectionné	
	support	taux d'erreur	support	taux d'erreur
Mots	69,5%	2,7%	30,5%	43%
Tokens	70,8%	2,3%	29,2%	43%

le lot sélectionné est supérieur pour POS2 que pour POS1. Sur l'ensemble des tokens, dans le lot sélectionné, là où le taux d'erreur de POS1 était de 2,3%, le taux d'erreur de POS2 est de 4,4%, ce qui reste acceptable.

Tab. 7.5 : Métriques sur POS2 (lot sélectionné)

	Précision		Rappel		F-mesure	
	N	V	N	V	N	V
Stanford	92,6%	95,9%	96,4%	93,7%	90%	90%
Alkhalil	48,8%	62,9%	55,6%	79,6%	50%	70%

Nous disposons donc d'une méthode nous permettant d'associer un degré de fiabilité aux catégories de mots de Stanford, tout en éliminant la nécessité de conserver les résultats d'Al-Khalil.

Pour les catégories autres que celles étudiées ci-dessus :

- ERR est une erreur dans le corpus d'origine (plusieurs mots collés) ; par conséquent, ni Stanford ni Al-Khalil ne peuvent la détecter.
- EOF est absente du corpus original, et ne peut être détectée. Elle peut être produite et détectée automatiquement dans le *Silver Corpus*, donc n'est pas à prendre en compte.
- ADD (mots ajoutés) est également invisible pour les deux étiqueteurs, et sera absente du *Silver Corpus*.

Les catégories des triplets POS1, POS2, POS3 de Stanford sont interdépendantes. Par conséquent, pour les convertir par *one-hot encoding*, nous n'avons pas besoin de remplacer chaque colonne par autant de colonnes qu'il y a de valeurs possibles. Pour POS1 et POS2, on dispose globalement des six valeurs : NS, NA, VI, VA, VO, PUNC, soit six colonnes, auxquelles s'ajoutent les trois valeurs de POS3. Le total serait donc neuf colonnes, auquelles s'ajoutent une mesure binaire de fiabilité, donc dix caractéristiques en tout.

7.2.2 Prétraitement et encodage des segmentations

Nos segmentations se présentent sous la forme de sept colonnes, avec six données catégorielles destinées à être encodées par *one-hot encoding*. Avec deux proclitiques initiaux, quatre proclitiques médians et deux proclitiques finaux, quatre préfixes et seize suffixes, ainsi que quinze clitiques, on obtient un total de 43 colonnes. Pour la donnée non catégorielle, le radical, il serait souhaitable d'utiliser un encodage par *deep embedding*, en testant Arabert, version de BERT préentraînée sur l'arabe, Word2Vec et Glove Pennington, Socher et C. D. Manning, 2014 ; ce type d'encodage prend typiquement autour de 300 colonnes. Une segmentation comprend donc environ 350 colonnes.

D'après notre étude, une combinaison de segmenteurs dans un *ensemble* même pondéré n'aurait que peu d'intérêt. En effet, de ce côté, nous n'avons pratiquement pas trouvé de règles qui améliorent la performance dans un domaine sans la dégrader dans un autre, ce qui confirme la difficulté de fabriquer un tel *ensemble*. Aucune combinaison de segmenteur n'est meilleure que Farasa globalement. La prise en compte du nombre de parties de la segmentation, pour exploiter le sur-découpage ou le sous-découpage, s'est également avérée une piste non concluante, sauf dans un cas où la règle suivante apporte un maigre mieux de 0,33% (principalement des particules sous-découpées) :

- Si nombre de parties de Farasa $<$ celles des deux autres
- $\Rightarrow$ choisis la segmentation de Stanford
- sinon prend Farasa

Étant donné que Farasa a très peu tendance à sous-découper les particules (voir tableau 6.36), et en tout cas pas plus que les autres segmenteurs, on peut estimer que cette règle n'est pas assez fiable pour être conservée.

Une piste que nous explorons pour améliorer Farasa est d'utiliser les résultats que nous avons eu dans le lot sélectionné pour les meilleures étiquettes de Stanford. Nous avons développé un programme, appelé Awdeh segmenter dans le site, qui inclut trois segmenteurs spécialisés par POS (noms, verbes, particules), en utilisant les étiquettes de Stanford comme guide.

Le tableau 7.6 compare les résultats de Farasa à ceux du segmenteur Awdeh et à ceux d'une combinaison du segmenteur Awdeh, sur le lot le plus fiable, avec Farasa sur le lot le moins fiable, à la fois globalement et par catégorie (Combi1 dans le tableau).

Une autre piste serait d'explorer la segmentation des clitiques uniquement, pour laquelle nous n'avons pas besoin des sous-catégories. Mais elle sera moins efficace pour pallier aux défauts de Farasa.

Tab. 7.6 : Comparaison entre Farasa & Awdeh et combinaison entre eux

POS	**Gold**	**Farasa**	**Awdeh**	**Combi**
N	17382	74,8%	84,88%	84,44%
P	7689	93,66%	97,7%	97,76%
V	3462	36,01%	66,52%	71,46%
Total	28533	75,17%	86,11%	86,46%

7.2.3 Encodage final

Au minimum, avec une dizaine de caractéristiques pour POS, une segmentation issue de Farasa éventuellement corrigées par le segmenteur spécialisé indiqué ci-dessus, les entrées représentent environ 360 colonnes. Il faut y ajouter le mot, qui à lui tout seul prendra environ 300 colonnes s'il est représenté par deep-embedding, ce qui est certainement le mieux, aboutissant ainsi à un 660 colonnes pour chaque vecteur.

La sortie correspond à une segmentation avec des étiquettes POS, donc autour de 360 colonnes.

7.3 Corpus d'entraînement et données fournies

Notre *gold corpus* est trop petit pour être utilisé comme corpus d'entraînement. D'où l'idée d'utiliser les éléments développés précédemment pour transformer Kalimat en *silver corpus*. Un *silver corpus* est un corpus annoté automatiquement par des procédures dont la fiabilité a été indirectement testée, ce qui est précisément ce que nous avons fait.

Sur notre page *gitlab* (`https://gitlab.com/hussein.awdeh/segmentationgoldarabiccorpus`), nous avons déposé tous les éléments nécessaires à la réalisation de ce segmenteur, sous licence CC by NC, pour rester conforme à la licence couvrant le corpus d'origine avec sa segmentation et ses annotations.

Ce dépôt, destiné aux chercheurs en traitement automatique et en annotation de corpus en arabe, est structuré en deux volets principaux, Gold Corpus et Silver Corpus, ainsi qu'un répertoire, Tools, contenant des outils utilisables pour les deux.

7.3.1 Gold Corpus

Le Gold Corpus contient le corpus de référence annoté manuellement. Il est subdivisé en trois répertoires :

- Original Files, qui contient les 51 fichiers d'origine avec leurs trois formes :
  - les fichiers de départ ;
  - la segmentation Al-Khalil avec ses POS ;
  - les étiquettes Stanford POS.
- Basics Files, qui contient :
  - les tokens du corpus par ordre d'apparition ;
  - les tokens uniques avec leur fréquence ;
  - les mots uniques avec leur fréquence ;
  - le tableau des rangs et des fréquences des mots (pour calculer la courbe de Zipf) ;
  - les mots à plusieurs segmentations dans tous leurs contextes.
- Segmentations, qui contient :
  - les segmentations par Farasa, Madamira, Stanford ;
  - la segmentation manuelle avec les étiquettes de parties du discours ;
  - les étiquettes POS d'Al-Khalil et de Stanford ont été converties.

Le fichier le plus important est *GoldSegmentations.csv*, contenant la liste des occurrences de mots (y compris les nombres), leur identifiant, leur segmentation distribuée dans un vecteur à sept colonnes et leurs étiquettes POS1 et POS2, le tout ayant été réalisé manuellement par mes soins et contrôlé par notre équipe.

Tous les autres fichiers servent à reproduire les expériences décrites dans cette thèse avec l'aide des outils de la section suivante.

7.3.2 Tools

Ce répertoire regroupe les scripts, programmes et données développés pour le gold et/ou le silver corpus. Il s'agit essentiellement de :

- tables de conversion des étiquettes Al-Khalil et Stanford dans notre jeu d'étiquettes ;
- programme de distribution des résultats des segmenteurs dans les colonnes des vecteurs, ce programme n'a pas été réalisé par moi, mais par Mai Khalil, membre de mon équipe ;

- programme d'aide à l'étiquetage et à la segmentation automatiques ;
- programme Awdeh de segmentation sur la base des POS ;
- liste des valeurs possibles pour chacune des colonnes catégorielles (six pour la segmentation et trois pour les étiquettes POS).

7.3.3 Silver Corpus

Le répertoire Silver Corpus contient tous les éléments permettant de transformer Kalimat en *silver corpus*. Étant donné sa taille, tous les fichiers ont été regroupés en six archives *tar.gz*, une par catégorie d'articles. Les noms de fichiers obéissent à la même règle : un numéro d'ordre suivi de la catégorie, puis du nombre d'articles, terminé par le mot *art*. Il contient :

- les trois séries de fichiers originaux : corpus, segmentation Al-Khalil, POS Stanford ;
- les fichiers contenant uniquement les tokens par ordre d'apparition ;
- les segmentations par les trois segmenteurs Farasa, Madamira, Stanford ;
- les vecteurs pour les trois segmentations
- les étiquettes POS converties dans notre jeu à partir des deux étiqueteurs : Al-Khalil, Stanford ;

Avec les tokens, on peut produire les identifiants des vecteurs ; le contenu des vecteurs d'entrée est fourni à partir des vecteurs de segmentation et des étiquettes POS.

7.4 Conclusion

Ce qui précède est ce que nous avons établi pour contribuer à la construction d'un segmenteur à apprentissage séquentiel. À l'origine, nous espérions aller jusqu'à le tester, mais les conditions ne nous l'ont pas permis. Nous avons établi un plan de travail et donné les éléments pour qu'il soit facile aux chercheurs de vérifier nos résultats et de mettre en oeuvre un nouveau segmenteur.

Nous avons dans ce chapitre montré comment réaliser un *silver corpus* d'assez grande taille en segmentant automatiquement et en annotant les segments. Cette étude de faisabilité nous semble concluante, même s'il reste encore beaucoup à

faire. Nous avons en particulier établi que les catégories d'Al-Khalil sont utiles pour effectuer un prétraitement sur les catégories de Stanford, mais qu'il ne faut pas les conserver ensuite ; que ce prétraitement permet indirectement d'améliorer la performance du segmenteur sélectionné et donc la qualité des annotations automatiques du corpus. La fiabilité du segmenteur sélectionné est largement améliorée avec la combinaison de ces prétraitements sur les catégories avec la segmentation de Farasa et notre segmentation catégorielle.

Nos observations nous amènent également à penser qu'il ne serait pas intéressant d'entraîner un segmenteur directement sur l'ensemble des caractéristiques de nos vecteurs. Un meilleur fonctionnement pourrait être obtenu avec un prétraitement du corpus. Nous pensons en particulier aux catégories grammaticales (POS) et à la segmentation des verbes, où nous avons constaté la faiblesse des segmenteurs existants.

Conclusion et perspectives

Le problème traité, qui au départ semblait d'assez bas niveau pour pouvoir être résolu par application d'un outillage moderne, disposant déjà de nombreuses solutions réalisées, s'est avéré beaucoup plus complexe que prévu pour de nombreuses raisons, certaines spécifiques au problème lui-même, comme la spécificité de la langue et de l'écriture arabe, et d'autres concernant les ressources utilisables pour aider à le résoudre, comme le manque de corpus libre et annoté correctement.

Il est ressorti du travail préliminaire que j'ai effectué dans l'état de l'art, avec l'impression de simplement avoir à moderniser des outils relativement anciens, que la très grande majorité des systèmes implémentés se contentaient d'un à peu près qui, nos données l'ont confirmé, fonctionne dans la majorité des cas. Mais pour résoudre le problème dans sa totalité, elles s'avèrent une impasse et il faudra absolument changer d'approche.

On relèvera également qu'il y a un nombre conséquent de découvertes négatives dans cette thèse ; des pistes qui s'avèrent infructueuses, des techniques inadaptées, des conceptualisations faussées. Par exemple, le manque d'attention des chercheurs pour la nature des objets segmentés (clitiques, affixes, radical, racine, etc) me semble avoir porté préjudice à la qualité de leurs résultats. Par contre, au niveau des découvertes positives, il y a les méthodes élaborées pour améliorer la segmentation du corpus Kalimat entier, sans compter les apports que constituent la mise à disposition de la communauté scientifique de tous les éléments et des plus importants outils de nos recherches. Les catégories et la segmentation de ce corpus ont été conséquemment fiabilisés par nos recherches, car la probabilité des solutions trouvées approche au plus bas 90%, et dépasse 95% pour une partie des données. Nous avons donc bien réussi à transformer le corpus Kalimat en *silver corpus*.

En fin de compte l'étude approfondie que j'ai faite du problème m'a permis de dégager des pistes vraiment sérieuses, sortant des sentiers battus, que les chercheurs en informatique linguistique de l'arabe pourront explorer. On considérera deux perspectives à poursuivre en parallèle : d'une part, trouver une méthode pour améliorer la qualité de l'ensemble d'apprentissage en minimisant l'intervention

humaine ; d'autre part, améliorer l'apprentissage lui-même, en particulier en ayant recours à des méthodes de *few shot learning*.

Ce type de méthodes, qui n'existait pas quand j'ai débuté cette thèse, s'est beaucoup développé ces dernières années, ainsi que les méthodes de *transfer learning* ou de *domain adaptation*. Au moment de terminer cette thèse, c'est dans ce contexte que je pense qu'il faudrait reprendre tout ce sujet différemment. Mon espoir est que ce travail permette à ceux qui voudraient le faire de s'épargner d'avoir à parcourir l'état du domaine, et qu'ils pourront trouver ici et sur ma page gitlab la plupart des réponses aux questions préliminaires qu'ils devront se poser, ainsi que de nombreux éléments techniques indispensables à la mise en oeuvre d'un projet de ce type, et beaucoup de données soigneusement contrôlées qui sont le nerf de ces projets.

Annexe A

Mots invariables

Le Tableau A.1 présente ces cinquante nouveaux mots vides et leur signification.

Tab. A.1 : Mots irréguliers

Mots	**English**	**Mots**	**English**
والتالي	And thus	بينما	Why
أخرى	Other	ودائما	And always
لكنها	But	وربما	And perhaps
لولا	Except	وإذا	And if
لأنها	It	ماذا	What
ولذلك	Thereful	بينهم	Including
الان	Now	بأن	That
أنها	It	بعدها	Then
إنها	It	ومنهن	And of Them
التالي	therefore	وكأنه	And like
إلا	But	بمكان	The place
بما	Including	الأحيان	Often
عما	What	فيما	With
وهم	And they	وأنه	And that
كلها	All	وحتى	Even
وإنما	But	معها	With
بعدما	After	ولابد	And must
عندما	When	ويمكن	And can be
أيضا	Also	وحدها	Alone
وأيضا	And also	منهما	Ofthem
ولماذا	And why	بأنهم	They
وهكذا	And so on	بعضها	Some
أحيانا	Sometimes	بينما	Including
وأحيانا	And somtimes	وبينه	And him
وآلما	And whenever	وحدهم	Alone

Tab. A.2 : Mots irréguliers pour Elrajubi stemmer

Mots	**English**	**Stem**	**English**
الله	Allah	الله	Allah
رضوان	Contentment	رضوان	Contentment
لبنان	Lebanon	لبنان	Lebanon
والمانيا	And Germany	المانيا	Germany
مليون	Million	مليون	Million
التلفزيون	The television	تلفزيون	television
والتلفزيون	And the television	تلفزيون	television
فرنسا	France	فرنسا	France
وبريطانيا	And Britain	بريطانيا	Britain
سوريا	Syria	سوريا	Syria
اوروبا	Europe	اوروبا	Europe
اميركا	United States	اميركا	United States
مليونا	Million	مليون	Million
الانترنت	The Internet	انترنت	Internet
للانترنت	For the Internet	انترنت	Internet
والإنترنت	And the Internet	انترنت	Internet
عشرين	Twentieth	عشرين	Twentieth
بخمسين	Fifty	خمسين	Fifty
يوما	Day	يوم	Day
التسعينيات	The Nineties	تسعين	The Nineties
اليابان	The Japan	اليابان	Japan
باريس	Paris	باريس	Paris
بغداد	Baghdad	بغداد	Baghdad
سنوات	Years	سنوات	Years
السعودية	The Saudi Arabia	السعودية	Saudi Arabia
البيانات	The Data	بيانات	Data

Tab. A.3 : Différences entre les versions de Light

Version	Antéposés	Postposés
Light1	ال, وال, بال, كال, فال ,ل	ا,ت
Light2	ajoute و	= Light1
Light3	= Light2	ajoute ة, ه
Light8	= Light2	ajoute ما, ان, ات, ون, ين, يه, ية,ي
Light10	ajoute لل	ajoute نا, هما
CondLight	ajoute ب ,ت , ل , ي , ف, س	ajoute هم , هن, وا

Annexe B

Segmentations multiples du gold corpus

Tab. B.1 : *ta marbuta* (4 cas)

Mot	Segment.	Contexte	Cat.	Glose
الحاجة	ال حاجة	أن الحاجة إليها	NS	[que] le besoin [(qu'on a) d'elle]
	ال حاج ة	أن الحاجة عليّه	NA	[que] la vieille [Aliyya]
مرة	مرة	لأول مرة	NS	[pour la première] fois
	مر ة	يحمل ذكريات مرة	NA	[porte des souvenirs] amers
المعرفة	ال معرفة	في زيادة المعرفة	NS	[dans l'augmentation] de la connaissance
	ال معرف ة	واقتصاديات المعرّفة	NA	[et les économies] définies
الكلية	ال كلية	ان الكلية نجحت	NS	la faculté [a réussi]
	ال كلي ة	المفاهيم الكلية	NA	[les concepts] les globaux

Tab. B.2 : Procl + P (4 cas)

Mot	Segment.	Contexte	Cat.	Glose
فان	فان	يقوم فان دوغ	NS (n. propre)	Van Gogh [fait]
	ف ان	فان هذا كله	procl.+P	En effet (litt. et que) [tout cela]
ولديه	ولد ي ه	مع ولديه	NS + encl.	[avec] ses enfants
	و لدي ه	ولديه ميول	procl.+P+encl.	et avec lui [(sont) des penchants = il a des penchants]
وفي	وفي	عندما وفي	V acc.	(quand il a été) fidèle
	و في	وفي المشهد الاخير	procl.+P	et dans (la scène finale)
فقد	فقد	فقد أبويه	V acc.	il a perdu [ses parents]
	ف قد	فقد سبقتها	procl.+P	et déjà [elle a accepté]

La combinaison proclitique + P est susceptible d'être confondue avec un substantif ou avec un verbe à l'accompli. Les confusions sont principalement dues aux

proclitiques و, ف et ا, qui sont susceptibles de faire partie du radical.

Tab. B.3 : Autres (5 cas)

Mot	**Segment.**	**Contexte**	**Cat.**	**Glose**
عليه	عليه علي ه	أن الحاجة عليّه ووافقت عليه	NS (n. propre) P +encl.	[que la vieille] Aliyya [et elle a approuvé] sur lui
يحيى	يحيى ي حيى	والصديق يحيى أن يحيى	NS (n. propre) préf. + V inacc	[et l'ami] Yahya [être en] vie
أحدث	أحدث أ حدث	أحدث أعماله أحدث عدد صفحاته	NA préf. + V inacc	(ses œuvres) les plus récentes je mets à jour (le nombre de ses pages)
تواصل	تواصل ت واصل	برنامج تواصل التي تواصل تقدمها	NA préf.+V inacc	(programme de) communication qui poursuit (sa progression)
تحول	ت حول تحول	قد تحول إلى قد تحول دون	préf. + V acc V inacc.	[déjà] s'est converti [en qc] [peut-être] elle empêche [qc]

	Mot	Gold	occs	Stanf.	Mad.	Far.	Alkh.
1	الحاجة	ال حاجة	1	+	+		+
		ال حاج ة	1	+		+	
2	مرة	مرة	8	+	+		+
		مر ة	1	+		+	
3	المعرفة	ال معرفة	9	+	+		+
		ال معرف ة	2	+		+	
4	الكلية	ال كلية	8	+	+		+
		ال كلي ة	1	+		+	
5	عليه	عليه	1	+	+		
		علي ه	26	+	+	+	+
6	فان	فان	1	+	+	+	+
		ف ان	15	+		+	
7	يحيى	يحيى	1	+	+	+	+
		ي حيي	1	+			+
8	ولديه	ولد ي ه	1	+			
		و لدي ه	2	+		+	+
9	وفي	وفي	1				
		و في	56	+	+	+	+
10	أحدث	أحدث	1	+	+	+	+
		أ حدث	1	+			
11	تواصل	تواصل	3	+	+	+	+
		ت واصل	1	+			
12	فقد	فقد	32	+	+	+	
		ف قد	1	+			+
13	تحول	ت حول	2	+			
		تحول	3	+	+	+	+
Total		26	180	25	13	14	14

Tab. B.4 : Jeu de tests d'ambiguïtés hors corpus

Mot	Segment.	Contexte	Cat.	Glose
مدينة	مدينة	في وسط مدينة	NS	[au centre-] ville
	مدين ة	وهي مدينة له	NA	[et elle (est)] débitrice [par rapport à lui]
مهمة	مهمة	لها مهمة واضحة	NS	[à elle (est)] une mission [claire]
	مهم ة	العناصر الغذائية مهمة	NA	[l'élément nutritif (est)] important
العقيدة	ال عقيدة	في العقيدة والرأي	NS	[dans] la doctrine [et l'opinion]
	ال عقيد ة	العقيدة مريم	NS/NA	la colonelle (Maryam)
لما	لما	دخلت لما إلى بيتها	NS (n. propre)	Lama [est entrée dans sa maison]
	ل ما	لما يخدم الناس	procl.+P	ce qui (litt. pour cela) [sert les gens]
لنا	لنا	قامت لنا بأعمال	NS (n. propre)	Lana [a fait des travaux]

continué à la page suivante

Mot	Segment.	Contexte	Cat.	Glose
	ل نا	لا بد لنا من ان نجعل	procl.+P	nous [devons faire en sorte que]
لاي	لاي	كان لاي ذاهبا الى عمله	NS (n. propre)	Louay [allait à son travail]
	ل اي	الشأن الحقيقي لاي أمة	procl.+P	[la véritable importance] de toute [
افان	افان	يسمى أفان تتر	NS (n. propre)	[il s'appelle] Avan Tetr
	ا ف ان	افان ذلك	procl.+P	est-ce donc [que cela]
والى	والى	والى الحكم	NS	le gouverneur [du pouvoir]
	و الى	والى اللقاء	procl.+P	et au revoir
وهما	وهم ا	يخشون وهما	NS	(ils craignent) une illusion
	و هما	وهما يرقصان في الشارع	procl.+P	et eux [dansent dans la rue]
وهم	وهم	يعيشون في وهم	NS	[ils vivent dans] une illusion
	و هم	وهم مستعدون للتضحية	procl.+P	et eux [sont prêts à se sacrifier]
الم	الم	هذا الم كبير	NS	[c'est une grande] douleur
	ا لم	الم تخشى	procl.+P	n'as-tu pas [peur]
وقد	و قد	وقد نجح المصورون	procl.+P	et [les photographes ont réussi]
	وقد	وقد الولد النار	V acc.	[le garçon] a allumé [le feu]
لكي	ل كي	وذلك لكي يعرض	procl.+P	[et cela] pour [montrer]
	لك ي	ان الكتاب لكي	P+encl.	[le livre est] pour toi
مهما	مهم ا	كون التعليم مهما بحد ذاته	NA + suff	(l'éducation soit) importante (en so
	مهما	مهما فعلت	P	quoi que (tu fasses)
ياسر	ياسر	الرئيس ياسر عرفات	NS (n. propre)	[le président] Yasser [Arafat]
	ي اسر	ياسر منهم	préf. + V inacc	il capture [certains d'entre eux]
تركتنا	ترك ت نا	لماذا تركتنا في	V acc. + suff. + encl.	[pourquoi] nous as-tu laissés [dedan
	تركت نا	هذه تركتنا	NS + encl.	[ceci (est)] notre part
كما	كما	كما اكدوا	P	Comme (ils l'ont) confirmé
	كم ا	هذا كما كبيرا	NS + suff.	(C'est une grande) quantité
اسرته	اسرت ه	من ان اسرته لن تتقبل هذا الامر	NS+encl.	(que) sa famille (ne l'acceptera pas
	اسر ت ه	وقد اسرته بعينيها	V acc.+suff+encl.	(et) elle l'a capturé (avec ses yeux)
امر	امر	وهذا امر سيء	NS	(et) c'est (mauvais)
	ا مر	امر من هنا	préf. + V inacc	Je passe (par ici)
تحدث	تحدث	وتحدث عن دور الهرمونات	V acc.	(et) il a parlé (du rôle des hormone
	ت حدث	وقد تحدث في اي مكان	préf. + V inacc	(et cela peut) se produire (partout)
وتخلص	و ت خلص	وتخلص آرمسترونغ الى ان	V acc.	(Armstrong) a conclu (que)
	و تخلص	وتخلص من الفتاة	V acc.	il s'est débarrassé (de la fille)
قبلتها	قبل ت ها	ما قبلتها	V inacc.	(Je ne l'ai pas) embrassée
	قبلت ها	واتجهت الى قبلتها	P	(et elle est tournée vers sa) qibla

	Mot	Gold	occs	Stanf.	Mad.	Far.	Alkh.
1	مدينة	مدينة	2	+	+		+
		مدين ة		+		+	
2	مهمة	مهمة	1	+	+		+
		مهم ة		+		+	
3	العقيدة	ال عقيدة	2	+	+		+
		ال عقيد ة		+		+	
4	لما	لما		+	+		+
		ل ما	12	+		+	
5	لنا	لنا		+	+		
		ل نا	13	+	+	+	+
6	لاي	لاي		+	+	+	+
		ل اي	2	+		+	
7	افان	افان		+	+	+	+
		ا ف ان	1	+			+
8	والى	والى		+			
		و الى	3	+		+	+
9	وهما	وهم ا					
		و هما	4	+	+	+	+
10	وهم	وهم		+	+	+	+
		و هم	7	+			
11	الم	الم		+	+	+	+
		ا لم	2	+			
12	وقد	و قد	45	+	+	+	
		وقد		+			+
13	لكي	ل كي	4	+			
		لك ي		+	+	+	+
14	مهما	مهم ا		+			
		مهما	2	+	+	+	+
15	ياسر	ياسر	3	+			
		ي اسر		+	+	+	+
16	تركتنا	ترك ت نا	2	+			
		تركت نا		+	+	+	+
17	كما	كما	69	+			
		كم ا		+	+	+	+
18	اسرته	اسرت ه	2	+			
		اسر ت ه		+	+	+	+
19	امر	امر	3	+			
		ا مر		+	+	+	+
20	تحدث	تحدث		+			
		ت حدث	2	+	+	+	+
21	وتخلص	و ت خلص	1	+			
		و تخلص		+	+	+	+
22	قبلتها	قبل ت ها	1	+			
		قبلت ها		+	+	+	+
Total		42	180	25	13	14	14

Mes publications

[5] A. Abdallah, H. Awdeh, Y. Zaki, G. Bernard et M. Hajjar. « Unsupervised Grammatical Pattern Discovery from Arabic Extra Large Corpora ». In : *Proceedings of the 13th International Joint Conference on Computational Intelligence*. SCITEPRESS - Science et Technology Publications, 2021. doi : 10.5220/0010651700003063 (cf. p. 12).

[39] H. Awdeh, A. Abdallah, G. Bernard et M. Hajjar. « A Gold Multipurpose Arabic Corpus (GAC) ». In : *Proceedings of the International Conference on Electrical, Computer and Energy Technologies (ICECET)* (2021), p. 1-5 (cf. p. 14, 158).

[40] H. Awdeh, A. Abdallah, G. Bernard et M. Hajjar. « A Silver Standard Arabic Corpus for Segmentation and Validation SAC ». In : *Proceedings of the international conference on Big Data and Cyber Security BDCSIntell'2019* (2021) (cf. p. 14).

[41] H. Awdeh. *Gold Corpus for Arabic segmentation*. 2025. url : https://gitlab.com/hussein.awdeh/segmentationgoldarabiccorpus/ (cf. p. 14).

[42] H. Awdeh, A. Abdallah, G. Bernard et M. Hajjar. « Overview of Arabic Sentence Corpora ». In : *Proceedings of the 13th International Joint Conference on Computational Intelligence*. SCITEPRESS - Science et Technology Publications, 2021. doi : 10.5220/0010651200003063 (cf. p. 14).

[43] H. Awdeh, M. Khalil, O. Saadiyeh, G. Bernard, M. Hajjar et A. Ramadan. « Methodology for building a Silver Arabic Corpus for segmentation ». In : *35th International Conference on Computer Theory and Applications*. Accepted, to appear. 2025 (cf. p. 14).

[44] H. Awdeh, O. Saadiyeh, M. Khalil, G. Bernard, M. Hajjar et A. Ramadan. « Survey of Arabic Word Segmentation : Evaluation and Comparative Study ». In : *Proceedings of the 2025 IEEE International Conference on Emerging Trends in Engineering and Computing (ETECOM) – Artificial Intelligence, Machine Learning, and Business Analytics*. Accepted, to appear. 2025 (cf. p. 14, 186).

Bibliographie

[1] M. Abbas et K. Smaïli. « Comparison of Topic Identification Methods for Arabic Language ». In : *Proceedings of the International Conference on Recent Advances in Natural Language Processing, (RANLP.)* (2005) (cf. p. 49).

[2] M. Abbas, K. Smaïli et D. Berkani. « Evaluation of Topic Identification Methods on Arabic Corpora ». In : *Journal of Digital Information Management* 9.5 (oct. 2011), p. 185-192 (cf. p. 49).

[3] M. Abbas, K. Smaïli et D. Berkani. « Multi-category support vector machines for identifying Arabic topics ». In : *Conference on Intelligent Text Processing and Computational Linguistics*. 2009. url : https://api.semanticscholar.org/CorpusID:61723878 (cf. p. 49).

[4] A. Abdallah. « Catégorisation sémantique et information grammaticale en arabe ». Thèse de doct. Université Paris 8, 2021 (cf. p. 11).

[5] A. Abdallah, H. Awdeh, Y. Zaki, G. Bernard et M. Hajjar. « Unsupervised Grammatical Pattern Discovery from Arabic Extra Large Corpora ». In : *Proceedings of the 13th International Joint Conference on Computational Intelligence*. SCITEPRESS - Science et Technology Publications, 2021. doi : 10.5220/0010651700003063 (cf. p. 12).

[6] A. Abdelali. « Adjir Corpora ». In : (2005). url : http://aracorpus.e3rab.com (cf. p. 50).

[7] A. Abdelali, J. Cowie et H. Soliman. « Arabic Modern Standard Corpus ». In : *Proceedings of the workshop on computational modeling of lexical acquisition* (juill. 2005) (cf. p. 50).

[8] A. Abdelali, K. Darwish, N. Durrani et H. Mubarak. « Farasa : A Fast and Furious Segmenter for Arabic ». In : *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Demonstrations*. Association for Computational Linguistics, 2016. doi : 10.18653/v1/n16-3003 (cf. p. 130, 149).

[9] A. Abdulhay. « Constitution d'une ressource sémantique arabe à partir d'un corpus multilingue aligné ». Thèse de doct. Université de Grenoble, 2012 (cf. p. 23).

[10] I. Abu El-Khair. « Abu El-Khair Corpus : A Modern Standard Arabic Corpus ». In : *International Journal of Recent Trends in Engineering and Research* 2 (nov. 2016), p. 5-13 (cf. p. 48, 52).

[11] I. AbuEl-Khair. « Arabic information retrieval ». In : *Annual review of information science and technology* 41 (2007), p. 505-533 (cf. p. 48, 54).

[12] R. A. Abumalloh, H. M. Al-Sarhan, O. Ibrahim et W. Abu-Ulbeh. « Arabic Part-of-Speech Tagging ». In : *Journal of Soft Computing and Decision Support Systems* 3.2 (2016). url : https://www.jscdss.com/index.php/files/article/view/94 (cf. p. 144).

[13] S. Afonso, V. Moreira et D. Santos. « TreeTagger and the Floresta Sintá (c)tica : An Experimental Evaluation ». In : *Proceedings of the V International Conference on Language Resources and Evaluation*. 2006 (cf. p. 73).

[14] A. E. S. Al Hajjar. « Extraction et gestion de l'information à partir des documents arabes ». Thèse de doct. Paris 8 University, 2010 (cf. p. 27, 30, 245).

[15] S. Alansary et M. Nagi. « The International Corpus of Arabic ICA ». In : *The International Corpus of Arabic : Compilation, Analysis and Evaluation. ANLP* (2014) (cf. p. 61).

[16] S. Alansary. « BAMAE : Buckwalter Arabic Morphological Analyzer Enhance ». In : *Proceedings of the 4th International Conference on Arabic Language Processing*. https://kipdf.com/bamae-buckwalter-arabic-morphological-analyzer-enhancer_5ad9c15c7f8b9a53638b459b.html. Mohamed V University Souissi. Rabat, Morocco, 2012 (cf. p. 123).

[17] S. Alansary. « BibAlex Standard Modern Arabic Morphological Analyzer (BASMA) ». In : *Egyptian Journal of Language Engineering* 6.1 (2019). Accessed : 2025-08-06, p. 1-15. url : https://ejle.journals.ekb.eg/article_60166_6ba513190f97c69f2b5bb89d407cf15a.pdf (cf. p. 123, 124, 245).

[18] F. Albogamy et A. Ramsay. « Fast and robust POS tagger for Arabic tweets using agreement-based bootstrapping ». In : *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*. Portorož, Slovenia, 2016, p. 1500-1506 (cf. p. 152).

[19] A. Alfaidi, H. Alwadei, A. Alshutayri et S. Alahdal. « Exploring the Performance of Farasa and CAMeL Taggers for Arabic Dialect Tweets ». In : *The International Arab Journal of Information Technology* 20.3 (2023), p. 355-362. url : https://iajit.org/paper/4809 (cf. p. 130).

[20] R. Alharbi, W. Magdy, K. Darwish, A. AbdelAli et H. Mubarak. « Part-of-speech tagging for Arabic Gulf dialect using Bi-LSTM ». In : *International Conference on Language Resources and Evaluation (LREC)*. 2019, p. 3925-3932 (cf. p. 143, 150).

[21] A. Alhasan et A. T. Al-Taani. *POS Tagging for Arabic Text Using Bee Colony Algorithm*. Manuscript posted on ResearchGate. Consulté en septembre 2025. 2018. url : https://www.researchgate.net/publication/328963130_POS_Tagging_for_Arabic_Text_Using_Bee_Colony_Algorithm (cf. p. 150).

[22] M. Aljlayl et O. Frieder. « Lemmatization of irregular plurals without dictionary for Arabic language ». In : *Journal of King Saud University-Computer and Information Sciences* 28.4 (2016), p. 426-436 (cf. p. 83).

[23] M. Aljlayl et O. Frieder. « On Arabic Search : Improving the retrieval effectiveness via a light stemming approach ». In : *Proceedings of the eleventh international conference on Information and knowledge management*. CIKM02. ACM, nov. 2002. doi : 10.1145/584792.584848 (cf. p. 83).

[24] E. Alkhalifa, M. Al-Saydi et H. S. Al-Khalifa. « Sarf : Fast and Application Customizable Arabic Morphological Analyzer ». In : *Natural Language Engineering* 20.4 (2014), p. 463-488 (cf. p. 105, 107, 245).

[25] R. Alluhaibi, T. Alfraidi, M. Abdeen et A. Yatimi. « A Comparative Study of Arabic Part of Speech Taggers Using Literary Text Samples from Saudi Novels ». In : *Information* 12 (déc. 2021). doi : 10.3390/info12120523 (cf. p. 142).

[26] K. Almeman et M. Lee. « Automatic building of Arabic multi dialect text corpora by bootstrapping dialect words ». In : *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. IEEE, fév. 2013, p. 1-6. doi : 10.1109/iccspa.2013.6487247 (cf. p. 45).

[27] F. Alotaiby, S. Foda et I. Alkharashi. « Clitics in Arabic Language : A Statistical Study ». In : *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*. Sous la dir. de R. Otoguro, K. Ishikawa, H. Umemoto, K. Yoshimoto et Y. Harada. Tohoku University, Sendai, Japan : Institute of Digital Enhancement of Cognitive Processing,

Waseda University, nov. 2010, p. 595-601. url : https://aclanthology.org/Y10-1068/ (cf. p. 30).

[28] C. Aloulou, L. Hadrich Belguith et A. Ben Hamadou. « MASPAR : A multi-agent system for parsing arabic ». In : *IEEE International Conference on Systems, Man and Cybernetics*. ICSMC-02. IEEE, 200. doi : 10.1109/icsmc.2002.1175732 (cf. p. 97).

[29] M. Alrabiah, A. Salman et E. Atwell. « King Saud University Corpus of Classical Arabic (KSUCCA) ». In : *Department of Computer Science, King Saud University* (2014) (cf. p. 57).

[30] A. Alsifayani. « المعالجة الآلية للغة العربية (التحليل الصرفي) ». In : المعالجة الآلية للغة العربية. King Fahd University of Petroleum & Minerals Information et Computer Science. 2008 (cf. p. 33).

[31] R. Alur, S. Chaudhuri et P. Madhusudan. « Languages of Nested Trees ». In : *Int. Conf. on Computer Aided Verification*. Seattle, WA, USA, 2006, p. 329-342 (cf. p. 108).

[32] L. Amador et O. Castillo. « Bee Colony Optimization Algorithm ». In : *Optimization of Type-2 Fuzzy Controllers Using the Bee Colony Algorithm*. Accessed : 2025-08-09. Springer International Publishing, 2017, p. 23-32. isbn : 9783319542959. doi : 10.1007/978-3-319-54295-9_4. url : https://link.springer.com/chapter/10.1007/978-3-319-54295-9_4 (cf. p. 150).

[33] H. Al-Ameed. « A proposed new model using a light stemmer for increasing the success of search in Arabic terms ». Thèse de doct. Bradford, UK : University of Bradford, 2006 (cf. p. 94).

[34] A. Anwar, G. I. Salama et M. B. Abdelhalim. « Video Classification and Retrieval Using Arabic Closed Caption ». In : *The 6th International Conference on Information Technology (ICIT)*. Arab Academy for Science, Technology & Maritime Transport. 2013 (cf. p. 87).

[35] J.-i. Aoe. « An efficient digital search algorithm by using a double-array structure ». In : *IEEE Transactions on Software Engineering* 15.9 (1989), p. 1066-1077 (cf. p. 106).

[36] M. Attia. « A Large-Scale Computational Processor of the Arabic Morphology, and Applications ». Mém. de mast. Cairo, Egypt : Faculty of Engineering, Cairo University, 2000 (cf. p. 80).

[37] M. Attia, P. Pecina, A. Toral, L. Tounsi et J. van Genabith. « An open-source finite state morphological transducer for modern standard Arabic ». In : *Workshop on Finite State Methods and Natural Language Processing*. 2011, p. 125-133 (cf. p. 131).

[38] M. Attia, M. Yaseen et K. Choukri. *Specifications of the Arabic Written Corpus produced within the NEMLAR project*. 2005. url : www.nemlar.org (cf. p. 58).

[39] H. Awdeh, A. Abdallah, G. Bernard et M. Hajjar. « A Gold Multipurpose Arabic Corpus (GAC) ». In : *Proceedings of the International Conference on Electrical, Computer and Energy Technologies (ICECET)* (2021), p. 1-5 (cf. p. 14, 158).

[40] H. Awdeh, A. Abdallah, G. Bernard et M. Hajjar. « A Silver Standard Arabic Corpus for Segmentation and Validation SAC ». In : *Proceedings of the international conference on Big Data and Cyber Security BDCSIntell'2019* (2021) (cf. p. 14).

[41] H. Awdeh. *Gold Corpus for Arabic segmentation*. 2025. url : https://gitlab.com/hussein.awdeh/segmentationgoldarabiccorpus/ (cf. p. 14).

[42] H. Awdeh, A. Abdallah, G. Bernard et M. Hajjar. « Overview of Arabic Sentence Corpora ». In : *Proceedings of the 13th International Joint Conference on Computational Intelligence*. SCITEPRESS - Science et Technology Publications, 2021. doi : 10.5220/0010651200003063 (cf. p. 14).

[43] H. Awdeh, M. Khalil, O. Saadiyeh, G. Bernard, M. Hajjar et A. Ramadan. « Methodology for building a Silver Arabic Corpus for segmentation ». In : *35th International Conference on Computer Theory and Applications*. Accepted, to appear. 2025 (cf. p. 14).

[44] H. Awdeh, O. Saadiyeh, M. Khalil, G. Bernard, M. Hajjar et A. Ramadan. « Survey of Arabic Word Segmentation : Evaluation and Comparative Study ». In : *Proceedings of the 2025 IEEE International Conference on Emerging Trends in Engineering and Computing (ETECOM) – Artificial Intelligence, Machine Learning, and Business Analytics*. Accepted, to appear. 2025 (cf. p. 14, 186).

[45] M. Badr et al. « CAMeL Tools : An Open-Source Toolkit for Arabic Natural Language Processing ». In : 2022. url : https://github.com/CAMeL-Lab/camel_tools (cf. p. 151).

[46] K. R. Beesley. « Arabic Finite-State Morphological Analysis and Generation ». In : *Proceedings of the 16th conference on Computational linguistics* -. Association for Computational Linguistics, 1996. doi : 10.3115/992628.992647 (cf. p. 76).

[47] L. H. Belguith, L. Baccour et G. Mourad. « Segmentation de textes arabes basée sur l'analyse contextuelle des signes de ponctuations et de certaines particules ». In : *Actes de la 12ème Conférence annuelle sur le Traitement Automatique des Langues Naturelles (TALN 2005)*. 2005, p. 451-456 (cf. p. 97).

[48] L. Belguith Hadrich. « Traitement des erreurs d'accord de l'Arabe basé sur une analyse syntagmatique étendue pour la vérification et une analyse multicritères pour la correction ». Thèse de doct. Faculté des Sciences de Tunis, 1999 (cf. p. 97).

[49] A. Ben Hamadou. « Vérification et correction automatique par analyse affixale des textes écrits en langage naturel : le cas de l'Arabe non voyellé ». Thèse de doct. Faculté des Sciences de Tunis, 1993 (cf. p. 97).

[50] A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M. Chen et J. Seiferas. « The Smallest Automaton Recognizing the Subwords of a Text ». In : *Theoretical Computer Science* 40.6 (1985), p. 31-55. issn : 0304-3975. doi : 10.1016/0304-3975(85)90157-4 (cf. p. 73).

[51] M. Boudchiche, A. Mazroui, M. Ould Abdallahi Ould Bebah, A. Lakhouaja et A. Boudlal. « AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer ». In : *Journal of King Saud University - Computer and Information Sciences* 29.2 (avr. 2017). Arabic Natural Language Processing : Models, Systems and Applications, p. 141-146. issn : 1319-1578. doi : 10.1016/j.jksuci.2016.05.002. url : https://www.sciencedirect.com/science/article/pii/S131915781630026X (cf. p. 104, 191).

[52] A. Boudlal, A. Lakhouaja, A. Mazroui, A. Meziane, O. A. O. B. M. et M. Shoul. « Alkhalil Morpho SYS1: a Morphosyntactic Analysis System for Arabic Texts ». In : *International Arab Conference on Information Technology*. Benghazi, Libya, 2010, p. 16 (cf. p. 104).

[53] K. Bouzoubaa, Y. Jaafar, D. Namly, R. Tachicart, R. Tajmout, H. Khamar, H. Jaafar, L. Aouragh et A. Yousfi. « A Description and Demonstration of SAFAR Framework ». In : *Proceedings of the EACL Workshop*. 2021 (cf. p. 125).

[54] T. Brants. « TnT – A Statistical Part-of-Speech Tagger ». In : *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP)*. 2000, p. 224-231 (cf. p. 142).

[55] E. Brill. « Rule-Based Part of Speech ». In : *Proceedings of the third Conference on Applied Natural Language Processing (ANLC)*. 1992, p. 152-155 (cf. p. 142).

[56] E. Brill. « Transformation-based error-driven learning and natural language processing : A case study in part-of-speech tagging ». In : *Computational linguistics* 21.4 (1995), p. 543-565 (cf. p. 142).

[57] S. Brin et L. Page. « The anatomy of a large-scale hypertextual web search engine ». In : *Proceedings of the seventh international conference on World Wide Web*. 1998, p. 107-117 (cf. p. 118).

[58] T. Buckwalter. « Issues in Arabic Orthography and Morphology Analysis ». In : *COLING 2004 Computational Approaches to Arabic Script-based Languages*. Geneva, 2004, p. 31-34 (cf. p. 59).

[59] T. Buckwalter et J. Parker. « The buckwalter arabic morphological analyzer ». In : *Proceedings of the Workshop on Computational Approaches to Semitic Languages : Common Issues and Resources*. 2002, p. 1-8 (cf. p. 84).

[60] Buckwalter, Tim. *Buckwalter Arabic Morphological Analyzer Version 2.0*. 2004. doi : 10.35111/050Q-5R95 (cf. p. 84).

[61] L. Burnard et S. Bauman. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. TEI Consortium. 2008 (cf. p. 125).

[62] N. K. Chaâben, L. Hadrich Belguith et A. Ben Hamadou. « The MORPH2 new version : A robust morphological analyzer for Arabic texts ». In : *Proceedings of 10th International Conference Journées d'Analyse statistique des Données Textuelles*. 2010 (cf. p. 96-98, 245).

[63] A. Chelli. *Assem's Arabic Stemmer*. https://figshare.com/. 2018. doi : 10.6084/M9.FIGSHARE.7295690.V1 (cf. p. 111).

[64] A. Chen et F. Gey. « Building an Arabic stemmer for information retrieval ». In : *Proceedings of TREC*. Gaithersburg, Maryland, 2002 (cf. p. 118).

[65] S. Chen, Y. Xu et H. Chang. « A Simple and Effective Unsupervised Word Segmentation Approach ». In : *AAAI*. Sous la dir. de W. Burgard et D. Roth. AAAI Press, 2011. url : http://dblp.uni-trier.de/db/conf/aaai/aaai2011.html#ChenXC11 (cf. p. 18).

[66] K. Cho, B. van Merrienboer, D. Bahdanau et Y. Bengio. « On the Properties of Neural Machine Translation : Encoder–Decoder Approaches ». In : *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, 2014. doi : DOI:10.3115/v1/W14-4012 (cf. p. 137).

[67] J. Chung, C. Gulcehre, K. Cho et Y. Bengio. « Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling ». In : *arXiv preprint arXiv :1412.3555* (2014). Comparaison GRU vs LSTM. url : https://arxiv.org/abs/1412.3555 (cf. p. 138).

[68] J. Civera, P. Martínez-Barco et A. Garcia-Serrano. « FreeLing 2.0: Five years of open-source language processing tools ». In : *International Journal of Computational Linguistics and Applications* 2.1 (2011), p. 95-104 (cf. p. 73).

[69] S. Corston-Oliver, A. Aue, K. Duh et E. Ringger. « Multilingual Dependency Parsing using Bayes Point Machines ». In : *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA : Association for Computational Linguistics, 2006, p. 160-167. url : https://aclanthology.org/N06-1021/ (cf. p. 61).

[70] M. Creutz et al. « Morphological segmentation is an important sub-task in many natural language processing (NLP) applications, aiming to break words into meaning-bearing sub-word units called morphemes ». In : (2007) (cf. p. 18).

[71] K. Darwish. « Building a Shallow Arabic Morphological Analyzer in One Day ». In : *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages* -. Association for Computational Linguistics, 2002. doi : 10.3115/1118637.1118643 (cf. p. 86, 94, 120).

[72] K. Darwish. « Probabilistic Methods for Searching OCR-Degraded Arabic Text ». Thèse de doct. Maryland, United States : University of Maryland, College Park, 2003 (cf. p. 86).

[73] K. Darwish. « Transcript-based morphological analysis for Arabic text ». In : *Arabic computational morphology : knowledge-based and empirical methods* 9.1 (2002), p. 85-96 (cf. p. 120).

[74] K. Darwish, A. Abdelali et H. Mubarak. « Using stem-templates to improve Arabic POS and gender/number tagging ». In : *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. 2014 (cf. p. 131).

[75] K. Darwish, W. Magdy et A. Mourad. « Language processing for Arabic microblog retrieval ». In : *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM-2012)*. ACM. 2012, p. 2427-2430 (cf. p. 131).

[76] K. Darwish et D. W. Oard. « CLIR Experiments at Maryland for TREC-2002: Evidence combination for Arabic-English Retrieval ». In : *Proceedings of TREC*. Gaithersburg, Maryland, fév. 2003. doi : 10.21236/ada452814 (cf. p. 120).

[77] J. Devlin, M. W. Chang, K. Lee et K. Toutanova. « BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding ». In : *Proceedings of NAACL-HLT 2019*. 2019, p. 4171-4186 (cf. p. 143).

[78] M. Diab. « Second generation AMIRA tools for Arabic processing : Fast and robust tokenization, POS tagging, and base phrase chunking ». In : *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools*. T. 110. Cairo, Egypt, 2010, p. 198 (cf. p. 130, 145).

[79] M. Diab, N. Habash, O. Rambow et R. Roth. *LDC Arabic Treebanks and Associated Corpora : Data Divisions Manual*. Technical Report CCLS-13-02. Center for Computational Learning Systems, Columbia University, 2013 (cf. p. 130).

[80] M. Diab, K. Hacioglu et D. Jurafsky. « Automatic Tagging of Arabic Text : From Raw Text to Base Phrase Chunks ». In : *Proceedings of HLT-NAACL 2004: Short Papers*. Boston, Massachusetts, USA : Association for Computational Linguistics, mai 2004, p. 149-152. url : https://www.aclweb.org/anthology/N04-4038 (cf. p. 145, 146).

[81] M. El Abdi, B. S. Ben Ali et S. Ben Yahia. « A New Approach of Morphological Analysis of Arabic Syntagmatic Units Based on a Linguistic Ontology ». In : *Computational Collective Intelligence - 14th International Conference, ICCCI 2022, Hammamet, Tunisia, September 28-30, 2022, Proceedings*. Sous la dir. de N. T. Nguyen, Y. Manolopoulos, R. Chbeir, A. Kozierkiewicz et B. Trawinski. T. 13501. Lecture Notes in Computer Science. Springer, 2022, p. 364-377. doi : 10.1007/978-3-031-16014-1_29. url : https://doi.org/10.1007/978-3-031-16014-1%5C_29 (cf. p. 112-115, 245, 247).

[82] O. M. Elrajubi. « An improved Arabic light stemmer ». In : *2013 International Conference on Research and Innovation in Information Systems (ICRIIS)*. IEEE, nov. 2013, p. 33-38. isbn : 978-1-4799-2486-8. doi : 10.1109/icriis.2013.6716682 (cf. p. 67, 88).

[83] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang et C.-J. Lin. « LIBLINEAR : A Library for Large Linear Classification ». In : *Journal of Machine Learning Research* 9 (2008), p. 1871-1874 (cf. p. 129).

[84] S. Fedaghi et F. Al-Anzi. « A new algorithm to extract Arabic root-pattern forms ». In : *The 11th national computer conference, Saudi Arabia*. 1989 (cf. p. 77).

[85] G. D. Forney. « The Viterbi Algorithm ». In : *Proceedings of the IEEE*. T. 61. 3. Analyse théorique et applications en télécommunications. 1973, p. 268-278. doi : 10.1109/PROC.1973.9030 (cf. p. 128).

[86] A. A. Freihat, G. Bella, H. Mubarak et F. Giunchiglia. « A single-model approach for Arabic segmentation, POS tagging, and named entity recognition ». In : *Proceedings of the 2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP)*. [Google Scholar]. Algiers, Algeria : IEEE, avr. 2018, p. 1-8. doi : 10.1109/icnlsp.2018.8374393 (cf. p. 152).

[87] A. A. Freihat, G. Bella, M. Abbas, H. Mubarak et F. Giunchiglia. « ALP : An Arabic Linguistic Pipeline ». English. In : *Analysis and Application of Natural Language and Speech Processing*. Signals and Communication Technology. Publisher Copyright : © 2023, The Author(s), under exclusive license to Springer Nature Switzerland AG. Germany : Springer International Publishing, août 2023, p. 67-99. isbn : 9783031110351. doi : 10.1007/978-3-031-11035-1_4 (cf. p. 152).

[88] D. Ghoul. « Développement de ressources pour l'entraînement et l'utilisation de l'étiqueteur morphosyntaxique TreeTagger sur l'arabe ». In : *Actes de la conférence RECITAL 2013*. Les Sables d'Olonne, France, 2013. url : https://aclanthology.org/F13-5006/ (cf. p. 148).

[89] J. Gimenez et L. Marquez. « Fast and accurate part-of-speech tagging : The SVM approach revisited ». In : *Proceedings of RANLP*. 2003, p. 158-165 (cf. p. 143).

[90] D. Goldhahn, T. Eckart et U. Quasthoff. « Building Large Monolingual Dictionaries at the Leipzig Corpora Collection : From 100 to 200 Languages ». In : *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*. 2012, p. 759-765 (cf. p. 57).

[91] I. Goodfellow, Y. Bengio et A. Courville. *Deep Learning*. Chapitre sur les RNN et GRU. MIT Press, 2016. isbn : 9780262035613. url : https://www.deeplearningbook.org (cf. p. 135).

[92] A. Goweder, M. Poesio, A. De Roeck et J. Reynolds. « Identifying Broken Plurals in Unvowelised Arabic Text ». In : *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Sous la dir. de D. Lin et D. Wu. Barcelona, Spain : Association for Computational Linguistics, juill. 2004, p. 246-253. url : https://aclanthology.org/W04-3232 (cf. p. 91, 92, 245).

[93] D. Graff et K. Walker. « LDC Corpus ». In : *Proceedings of the Arabic newswire part 1. Linguistic Data Consortium* LDC2001T55 (2001). url : https://catalog.ldc.upenn.edu/LDC2001T55 (cf. p. 54).

[94] D. Graff, J. Kong, K. Chen et K. Maeda. « Arabic Gigaword ». In : *Linguistic Data Consortium, Philadelphia* 4.1 (2003), p. 34 (cf. p. 55).

[95] M. Graja, M. Jaoua et L. Hadrich-Belguith. « Lexical Study of A Spoken Dialogue Corpus in Tunisian Dialect ». In : (2010) (cf. p. 45).

[96] S. Green et J. DeNero. « A Class-Based Agreement Model for Generating Accurately Inflected Translations ». In : *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea : Association for Computational Linguistics, juill. 2012, p. 146-155. url : https://www.aclweb.org/anthology/P12-1016 (cf. p. 18).

[97] L. V. Guilder. *Automated Part of Speech Tagging : A Brief Overview*. Handout for LING361, Fall 1995, Georgetown University. 1995 (cf. p. 142).

[98] N. Habash. « Arabic Morphological Representations for Machine Translation ». In : *Arabic Computational Morphology : Knowledge-based and Empirical Methods*. Sous la dir. d'A. Soudi, A. v. d. Bosch et G. Neumann. Dordrecht : Springer Netherlands, 2007, p. 263-285. isbn : 978-1-4020-6046-5. doi : 10.1007/978-1-4020-6046-5_14. url : https://doi.org/10.1007/978-1-4020-6046-5_14 (cf. p. 86, 129).

[99] N. Habash. « Arabic Morphology and Phonology Based on the Lexicon ». Thèse de doct. University of Maryland, College Park, 2005 (cf. p. 18).

[100] N. Habash, R. Gabbard, O. Rambow, S. Kulick et M. Marcus. « Determining Case in Arabic : Learning Complex Linguistic Behavior Requires Complex Linguistic Features ». In : *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 2007, p. 1084-1092 (cf. p. 59).

[101] N. Habash, R. M. Marzouk, C. Khairallah et S. Khalifa. « Morphotactic Modeling in an Open-source Multi-dialectal Arabic Morphological Analyzer and Generator ». In : *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, 2022. doi : 10.18653/v1/2022.sigmorphon-1.10. url : https://api.semanticscholar.org/CorpusID:250390677 (cf. p. 12, 73).

[102] N. Habash et O. Rambow. « Arabic Diacritization through Full Morphological Tagging ». In : *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics ; Companion Volume, Short Papers*. Rochester, New York : Association for Computational Linguistics, avr. 2007, p. 53-56 (cf. p. 59).

[103] N. Habash et O. Rambow. « Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop ». In : *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan, 2005, p. 573-580 (cf. p. 59).

[104] N. Habash et O. Rambow. « Extracting a Tree Adjoining Grammar from the Penn Arabic Treebank ». In : *JEP-TALN 2004, Session Traitement Automatique de l'Arabe*. Fes, Morocco, avr. 2004 (cf. p. 59).

[105] N. Habash, O. Rambow et R. Roth. « MADA+TOKAN : A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization ». In : *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR)* (jan. 2009) (cf. p. 129).

[106] K. Hacioglu et W. Ward. « Target word detection and semantic role chunking using support vector machines ». In : *Proceedings of HLT-NAACL*. 2003, p. 25-27 (cf. p. 145).

[107] L. Hadrich Belguith et N. Chaâben. « Analyse et désambigüisation morphologiques de textes arabes non voyellés ». French. In : *Actes de la 13ème conférence sur le Traitement Automatique des Langues Naturelles. Posters*. Sous la dir. de P. Mertens, C. Fairon, A. Dister et P. Watrin. Leuven, Belgique : ATALA, avr. 2006, p. 493-501. url : https://aclanthology.org/2006.jeptalnrecital-poster.13 (cf. p. 96).

[108] M. El-Haj et R. Koulali. « KALIMAT a multipurpose Arabic Corpus ». In : (2013), p. 22-25 (cf. p. 56, 162).

[109] J. Hajič, E. Hajičová, J. Panevová, P. Sgall, P. Pajas, J. Štěpánek, J. Havelka et M. Mikulová. « Prague Dependency Treebank 2.0 ». In : *LDC Catalog No.: LDC2006T01*. Linguistic Data Consortium. 2006 (cf. p. 60, 61).

[110] J. Hajič, O. Smrž, T. Buckwalter et H. Jin. « Feature-Based Tagger of Approximations of Functional Arabic Morphology ». In : *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*. Barcelona, Spain, 2005, p. 53-64 (cf. p. 59, 61).

[111] J. Hajič, O. Smrž, P. Zemánek, P. Pajas, J. Šnaidauf, E. Beška, J. Kráčmar et K. Hassanová. « Prague Arabic Dependency Treebank 1.0 ». In : *LDC2004T23*. Linguistic Data Consortium, 2004. url : https://catalog.ldc.upenn.edu/LDC2004T23 (cf. p. 60).

[112] J. Hajič et P. Zemánek. « Prague Arabic dependency treebank : Development in data and tools ». In : *In Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*. 2004, p. 110-117 (cf. p. 61).

[113] E. Hajičová et P. Sgall. « Dependency Syntax in Functional Generative Description ». In : *Dependenz und Valenz–Dependency and Valency, volume I*. Walter de Gruyter. 2003, p. 570-592 (cf. p. 60).

[114] A. Hakkoum, H. Kharrazi et S. Raghay. « A Portable Natural Language Interface to Arabic Ontologies ». In : *International Journal of Advanced Computer Science and Applications* 9.3 (2018) (cf. p. 113).

[115] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann et I. H. Witten. « The WEKA Data Mining Software : An Update ». In : *ACM SIGKDD Explorations Newsletter* 11.1 (nov. 2009), p. 10-18. issn : 1931-0153. doi : 10.1145/1656274.1656278 (cf. p. 78, 86).

[116] B. Hammo, F. Al-Shargi, S. Yagi et N. Obeid. « Arabic Corpus UJAC ». In : *Proceedings of the Second Workshop on Arabic Corpus Linguistics (WACL-2)* (2013) (cf. p. 51).

[117] X. Han, Y. Sun et S. Liu. « Chinese Natural Language Processing at Scale with the Online-Debugging Open-Source HanLP ». In : *Proceedings of the 57th Conference of the Association for Computational Linguistics : System Demonstrations*. 2019, p. 37-42 (cf. p. 73).

[118] Z. S. Harris. « Mathematical structures of language ». In : *Interscience tracts in pure and applied mathematics*. 1968 (cf. p. 68).

[119] A. Hasnah. « Al-Raya Corpus, Full Text Processing and Retrieval : Weight Ranking, Text Structuring, and Passage Retrieval for Arabic Documents ». Thèse de doct. Illinois Institute of Technology, 1996 (cf. p. 48).

[120] I. Hmeidi, G. Kanaan et M. Evens. « Design and implementation of automatic indexing for information retrieval with Arabic documents ». en. In : *Proceedings of the JASIS* 48.10 (oct. 1997), p. 867-881 (cf. p. 47).

[121] I. I. Hmeidi, R. F. Al-Shalabi, A. T. Al-Taani, H. Najadat et S. A. Al-Hazaimeh. « A novel approach to the extraction of roots from Arabic words using bigrams ». In : *Journal of the American Society for Information Science and Technology* 61.3 (déc. 2009), p. 583-591. issn : 1532-2890. doi : 10.1002/asi.21247 (cf. p. 100).

[122] S. Hochreiter et J. Schmidhuber. « Long Short-Term Memory ». In : *Neural Computation* 9.8 (nov. 1997), p. 1735-1780. issn : 1530-888X. doi : 10.1162/neco.1997.9.8.1735. eprint : https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. url : https://doi.org/10.1162/neco.1997.9.8.1735 (cf. p. 135).

[123] P. Hudak. *The Haskell School of Expression : Learning Functional Programming through Multimedia*. New York : Cambridge University Press, 2000. isbn : 978-0521644082 (cf. p. 99).

[124] Y. Jaafar et K. Bouzoubaa. « Arabic Natural Language Processing from Software Engineering to Complex Pipeline ». In : *2015 First International Conference on Arabic Computational Linguistics (ACLing)*. Cairo, Egypt : IEEE, avr. 2015. doi : 10.1109/acling.2015.11 (cf. p. 125-127, 245).

[125] Y. Jaafar et K. Bouzoubaa. « Benchmark of Arabic morphological analyzers challenges and solutions ». In : *2014 9th International Conference on Intelligent Systems : Theories and Applications (SITA-14)*. IEEE, mai 2014, p. 1-6. doi : 10.1109/sita.2014.6847312 (cf. p. 125).

[126] Y. Jaafar, D. Namly, K. Bouzoubaa et A. Yousfi. « Enhancing Arabic stemming process using resources and benchmarking tools ». In : *Journal of King Saud University - Computer and Information Sciences* 29.2 (avr. 2017), p. 164-170. issn : 1319-1578. doi : 10.1016/j.jksuci.2016.11.010 (cf. p. 87).

[127] M. Jarrar, D. Akra et T. Hammouda. « ALMA : Fast Lemmatizer and POS Tagger for Arabic ». In : *Proceedings of the 2024 AI in Computational Linguistics (ACLING 2024)*. Dubai : ELSEVIER, 2024 (cf. p. 153).

[128] T. Joachims. « Text categorization with Support Vector Machines : Learning with many relevant features ». In : *Proceedings of ECML*. Springer, 1998, p. 137-142 (cf. p. 145).

[129] T. Joachims. « Training linear SVMs in linear time ». In : *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD06. ACM, août 2006. doi : 10.1145/1150402.1150429 (cf. p. 131).

[130] Y. Kadri. « Query translation for English-Arabic cross language information retrieval ». In : *TALN conference*. 2004 (cf. p. 92).

[131] Y. Kadri et J.-Y. Nie. « Effective Stemming for Arabic Information Retrieval ». In : *Proceedings of the International Conference on the Challenge of Arabic for NLP/MT*. London, UK, oct. 2006, p. 68-75. url : https://aclanthology.org/2006.bcs-1.6 (cf. p. 92).

[132] G. Karthik Kumar, K. Sudheer et P. Avinesh. « Comparative Study of Various Machine Learning Methods For Telugu Part of Speech Tagging ». In : *Proceedings of the NLPAI Machine Learning 2006 Competition*. 2006 (cf. p. 142).

[133] U. Keller, H. Lausen, D. Fensel et R. Lara. « Semantic web service discovery in the WSMO framework ». In : *Semantic Web Services : Theory, Tools and Applications* (avr. 2004), p. 1-44 (cf. p. 113).

[134] I. A. Al-Kharashi et M. W. Evens. « Comparing words, stems, and roots as index terms in an Arabic Information Retrieval system ». In : *Journal of the American Society for Information Science* 45.8 (sept. 1994), p. 548-560. issn : 1097-4571. doi : 10.1002/(sici)1097-4571(199409)45:8<548::aid-asi3>3.0.co;2-x (cf. p. 33).

[135] I. A. A. Kharashi et I. A. A. Sughaiyer. « Rule Merging in a Rule-Based Arabic Stemmer ». In : *COLING 2002: The 19th International Conference on Computational Linguistics*. 2002. url : https://aclanthology.org/C02-1037 (cf. p. 66).

[136] R. M. Al-Khatib, T. Zerrouki, M. M. A. Shquier, A. Balla et A. Al-Khateeb. « A New Enhanced Arabic Light Stemmer for IR in Medical Documents ». In : *Computers, Materials & Continua* (2021). url : https://api.semanticscholar.org/CorpusID:234227480 (cf. p. 109, 110, 245).

[137] R. M. Al-Khatib, T. Zerrouki, M. M. Abu Shquier et A. Balla. « Tashaphyne 0.4: a new arabic light stemmer based on rhyzome modeling approach ». In : *Information Retrieval Journal* 26.1–2 (déc. 2023). issn : 1573-7659. doi : 10.1007/s10791-023-09429-y (cf. p. 107).

[138] S. Khoja. « APT : Arabic part-of-speech tagger ». In : *Proceedings of the Student Workshop at NAACL*. 2001, p. 20-25 (cf. p. 144, 145).

[139] S. Khoja et R. Garfield. *Stemming Arabic Text*. Rapp. tech. Lancaster, UK. Lancaster University Computer Department, 1999 (cf. p. 78).

[140] L. Khreisat. « Arabic text classification using N-gram frequency statistics : a comparative study ». In : *Proceedings of the 2006 International Conference on Data Mining*. USCCM. Las Vegas, NV, 2006, p. 78-82 (cf. p. 100).

[141] J. M. Kleinberg. « Authoritative sources in a hyperlinked environment ». In : *Journal of ACM* 46 (1999), p. 604-632 (cf. p. 118).

[142] K. Koskenniemi. « Two-Level Model for Morphological Analysis ». In : *Proceedings of the 8th International Joint Conference on Artificial Intelligence. Karlsruhe, FRG, August 1983*. Sous la dir. d'A. Bundy. T. 2. William Kaufmann, 1983, p. 683-685. url : http://ijcai.org/Proceedings/83-2/Papers/020.pdf (cf. p. 12, 73).

[143] R. Koulali et A. Meziane. « A contribution to Arabic named entity recognition ». In : *2012 Tenth International Conference on ICT and Knowledge Engineering*. IEEE. 2012, p. 46-52 (cf. p. 56).

[144] D. Kouloughli. « Sur la phonographématique arabe ». In : *Analyses/Théorie* 1 (1982), p. 79-151 (cf. p. 24).

[145] T. Kudo, K. Yamamoto et Y. Matsumoto. « Japanese morphological analysis using a lattice of finite automata ». In : *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2002, p. 41-49 (cf. p. 73).

[146] T. Kudoh et Y. Matsumoto. « Use of support vector learning for chunk identification ». In : *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning* -. Association for Computational Linguistics, 2000. doi : 10.3115/1117601.1117635 (cf. p. 145).

[147] S. Kulick, R. Gabbard et M. Marcus. « Parsing the Arabic Treebank : Analysis and Improvements ». In : *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT 2006)*. Prague, Czech Republic, 2006, p. 31-42 (cf. p. 59).

[148] D. Kumawat et V. Jain. « POS tagging approaches : a comparison ». In : *International Journal of Computer Applications* 118.6 (2015), p. 32-38 (cf. p. 142).

[149] J. Lafferty, A. McCallum et F. Pereira. « Conditional random fields : Probabilistic models for segmenting and labeling sequence data ». In : *Proceedings of the 18th International Conference on Machine Learning (ICML)*. T. 1. 2. 2001, p. 282-289 (cf. p. 133).

[150] Y. A. Al-Lahham, K. Matarneh et M. Hasan. « Conditional arabic light stemmer : condlight ». In : *Int. Arab J. Inf. Technol.* 15 (2018), p. 559-564. url : https://api.semanticscholar.org/CorpusID:49601822 (cf. p. 89).

[151] L. S. Larkey, L. Ballesteros et M. E. Connell. « Light Stemming for Arabic Information Retrieval ». In : *Arabic Computational Morphology : Knowledge-based and Empirical Methods*. Sous la dir. d'A. Soudi, A. v. d. Bosch et G. Neumann. Dordrecht : Springer Netherlands, 2007, p. 221-243. isbn : 9781402060458. doi : 10.1007/978-1-4020-6046-5_12. url : https://doi.org/10.1007/978-1-4020-6046-5_12 (cf. p. 86, 87).

[152] L. S. Larkey et L. Ballesteros. « Improving stemming for Arabic information retrieval : light stemming and co-occurrence analysis ». In : *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Tampere, Finland, 2002 (cf. p. 79, 86).

[153] L. S. Larkey et M. E. Connell. « Arabic Information Retrieval at UMass in TREC-10 ». In : *Text Retrieval Conference*. University of Massachusetts. Jan. 2001, p. 562-570. doi : 10.21236/ada456273 (cf. p. 79).

[154] G. Lebboss. « Contribution à l'analyse sémantique des textes arabes ». Thèse de doct. Université Paris 8, 2016 (cf. p. 11).

[155] C. Lv, H. Liu, Y. Dong et Y. Chen. « Corpus based part-of-speech tagging ». In : *International Journal of Speech Technology* 19.3 (2016), p. 647-654 (cf. p. 142).

[156] M. Maamouri et A. Bies. « Developing an Arabic Treebank : Methods, Guidelines, Procedures, and Tools ». In : *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*. Geneva, Switzerland : COLING, août 2004, p. 2-9. url : https://aclanthology.org/W04-1602 (cf. p. 59).

[157] M. Maamouri, A. Bies, T. Buckwalter et W. Mekki. « The penn arabic treebank : Building a large-scale annotated arabic corpus ». In : *NEMLAR Conference on Arabic Language Resources and Tools*. Jan. 2004 (cf. p. 59).

[158] J. Makhlouta et H. Harkous. « AUBSarf : Compositional Non-deterministic Finite-state Automata for Arabic Morphological Analysis ». Link to researchgate is the only pointer to this work. 2010. url : https://www.researchgate.net/publication/50996104_AUBSarf_Compositional_Non-deterministic_Finite-state_Automata_for_Arabic_Morphological_Analysis (cf. p. 105).

[159] C. Manning et al. « StanfordCoreNLP : A Java Suite for Natural Language Processing ». In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. 2014 (cf. p. 151).

[160] M. P. Marcus, B. Santorini et M. A. Marcinkiewicz. « Building a Large Annotated Corpus of English : The Penn Treebank ». In : *Computational Linguistics* 19 (1993), p. 313-330 (cf. p. 59, 60).

[161] N. Marques et G. Lopes. « Using Neural Nets for Portuguese Part-of-Speech Tagging ». In : *Proceedings of the Fifth International Conference on Cognitive Science of Natural Language Processing*. 1996, August (cf. p. 143).

[162] G. de Melo. « While some of these may be found in external resources (de Melo 2014), others may be fairly rare ». In : (2014) (cf. p. 18).

[163] M. Moftah, W. akhr, S. Abdou et M. Rashwan. « Stem-based Arabic language models experiments ». In : *Proceedings of the Second International Conference on Arabic Language Resources and Tools, Cairo, Egypt, April. The MEDAR Consortium*. Citeseer. 2009 (cf. p. 27).

[164] B. Mohamed et M. Azzeddine. « Approche hybride pour le développement d'un lemmatiseur pour la langue arabe ». In : *13éme Colloque Africain sur la Recherche en Informatique et Mathématiques Appliquées*. 2016 (cf. p. 126, 127).

[165] S. Mohamed, W. Ata et N. Darwish. « A new technique for automatic text categorization for Arabic documents ». In : *Proceedings of the 5th IBIMA International Conference on Internet and Information Technology in Modern Organizations*. Cairo, Egypt, 2005 (cf. p. 50).

[166] W. Monroe, S. Green et C. D. Manning. « Word Segmentation of Informal Arabic with Domain Adaptation ». In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2014. doi : 10.3115/v1/p14-2034 (cf. p. 132).

[167] M. Mustafa, A. Eldeen, S. Bani-Ahmad et A. Elfaki. « A Comparative Survey on Arabic Stemming : Approaches and Challenges ». In : *Intelligent Information Management* 9 (2017), p. 39-67. doi : 10.4236/iim.2017.92003 (cf. p. 79).

[168] S. H. Mustafa. « Word Stemming for Arabic Information Retrieval : The Case for Simple Light Stemming ». In : *Abhath Al-Yarmouk : Science & Engineering Series* 21.1 (2012), p. 123-144 (cf. p. 57).

[169] D. Namly, K. Bouzoubaa, A. El Jihad et S. L. Aouragh. « Improving Arabic Lemmatization Through a Lemmas Database and a Machine-Learning Technique ». In : *Recent Advances in NLP : The Case of Arabic Language*. Cham : Springer, 2020, p. 81-100 (cf. p. 126).

[170] M. Y. Al-Nashashibi, D. Neagu et A. A. Yaghi. « An improved root extraction technique for Arabic words ». In : *2010 2nd International Conference on Computer Technology and Development*. IEEE, nov. 2010, p. 264-269. doi : 10.1109/icctd.2010.5645872 (cf. p. 95).

[171] M. Y. Al-Nashashibi, D. Neagu et A. A. Yaghi. « Stemming techniques for Arabic words : A comparative study ». In : *2010 2nd International Conference on Computer Technology and Development*. IEEE, nov. 2010. doi : 10.1109/icctd.2010.5645873 (cf. p. 94, 95).

[172] R. Nelken et S. M. Shieber. « Arabic Diacritization Using Finite-State Transducers ». In : *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*. Ann Arbor, 2005, p. 79-86 (cf. p. 59).

[173] H. Ney et U. Essen. « On smoothing techniques for bigram-based natural language modelling ». In : *ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*. T. 2. IEEE. 1991, p. 825-828 (cf. p. 127).

[174] R. Parker, D. Graff, K. Chen, J. Kong et K. Maeda. *Arabic Gigaword Fifth Edition*. Version V1. 2011. doi : 11272.1/AB2/CP764S. url : https://hdl.handle.net/11272.1/AB2/CP764S (cf. p. 55).

[175] A. Pasha, M. Al-Badrashiny, M. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow et R. M. Roth. « MADAMIRA : A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic ». In : *International Conference on Language Resources and Evaluation*. 2014, p. 4394-4398. url : https://api.semanticscholar.org/CorpusID:10887722 (cf. p. 129).

[176] J. Pennington, R. Socher et C. D. Manning. « Glove : Global vectors for word representation ». In : *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, p. 1532-1543 (cf. p. 200).

[177] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee et L. Zettlemoyer. « Deep Contextualized Word Representations ». In : *Proceedings of NAACL-HLT 2018*. 2018, p. 2227-2237 (cf. p. 143).

[178] M. Porter. « An algorithm for suffix stripping ». In : *Program* 14.3 (1980), p. 130-137 (cf. p. 111).

[179] M. F. Porter. « Snowball : A language for stemming algorithms ». In : *Information Retrieval* 4.1 (2001), p. 67-85 (cf. p. 73).

[180] P. Qi, T. Dozat, Y. Zhang et C. D. Manning. « Universal Dependency Parsing from Scratch ». In : *Proceedings of the CoNLL 2018 Shared Task : Multilingual Parsing from Raw Text to Universal Dependencies*. Sous la dir. de D. Zeman et J. Hajič. Brussels, Belgium : Association for Computational Linguistics, oct. 2018, p. 160-170. doi : 10.18653/v1/K18-2016. url : https://aclanthology.org/K18-2016 (cf. p. 134).

[181] P. Qi, Y. Zhang, Y. Zhang, J. Bolton et C. D. Manning. « Stanza : A Python Natural Language Processing Toolkit for Many Human Languages ». In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics : System Demonstrations*. Sous la dir. d'A. Celikyilmaz et T.-H. Wen. Online : Association for Computational Linguistics, juill. 2020, p. 101-108. doi : 10.18653/v1/2020.acl-demos.14. url : https://aclanthology.org/2020.acl-demos.14 (cf. p. 134).

[182] M. Al-Rabiah. « Building A Distributional Semantic Model for Traditional Arabic and Investigating its Novel Applications to The Holy Quran ». Thèse de doct. King Saud University, Riyadh, 2014 (cf. p. 57).

[183] A. Ratnaparkhi. « A Maximum Entropy Model for Part-Of-Speech Tagging ». Ph.D. Thesis. Philadelphia, PA : University of Pennsylvania, 1996. url : https://repository.upenn.edu/dissertations/AAI9638984 (cf. p. 149).

[184] A. Roman. « Le système phonologique de l'arabe "classique" contemporain ». In : *Revue des mondes musulmans et de la Méditerranée* 18.1 (1974), p. 125-130. doi : 10.3406/remmm.1974.1288. url : https://www.persee.fr/doc/remmm_0035-1474_1974_num_18_1_1288 (cf. p. 19).

[185] M. K. Saad et W. M. Ashour. « Arabic Morphological Tools for Text Mining ». In : *6th International Conference on Electrical and Computer Systems*. T. 18. 18. Accessed : Jul. 01, 2020. 2010, p. 112-117. url : https://iugspace.iugaza.edu.ps/handle/20.500.12358/25196 (cf. p. 78, 86).

[186] M. K. Saad et W. M. Ashour. « Open Source Arabic Corpora OSAC ». In : *Proceedings of the 6th ArchEng International Symposiums, 6th International Conference on Electrical and Computer Systems (EECS'10)* (2010) (cf. p. 51).

[187] S. Salameh. « A Review of Part of Speech Tagger for Arabic Language ». In : *International Journal of Computation and Applied Sciences (IJOCAAS)* 4.3 (2018). Accessed : 2025-08-09, p. 1867-1873. issn : 2399-4509. url : https://www.academia.edu/87055447/A_Review_of_Part_of_Speech_Tagger_for_Arabic_Language (cf. p. 143, 245).

[188] M. Saleh et A. Al-Haj. « Linguistic Corpus Project ». In : *Proceedings of the Journal of the Arab Tongue* (1999) (cf. p. 48).

[189] M. Saleh et A. Al-Haj. « this is the Arabic Linguistic Corpus Project ; and this is the Algerian percep- tion of it ». In : *Proceedings of the Algerian news today* (2014) (cf. p. 48).

[190] H. A. Salem. « A microcomputer based Arabic bibliographic information retrieval system with relational thesauri (Arabic-IRS) ». Thèse de doct. Illinois Institute of Technology, 1992 (cf. p. 47).

[191] M. Sawalha, E. Atwell et M. A. Abushariah. « SALMA : Standard Arabic Language Morphological Analysis ». In : *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. IEEE. 2013 (cf. p. 147, 148, 245).

[192] H. Schmid. « Probabilistic part-of-speech tagging using decision trees ». In : *Proceedings of International Conference on New Methods in Language Processing*. 1994, p. 44-49 (cf. p. 148).

[193] H. Schmid. « Probabilistic part-of-speech tagging using decision trees ». In : *International conference on new methods in language processing*. Citeseer. 1995, p. 44-49 (cf. p. 73).

[194] H. Sellami et D. Daoui. « Multivariate Analysis and Forecasting of the MASI Index : Comparison of LSTM and GRU Approaches ». In : *Revue du Contrôle de la Comptabilité et de l'Audit* 9.2 (2025). Comparaison empirique entre LSTM et GRU sur séries temporelles financières, p. 319-334. issn : 2550-469X. url : https://www.revuecca.com/index.php/home/article/download/1164/1003/4250 (cf. p. 138).

[195] P. Sgall, E. Hajičová et J. Panevová. *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*. Dordrecht : Reidel, 1986 (cf. p. 60).

[196] H. Al-Shalabi, S. Tiun, N. Omar, K. Alezabi et F. Al-Aswadi. « The Effectiveness of Arabic Stemmers Using Arabized Word Removal ». In : *International Journal of Information Science and Management* 20 (oct. 2022), p. 85-100 (cf. p. 115).

[197] R. Al-Shalabi et M. Evens. « A Computational Morphology System for Arabic ». In : *Computational Approaches to Semitic Languages*. 1998. url : https://aclanthology.org/W98-1009 (cf. p. 77).

[198] M. Slim. « Analyse morpho-syntaxique automatique et reconnaissance des entités nommées en arabe standard ». Thèse de doct. Besançon, 2008 (cf. p. 33).

[199] B. Smine et C. Mankai. « تركيبة المركب الفعلي وطريقة تحليله مورفولوجيا ». In : *The Proceedings of the Third International Conference of Computer Science Practices in Arabic (CSPA '06)*. Sharjah, UAE, mars 2006 (cf. p. 113).

[200] N. A. Smith, D. A. Smith et R. W. Tromble. « Context-Based Morphological Disambiguation with Random Fields ». In : *Proceedings of HLT/EMNLP 2005*. Vancouver : Association for Computational Linguistics, 2005, p. 475-482 (cf. p. 59).

[201] O. Smrž. « ElixirFM — Implementation of Functional Arabic Morphology ». In : *ACL 2007 Proceedings of the Workshop on Computational Approaches to Semitic Languages : Common Issues and Resources*. Prague, Czech Republic : Association for Computational Linguistics, juin 2007, p. 1-8 (cf. p. 60, 99).

[202] O. Smrž. « Functional Arabic Morphology : Formal System and Implementation ». PhD thesis. Prague, Czech Republic : Charles University in Prague, 2007 (cf. p. 60, 61, 99).

[203] O. Smrž et J. Hajič. « The Other Arabic Treebank : Prague Dependencies and Functions ». In : *Arabic Computational Linguistics : Current Implementations*. Sous la dir. d'A. Farghaly. Stanford, CA, USA : CSLI Publications, 2008 (cf. p. 61).

[204] O. Smrž et P. Pajas. « MorphoTrees of Arabic and Their Annotation in the TrEd Environment ». In : *NEMLAR International Conference on Arabic Language Resources and Tools*. ELDA, 2004, p. 38-41 (cf. p. 60).

[205] F. Song et B. Croft. « A general language model for information retrieval ». In : *Proceedings of the Eighth International Conference on Information and Knowledge Management*. ACM. 1999, p. 316-321 (cf. p. 126).

[206] A. Soudi, E. Atwell et T. Hedlund. « Al-Khalil : an Arabic morphological analyzer and part-of-speech tagger ». In : *LREC*. 2012 (cf. p. 104).

[207] A. Soudi, A. van den Bosch et G. Neumann. *Arabic Computational Morphology : Knowledge-based and Empirical Methods*. 1st. Springer Publishing Company, Incorporated, 2007. isbn : 1402060459 (cf. p. 66).

[208] A. Spencer. *Morphological Theory : An Introduction to Word Structure in Generative Grammar*. Oxford & Cambridge, MA : Basil Blackwell, 1991 (cf. p. 110).

[209] R. Sproat. *Morphology and Computation*. The MIT Press, avr. 1992. isbn : 9780262284172. doi : 10.7551/mitpress/4775.001.0001. url : https://doi.org/10.7551/mitpress/4775.001.0001 (cf. p. 74).

[210] L. Al-Sulaiti. « Designing and developing a corpus of contemporary Arabic ». Thèse de doct. University of Leeds, 2004 (cf. p. 49).

[211] L. Al-Sulaiti et E. S. Atwell. « Extending the corpus of contemporary Arabic ». In : *Proceedings of CL'2005 Corpus Linguistics Conference*. Jan. 2005 (cf. p. 49).

[212] L. Al-Sulaiti et E. S. Atwell. « The design of a corpus of Contemporary Arabic ». In : *International Journal of Corpus Linguistics* 11.2 (juill. 2006), p. 135-171. issn : 1569-9811. doi : 10.1075/ijcl.11.2.02als (cf. p. 49).

[213] K. Taghva, R. Elkoury et J. Coombs. « A fast and accurate Arabic stemmer ». In : *Proceedings of the HLT/EMNLP on Arabic Language and Linguistics Workshop*. Association for Computational Linguistics. 2005, p. 10-17 (cf. p. 80).

[214] A. Al-Thubaity. « King Abdulaziz City for Science and Technology Arabic Corpus KACST ». In : *Proceedings of the Language Resources and Evaluation* 49.3 (sept. 2015), p. 721-751 (cf. p. 52).

[215] A. O. Al-Thubaity. « A 700M+ Arabic corpus : KACST Arabic corpus design and construction ». In : *Language Resources and Evaluation* 49.3 (2015), p. 721-751. doi : 10.1007/s10579-014-9284-1. url : https://doi.org/10.1007/s10579-014-9284-1 (cf. p. 52).

[216] K. Toutanova, D. Klein, C. Manning et Y. Singer. « Feature-Rich Partof-Speech Tagging with a Cyclic Dependency Network ». In : *Proceedings of the JHLT-NAACL* (2003), p. 252-259 (cf. p. 132).

[217] V. N. Vapnik. *The Nature of Statistical Learning Theory*. New York, USA : Springer-Verlag, 1995. isbn : 978-0387987804. url : https://link.springer.com/book/10.1007/978-1-4757-3264-1 (cf. p. 129).

[218] A. Viterbi. « Error bounds for convolutional codes and an asymptotically optimum decoding algorithm ». In : *IEEE Transactions on Information Theory* 13.2 (1967), p. 260-269 (cf. p. 126, 128).

[219] E. M. Voorhees. « Overview of TREC 2002 ». In : *The Eleventh Text Retrieval Conference (TREC 2002)*. 2002. url : https://trec.nist.gov/pubs/trec11/papers/OVERVIEW.11.pdf (cf. p. 48).

[220] E. M. Voorhees et D. Harman. « Overview of TREC 2001 ». In : *The Tenth Text Retrieval Conference, TREC 2001*. 2001. url : https://trec.nist.gov/pubs/trec10/paper (cf. p. 48).

[221] B. Waldvogel. « LIBLINEAR : A Library for Large Linear Classification ». In : (2008). url : http://liblinear.bwaldvogel.de (cf. p. 129).

[222] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini et al. *OntoNotes Release 5.0*. Rapp. tech. LDC2013T19. Philadelphia, PA : Linguistic Data Consortium, 2013 (cf. p. 61).

[223] J. Xu, A. Fraser et R. M. Weischedel. « Empirical Studies in Strategies for Arabic Retrieval ». In : *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*. Tampere, Finland, 2002, p. 269-274 (cf. p. 66).

[224] S. Yang et Y. Zhou. « LSTM and GRU Neural Network Performance Comparison Study : Taking Yelp Review Dataset as an Example ». In : *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*. Étude comparative sur corpus textuel Yelp. IEEE, 2020, p. 98-102. doi : 10.1109/IWECAI50956.2020.00027. url : https://www.researchgate.net/publication/347267378 (cf. p. 138).

[225] d. K. Y. Yemane Tedl. « Morphological Segmentation with Window LSTM Neural Networks ». In : *International Journal on Natural Language Computing* 7.2 (2018). doi : 10.5121/ijnlc.2018.7 (cf. p. 18).

[226] Z. Zabokrtsky et O. Smrž. « Arabic Syntactic Trees : from Constituency to Dependency ». In : *10th Conference of the European Chapter of the Association for Computational Linguistics*. Budapest, Hungary, 2003, p. 183-186 (cf. p. 61).

[227] Y. Zaki. « Induction de grammaire de phrase en arabe standard moderne ». Thèse de doct. Université Paris 8, 2022 (cf. p. 11).

[228] F. Zaraket et J. Makhlouta. « Arabic Morphological Analyzer with Agglutinative Affix Morphemes and Fusional Concatenation Rules ». In : *Proceedings of COLING 2012: Demonstration Papers*. Sous la dir. de M. Kay et C. Boitet. Mumbai, India : The COLING 2012 Organizing Committee, déc. 2012, p. 517-526. url : https://aclanthology.org/C12-3065 (cf. p. 110).

[229] I. Zeroual, D. Goldhahn, T. Eckart et A. Lakhouaja. « OSIAN : Open Source International Arabic News Corpus - Preparation and Integration into the CLARIN-infrastructure ». In : *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Sous la dir. de W. El-Hajj, L. H. Belguith, F. Bougares, W. Magdy, I. Zitouni, N. Tomeh, M. El-Haj et W. Zaghouani. Florence, Italy : Association for Computational Linguistics, août 2019, p. 175-182. doi : 10.18653/v1/W19-4619. url : https://aclanthology.org/W19-4619 (cf. p. 57).

[230] I. Zeroual et A. Lakhouaja. « Adapting a Decision Tree Based Tagger for Arabic ». In : *Proceedings of the International Conference on Information Technology for Organizations Development (IT4OD)*. IEEE. 2016, p. 1-6 (cf. p. 149).

[231] I. Zeroual, A. Lakhouaja et R. Belahbib. « Towards a standard Part of Speech tagset for the Arabic language ». In : *Journal of King Saud University - Computer and Information Sciences* 29.2 (2017). Arabic Natural Language Processing : Models, Systems and Applications, p. 171-178. issn : 1319-1578. doi : https://doi.org/10.1016/j.jksuci.2017.01.006. url : https://www.sciencedirect.com/science/article/pii/S1319157817300265 (cf. p. 142).

[232] T. Zerrouki et A. Balla. « Tashkeela : Novel corpus of Arabic vocalized texts, data for auto-diacritization systems ». In : *Proceedings of The National Computer Science Engineering School (ESI)* (2017) (cf. p. 53).

[233] C. Zhai et J. D. Lafferty. « Model-based feedback in the language modeling approach to information retrieval ». In : *International Conference on Information and Knowledge Management*. 2001. url : https://api.semanticscholar.org/CorpusID:1043470 (cf. p. 93).

[234] A. Zhang, Z. C. Lipton, M. Li et A. J. Smola. *Dive into Deep Learning*. https://d2l.ai/chapter_recurrent-modern/gru.html. Tutoriel interactif sur GRU. 2023 (cf. p. 137).

[235] I. Zitouni, J. S. Sorensen et R. Sarikaya. « Maximum Entropy Based Restoration of Arabic Diacritics ». In : *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia : Association for Computational Linguistics, juill. 2006, p. 577-584 (cf. p. 59).

Table des figures

1.1 Génération d'un mot arabe (Al Hajjar, 2010) 27

2.1 Analyse morphologique de Kalimat 56
2.2 Analyse morphologique d'un mot dans PADT 60

3.1 Rappel et précision moyens (BP = pluriel irrég.) (Goweder et al., 2004) 92
3.2 Morph2 (N. K. Chaâben, Hadrich Belguith et Ben Hamadou, 2010) 97
3.3 Vocalisation (N. K. Chaâben, Hadrich Belguith et Ben Hamadou, 2010) 98
3.4 Exemple de segmentation fait par Sarf 106
3.5 Fonctionnalités de Sarf 106
3.6 Diagramme de Sarf (Alkhalifa, Al-Saydi et Al-Khalifa, 2014) 107
3.7 Segmentation de فسيكتبونهما(Al-Khatib, Zerrouki, Shquier et al., 2021) 109
3.8 Extraction de préfixes par arbre imbriqué (Al-Khatib, Zerrouki, Shquier et al., 2021) 110
3.9 Classes supérieures d'Aramo (El Abdi, Ben Ali et Ben Yahia, 2022) 113
3.10 Classes liées au verbe dans Aramo (El Abdi, Ben Ali et Ben Yahia, 2022) 114
3.11 Architecture d'Aramo (El Abdi, Ben Ali et Ben Yahia, 2022) 114

4.1 Architecture de BASMA (Alansary, 2019) 124
4.2 Architecture du lemmatiseur SAFAR, Jaafar et Bouzoubaa (2015) . 126
4.3 Lemmatisation sans contexte de SAFAR, Jaafar et Bouzoubaa (2015) 127
4.4 Vue d'ensemble du pipeline NLP neuronal de Stanza 134

5.1 Catégories d'étiqueteurs POS (Salameh, 2018) 143
5.2 Algorithme Salma Pos (Sawalha, E. Atwell et Abushariah, 2013) . . 148

6.1 Structure de Kalimat corpus 163
6.2 Absence de segmentation dans Kalimat 164
6.3 Courbe Zipf du Gold Corpus et mots fréquents 178
6.4 Outil d'aide à la correction manuelle 182
6.5 Processus d'évaluation 187

Liste des tableaux

1.1 Table des consonnes . . . 20
1.2 Alphabet (unicode) . . . 22
1.3 Les diacritiques arabes . . . 23
1.4 Analyse du mot أتلعبون/ʔatlʕabūn/ (Est-ce que vous jouez) . . . 28
1.5 Exemples de segmentation . . . 29
1.6 Clitiques . . . 31
1.7 Préfixes flexionnels . . . 38
1.8 Suffixes flexionnels . . . 39

2.1 La presse arabe dans le monde . . . 46
2.2 Corpus libres non annotés . . . 47
2.3 Corpus non annotés, non libres . . . 54
2.4 Corpus annotés libres . . . 56
2.5 Corpus annotés non libres . . . 58

3.1 Segmenteurs à base de règles . . . 75
3.2 Les analyses possibles du mot كتاب . . . 85
3.3 Résultats en recherche d'information selon les deux méthodes . . . 94
3.4 Exemples de radicaux trouvés par les deux approches . . . 94
3.5 Précision . . . 103
3.6 Accuracy sur Dice . . . 103
3.7 Accuracy sur Manhattan . . . 103
3.8 Évaluation d'Aramo (El Abdi, Ben Ali et Ben Yahia, 2022) . . . 115

4.1 Segmenteurs probabilistes . . . 118
4.2 Hypothèses de radicaux . . . 121

4.3 Segmenteurs contextuels par sélection . . . 122
4.4 Segmenteurs contextuels probabilistes . . . 129

5.1 Étiqueteurs pour l'arabe . . . 144

6.1 Clitiques normalisés . . . 159
6.2 Proclitiques après normalisation . . . 159
6.3 Enclitiques après normalisation . . . 160
6.4 Préfixes flexionnels . . . 160
6.5 Suffixes flexionnels . . . 161
6.6 Collection de données . . . 162
6.7 Statistiques de Kalimat . . . 162
6.8 Occurrences . . . 162
6.9 Tokens uniques . . . 162
6.10 Fréquence des mots les plus courants . . . 163
6.11 Conversion catégories Al-Khalil . . . 166
6.11 Conversion catégories Al-Khalil . . . 167
6.12 Conversion catégories Stanford . . . 168
6.13 Statistiques sur les POS . . . 170
6.14 Comparaison entre Stanford et Alkhalil (tous les mots) . . . 171
6.15 Mots uniques pour chaque POS . . . 172
6.16 Ambiguïtés de POS1 . . . 173
6.17 Ambiguïtés de POS2 . . . 173
6.18 Distribution des segments par catégorie . . . 175
6.19 Distribution des segments par catégorie sans POS2 . . . 175
6.20 Statistiques du corpus de référence . . . 177
6.21 Mots les plus fréquents . . . 178
6.22 Catégories finales des tokens . . . 179
6.23 Matrice de confusion de Stanford, POS1 . . . 180
6.24 Matrice de confusion de Stanford, POS2 . . . 180
6.25 Matrice de confusion d'Al-Khalil, POS1 . . . 181
6.26 Matrice de confusion d'Al-Khalil, POS2 . . . 181
6.27 Répartitions des segmentations par nombre de segments . . . 183
6.28 Type de segmentations les plus fréquentes . . . 183
6.29 Distribution des clitiques et affixes . . . 184

6.30 Segmentations multiples d'un même mot 185
6.31 Détection du radical 189
6.32 Erreurs les plus courantes 190
6.33 Segmentation complète 191
6.34 % par catégorie grammaticale 192
6.35 % des parties correctes 193
6.36 Nombre de parties 193

7.1 Exactitude par POS 197
7.2 Règles sur POS1 198
7.3 Règles sur POS2 et 3 198
7.4 Lots de Stanford 199
7.5 Métriques sur POS2 (lot sélectionné) 199
7.6 Comparaison entre Farasa & Awdeh et combinaison entre eux . . . 201

A.1 Mots irréguliers 208
A.2 Mots irréguliers pour Elrajubi stemmer 209
A.3 Différences entre les versions de Light 210

B.1 *ta marbuta* (4 cas) 211
B.2 Procl + P (4 cas) 211
B.3 Autres (5 cas) 212
B.4 Jeu de tests d'ambiguïtés hors corpus 213

Table des matières

Remerciements **3**

Résumé **7**

Abstract **9**

Introduction **11**

I Problématique **15**

1 La segmentation en arabe standard moderne **17**

1.1 Introduction 18
1.2 Bref aperçu phonologique 19
1.3 La graphie arabe 21
1.3.1 Les unités de la graphie 24
1.3.2 Variantes graphiques 24
1.4 Structure du mot 26
1.4.1 Clitiques et catégories 30
1.5 Morphologie du mot 32
1.5.1 Noms et adjectifs 33
1.5.2 Verbes 34
1.5.3 Verbes faibles 36
1.5.4 Affixes et catégories morphosyntaxiques 37
1.6 Types de segmentation 39
1.7 Conclusion 41

2 Corpus en langue arabe **43**
2.1 Introduction . . . 45
2.2 Les corpus non annotés d'accès libre . . . 47
2.2.1 SACS 1992-1997 . . . 47
2.2.2 Al-Raya 1996 . . . 48
2.2.3 Arabic Corpus Project 1999 . . . 48
2.2.4 TREC 2001-2002 . . . 48
2.2.5 Contemporary Arabic Corpus 2004-2005 . . . 49
2.2.6 Abbas & Smaili 2004-2005 . . . 49
2.2.7 Adjir Corpora 2005 . . . 50
2.2.8 Modern Standard Arabic Corpus 2005 . . . 50
2.2.9 Corpus Aljazira 2005 . . . 50
2.2.10 Open Source Arabic Corpus (OSAC) 2010 . . . 51
2.2.11 University of Jordan Arabic Corpus (UJAC) 2013 . . . 51
2.2.12 KACST 2015 . . . 52
2.2.13 Abu El-Khair 2016 . . . 52
2.2.14 Tashkeela 2017 . . . 53
2.3 Les corpus non annotés d'accès non libre . . . 53
2.3.1 LDC 2001 . . . 54
2.3.2 ELRA 2001 . . . 54
2.3.3 Arabic Gigaword 2003-2011 . . . 55
2.4 Les corpus annotés d'accès libre . . . 55
2.4.1 KALIMAT 2013 . . . 56
2.4.2 KSUCCA 2014 . . . 57
2.4.3 OSIAN 2019 . . . 57
2.5 Les corpus annotés non libres . . . 58
2.5.1 NEMLAR 2003-2005 . . . 58
2.5.2 ATB 2003-2011 . . . 59
2.5.3 PADT 2004 . . . 60
2.5.4 OntoNotes 2013 . . . 61
2.5.5 The International Corpus of Arabic (ICA) 2014 . . . 61
2.6 Conclusion . . . 62

II État de l'art 63

Introduction à l'état de l'art 65

3 La segmentation par règles 71
3.1 Introduction . . . 73
3.2 Segmenteurs . . . 73
3.2.1 ALPNET . . . 76
3.2.2 Al-Shalabi et Evens . . . 77
3.2.3 Khoja . . . 78
3.2.4 Morpho3 . . . 80
3.2.5 ISRI . . . 80
3.2.6 Aljlay & Frieder . . . 83
3.2.7 AraMorph (BAMA, SAMA) . . . 84
3.2.8 Light10 (Al-Stem) . . . 86
3.2.9 Light10 → Elrajubi . . . 88
3.2.10 Light10 → CondLight . . . 89
3.2.11 Goweder . . . 91
3.2.12 Kadri & Nie . . . 92
3.2.13 Al-Ameed . . . 94
3.2.14 Morph2 . . . 96
3.2.15 ElixirFM . . . 99
3.2.16 Bigrammes . . . 100
3.2.17 Alkhalil . . . 104
3.2.18 SARF . . . 105
3.2.19 Tashaphyne . . . 107
3.2.20 Zaraket . . . 110
3.2.21 Assem . . . 111
3.2.22 ARAMO . . . 112
3.3 Conclusion . . . 115

4 La segmentation complexe 117
4.1 Introduction . . . 118
4.2 Segmenteurs non contextuels probabilistes . . . 118
4.2.1 Chen & Gey . . . 118

4.2.2 Sebawai . . . 120
4.3 Segmenteurs contextuels par sélection . . . 122
4.3.1 BAMAE, BASMA . . . 123
4.3.2 SAFAR . . . 125
4.4 Segmenteurs contextuels probabilistes . . . 128
4.4.1 MADA, MADAMIRA . . . 129
4.4.2 Farasa . . . 130
4.4.3 Stanford NLP . . . 132
4.5 Segmenteurs contextuels neuronaux . . . 133
4.6 Modèles d'apprentissage séquentiel . . . 135
4.6.1 LSTM . . . 135
4.6.2 GRU . . . 137
4.7 Conclusion . . . 138

5 Étiquetage des parties du discours 141
5.1 Introduction . . . 142
5.2 Modèles pour l'arabe . . . 144
5.2.1 APT de Khoja . . . 144
5.2.2 Arabic SVM POS tagger . . . 145
5.2.3 Salma . . . 147
5.2.4 TreeTagger . . . 148
5.2.5 MADAMIRA POS . . . 149
5.2.6 Farasa POS tagger . . . 149
5.2.7 Etiqueteur par colonie d'abeilles . . . 150
5.2.8 LSTM POS Tagger . . . 150
5.2.9 CAMeL POS tagger . . . 151
5.2.10 Stanford POS Tagger . . . 151
5.2.11 ALP POS tagger . . . 152
5.2.12 Alma . . . 153
5.3 Conclusion . . . 154

III Système et réalisations 155

6 Corpus de référence et évaluation des segmenteurs 157

6.1 Introduction . . . 158
6.2 Travail sur la normalisation . . . 158
6.3 Corpus de référence initial . . . 161
6.3.1 Kalimat . . . 161
6.3.2 Prétraitement catégoriel . . . 165
6.3.3 Comparaison catégories Stanford / Al-Khalil . . . 169
6.3.4 Segmentation automatique . . . 174
6.4 Production du corpus de référence . . . 177
6.4.1 Analyse catégorielle . . . 178
6.4.2 Segmentation manuelle . . . 181
6.4.3 Description de la segmentation . . . 183
6.4.4 Description du gold standard . . . 186
6.5 Évaluation des segmenteurs . . . 186
6.5.1 Analyse des résultats . . . 188
6.6 Conclusion . . . 194

7 Éléments pour un segmenteur 195
7.1 Introduction . . . 196
7.2 Données d'entrée . . . 196
7.2.1 Prétraitement et encodage des données catégorielles . . . 197
7.2.2 Prétraitement et encodage des segmentations . . . 200
7.2.3 Encodage final . . . 201
7.3 Corpus d'entraînement et données fournies . . . 201
7.3.1 Gold Corpus . . . 201
7.3.2 Tools . . . 202
7.3.3 Silver Corpus . . . 203
7.4 Conclusion . . . 203

Conclusion et perspectives 205

A Mots invariables 207

B Segmentations multiples du gold corpus 211

Mes publications 217

Bibliographie **219**

Table des figures **245**

Liste des tableaux **247**

Table des matières **256**