ENSEMBLE MORPHOSYNTACTIC ANALYSER FOR CLASSICAL ARABIC

Abdulrahman Mohammed S. Alosaimy

Submitted in accordance with the requirements for the degree of Doctor of Philosophy

The University of Leeds

School of Computing

October 2018

PUBLICATIONS

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Chapters 4, 5, 6, 8, and 9 of this thesis are based on jointly-authored publications. The candidate is the principal author of all original contributions presented in these papers, the co-authors acted in an advisory capacity, providing feedback, general guidance and comments.

Chapter 4

Alosaimy, A. and Atwell, E. (2015) 'A Review of Morphosyntactic Analysers and Tag-Sets for Arabic Corpus Linguistics', in *Eighth International Corpus Linguistics conference (CL2015)*, pp. 16–19.

Alosaimy, A. and Atwell, E. (2017) 'Tagging Classical Arabic Text using Available Morphological Analysers and Part of Speech Taggers', *Journal for Language Technology and Computational Linguistics*. German Society for Computational Linguistics & Language Technology (GSCL), 32(1), pp. 1–26.

Chapter 5

Alosaimy, A. and Atwell, E. (2016) 'Ensemble Morphosyntactic Analyser for Classical Arabic', in *Second International Conference on Arabic Computational Linguistics*. Konya, Turkey.

Alosaimy, A. and Atwell, E. (2016) 'SAWAREF: Multi-component Toolkit for Arabic Morphosyntactic Tagging', in *the 9th Saudi Students Conference*. Birmingham, UK. (poster)

Alosaimy, A. and Atwell, E. (2018) 'Diacritisation of a Highly Cited Text: A Classical Arabic Book as a Case', in *2nd IEEE International Workshop on Arabic and derived Script Analysis and Recognition (ASAR 2018)*. London, UK.

Chapter 6

Alosaimy, A. and Atwell, E. (2017) 'Joint Alignment of Segmentation and Labelling for Arabic Morphosyntactic Taggers', *International Journal of Computational Linguistics*. CSC Journals.

Alosaimy, A. and Atwell, E. (2017) ‘Ensemble Joint Segmentation and POS Tagger for Arabic’ in *The Workshop on Computational Approaches to Morphologically Rich Languages CAMRL.* Leeds, UK. (presentation).

Chapter 8

Alosaimy, A. and Atwell, E. (2017) ‘Sunnah Arabic Corpus: Design and Methodology’, in *Proceedings of the 5th International Conference on Islamic Applications in Computer Science and Technologies*. Semarang, Indonesia. (in press)

Chapter 9

Alosaimy, A. and Atwell, E. (2018) ‘Web-based Annotation Tool for Inflectional Language Resources Major features’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Miyazaki, Japan: European Language Resources Association (ELRA), pp. 3933–3939.

ACKNOWLEDGEMENTS

First and foremost, all praise and thanks to Allah The Almighty for His graces and guidance. Peace and Blessings of Allah be upon the Prophet Muhammad.

I would like to express my heartfelt gratitude and deepest thanks to my PhD supervisor Prof. Eric Atwell for his continuous encouragement, assistance and valuable advice throughout this work. He kept motivating and supporting me along the way, and his advice returne me back to the right path when I lose confidence. He encouraged me to write papers and attend meetings and conferences.

I would never have the chance to study in the UK without the financial aid of my sponsor, Al-Imam Mohammed bin Saud University, and the government of the Kingdom of Saudi Arabia. I hope to be able to repay part of the debt by transferring the knowledge and promoting research in Saudi Arabia.

I would like to thank Dr Abdullah Alfaifi, Ayman Alghamdi for their support in many linguistic aspects of my work. They voluntarily accept my request for help in some linguistic experiments, and help in some of decisions.

I would like to thank my colleagues: Dr. Sameer Alrehaili, and Mohammed Alqahtani for their continued support. Our meetings where we discuss our projects will never be forgotten.

DEDICATION

This thesis is dedicated to my family and parents who endured my years-long absence in silence

To my father, Mohammed, who is always inspirational throughout my life. His Arabic challenges when I was a kid helped me improve my critical thinking and reach the level where I am now.

To my mother, Asmaa, who is like a candle that consumes itself to light the way for others, with love and passion. Without your prayers and encouragement, I would not have reached my goals.

To my wife, Fatimah, who has been with me through thick and thin. It is because of you and your patience, understanding and support in the difficult times, I kept upholding to my aims.

To my two lovely kids: Mohammed and Asmaa. The smiles on their faces when I come back home late energizes me again and again.

ABSTRACT

Classical Arabic (CA) is an influential language for Muslim lives around the world. It is the language of two sources of Islamic laws: the Quran and the Sunnah, the collection of traditions and sayings attributed to the prophet Mohammed. However, classical Arabic in general, and the Sunnah, in particular, is underexplored and under-resourced in the field of computational linguistics. This study examines the possible directions for adapting existing tools, specifically morphological analysers, designed for modern standard Arabic (MSA) to classical Arabic.

Morphological analysers of CA are limited, as well as the data for evaluating them. In this study, we adapt existing analysers and create a validation data-set from the Sunnah books. Inspired by the advances in deep learning and the promising results of ensemble methods, we developed a systematic method for transferring morphological analysis that is capable of handling different labelling systems and various sequence lengths.

In this study, we handpicked the best four open access MSA morphological analysers. Data generated from these analysers are evaluated before and after adaptation through the existing Quranic Corpus and the Sunnah Arabic Corpus. The findings are as follows: first, it is feasible to analyse under-resourced languages using existing comparable language resources given a small sufficient set of annotated text. Second, analysers typically generate different errors and this could be exploited. Third, an explicit alignment of sequences and the mapping of labels is not necessary to achieve comparable accuracies given a sufficient size of training dataset.

Adapting existing tools is easier than creating tools from scratch. The resulting quality is dependent on training data size and number and quality of input taggers. Pipeline architecture performs less well than the End-to-End neural network architecture due to error propagation and limitation on the output format. A valuable tool and data for annotating classical Arabic is made freely available.

TABLE OF CONTENTS

List of Tables

LIST OF FIGURES

Part I

Introduction and Literature Review

1 INTRODUCTION

1.1 This research

The topic of this research is the morphological analysis and POS tagging of classical Arabic (CA) texts. Morphological analysis and POS tagging are two preliminary steps in many text analytics applications from different disciplines. Many systems were developed to identify and analyse the Arabic text morphologically, i.e. by studying and analyzing the form of the word. They vary in complexity from light stemmers, linguistically based stemmers, lemmatisers, simple table-lookup analysers, complex morphology analysers, and POS taggers. These analysers handle Arabic's morphological-rich problem, and are useful for many downstream applications, such as syntax analysis, machine translation, information retrieval, question answering, and ontology construction.

However, most of the Arabic morphological analysers are designed and tuned for Modern Standard Arabic (MSA) and adapting these tools to under-resourced domains/languages is challenging.

This research proposes a systematic method for adapting multiple MSA morphological analysers **to the domain of classical Arabic text**, specifically for the Sunnah texts. Instead of adopting a single tool, like (Almeman, 2015) for dialects, or (Dukes and Habash, 2010) for Quranic Arabic, we pursue the method of **combining heterogeneous taggers** for the purpose of more robust and accurate morphological tagging of the Sunnah Arabic texts.

The Sunnah, also known as Hadith, is the collection of traditions and sayings attributed to the prophet of the Muslims, Mohammed (peace be upon him). The Sunnah, in particular, and classical Arabic, in general, with the exception of the

Quranic text, lack many computational linguistic resources such as treebanks and morphological annotation. In this research, we aim to fill the gap by implementing an accurate tagger for classical Arabic and **providing a semi-automatically morphologically annotated corpus** for a collection of Sunnah sayings.

While this research scope uses domain adaptation methods for adapting MSA taggers to classical Arabic text, it is designed to be language-agnostic and provide a systematic way for overcoming challenges of knowledge transfer. These challenges include mismatch of labelling schema between individual taggers and target classical text tagset. Besides, segmentation schemas of the input and target output are not identical, which required some alignment between the two sequences of words and morphemes. In this research, we report the performance of multiple ways of ensemble methods that overcome these obstacles of heterogeneity.

1.2 Motivation and Aim

The field of Arabic Natural Language Processing (NLP) has received many contributions recently. Most morphological analysers handle the morphological-rich problem in Modern Standard Arabic text (MSA), and there are at least seven open access morphological analysers. However, the choice between these taggers is challenging, and there is no open-access tagger explicitly designed for CA to the best of the author's knowledge.

Experiments that used these MSA-based taggers for classical Arabic **reported a significant drop in the accuracy**. Even though the morphology of classical Arabic is the father of MSA, some studies showed that CA texts are not compatible with MSA taggers. Alrabiah (2014) compared two MSA-based taggers both designed for MSA to annotate the KSUCCA classical Arabic corpus. Using five samples from different genres of classical Arabic, an evaluation of these two systems showed a drop in their accuracy by 10-15%. In addition, the semi-annotation of the QAC corpus used an MSA morphological analyser (Buckwalter analyser), but the manual verification step made corrections to at least 24% of words, nearly a quarter of text words, although the text is fully diacritised. A more comprehensive experiment tends to reaffirm similar findings for all MSA taggers (See Chapter 4). These studies show that current taggers might need to be adapted for classical Arabic and their dictionaries need to include a classical lexicon.

For word segmentation and POS labelling, supervised learning has become a dominant model. Its progress is due to the development of annotated corpora and NLP techniques. Although many corpora are released in the literature, obtaining sufficient amounts of high-quality training data remains a major obstacle, especially for morphologically rich languages. Although most of Arabic annotated corpora are for MSA, **not exploiting these related corpora for classical Arabic seems wasteful**. Because underlying linguistic theories differ, annotation schemes for corpora are adversarial, and consequently taggers trained on them. Sadly, although there are multiple resources, it is not possible to merely collate such data for training systems, since almost all existing NLP systems assume a homogeneous annotation. Therefore, it is essential to consider how to use and exploit heterogeneous resources to improve Arabic word annotation and segmentation.

Building a specific tagger or lexicon for CA is expensive and a waste of existing resources. Inspired by the successful results of ensemble methods, specifically (Qiu, Zhao and Huang, 2013; Alabbas and Ramsay, 2014), we decided to pursue the idea of **combining and reusing available morphological taggers to adapt resources in rich languages to under-resourced languages**.

1.3 Research Questions

My research questions are the following:

1. Do MSA-based taggers perform well on CA texts? Can the annotation of CA texts benefit from existing MSA or unsupervised resources?
2. Is it feasible to transfer knowledge from MSA-based taggers to tag classical Arabic texts through combining heterogeneous POS taggers?
3. Does aligning and mapping different segmentation and labelling schemas help ensemble taggers? Can this alignment be learned implicitly?

Chapters 4 and 5 try to answer the first question. It compares and evaluates different taggers on a set of classical Arabic excerpts. The thesis overall illustrates how reusing other resources, e.g. especially diacritised texts and morphological analysers, can help in reducing the ambiguity and help in annotation. The remaining chapters try to reply to the second and third research questions by developing different combinations of strategies and assessing these combinations on a newly

created Sunnah Arabic Corpus. They propose different ways for tackling the annotation-style adaptation.

1.4 Thesis Contributions

In our PhD research, we provide the following contributions:

- **A comprehensive comparison** between open access Arabic POS taggers and morphological analysers with the focus on classical text annotation. This comparative evaluation should ease the choice of a tagger.
- **A novel systematic way of combining multiple heterogeneous tagging algorithms** to achieve improved robustness.
- **An ensemble POS tagger** for classical Arabic from four open access Arabic POS taggers designed for MSA.
- **An open-access semi-automatic annotated Sunnah Arabic corpus** of Hadith collections (a genre of classical Arabic) using the built ensemble tagger and manually verified.
- **An easy-to-use web-based toolkit** that aggregates available morphological analysers and POS taggers. This should ease the usage of those POS taggers for developers.
- **An efficient web-based annotation tool** for semi- and manual- annotation of gold standard corpus which integrates a set of features needed in highly inflectional languages.
- **Arabic multi-tagged corpus, annotated with four POS taggers and aligned to the morpheme-level**. This corpus is useful for evaluation purposes, presenting differences, and possibly learning mappings from one tagger results to another.
- **A novel method for increasing the diacritisation level** of highly cited classical Arabic text for the goal to reduce the word ambiguity level.

1.5 The scope of this research

While this research tried to provide a systematic way of transferring knowledge from any language, the case study in this research is classical Arabic. The results of this research need to be taken cautiously when it is directly applied to other languages. Part of the methodology is tailored to Arabic specifications. An example is the reuse of diacritised texts to reduce the morphological ambiguity.

The results as well have a high correlation with the quality of individual taggers. It is also influenced by the tagger similarity with the required target morphological analysis schema. In this research, the target morphological analysis is based on the traditional Arabic grammar while all individual MSA taggers do not. However, there are evidences of similarity in different aspects (tagset, segmentation, morphological features).

While the research aimed at the beginning to support non-deterministic morphological analysers, they are excluded from this research and the scope is narrowed to only deterministic analysers (*taggers*).

1.6 Thesis Outline

After a brief background in the following chapter, this thesis is divided into three parts: evaluation and classical Arabic adaptation, morphosyntactic ensemble analyser and corpus annotation.

The literature review covers four aspects of this research. It starts with a survey of corpora as they play a critical role in tagging and segmentation. Then it surveys the annotation tools used to create similar corpora that are adapted to Arabic needs. Then it discusses the morphological annotation representation aspects such as tagsets, mapping tagsets, segmentation, etc. Finally, it explores different methods in the literature that combine and exploit heterogeneous annotation.

Then, in the first part of the thesis, open access Arabic taggers and analysers are surveyed, Chapter 4. They are illustrated to contrast their differences using one classical sentence. Then, the results of using several open access MAs and POS taggers to tag classical Arabic are reported. A multi-tagged corpus by several MSA taggers for the Quran is developed that is proofread and manually checked.

The second part introduces the ensemble tagger in more detail. It is divided into three chapters: Chapter 5 describes the challenges of the ensemble method and provides a common ground design for subsequent experiments. It gives the necessary multi-component framework (named SAWAREF) that provides an easy interface for running several taggers, comparing and evaluating between them, and standardising the outputs of each component. Chapter 6 continues the work by delivering concrete methods to tackle the alignment problem and illustrates the effect of this alignment on a pipeline ensemble approach. Chapter 7 moves in

another direction and provides an end-to-end systematic ensemble method for morphological analysing using deep learning.

The third part is divided into two chapters: Chapter 8 provides the design, structure, and annotation of the Sunnah Arabic corpus. It also includes the process of decreasing the word ambiguity level of the original text using a novel method of borrowing diacritisation from similar contexts. It also provides detailed guidelines of the annotation. Chapter 9 presents an open-source web-based annotation tool that aims to increase the annotation speed and consistency of several morphosyntactic annotation tasks by reusing other resources like morphological analysers.

At the end of this thesis, the research is concluded by highlighting the findings and providing a roadmap for future work.

2 BACKGROUND

Chapter Summary:

This chapter aims to provide a brief background on different terminologies discussed in this research. It starts with a brief background on Arabic and its morphology. Then, it discusses the morphological analysis of the language in the computational linguistic point of view and the challenges that face Arabic morphological analysis.

2.1 Introduction

Arabic is a major world language and is one of the six languages officially recognised by the United Nations. It is the first language for around 250-300 million people (Clive Holes, 2004). It is an official language for at least twenty independent Middle Eastern and African countries. It is the language of Islam's holy book: the Quran, and Islam's prophet, Mohammed. Verses attributed to his tradition and sayings, i.e. the Sunnah or Hadith, are also reported in Arabic. Nearly a quarter of the world's population are Muslims, and they use classical Arabic, especially the Quran and the Sunnah, in their prayer and worship.

2.2 Part-of-Speech Tagging and Morphological Analysis

Part of Speech (POS) tagging is a common and well-known problem in the field of Natural Language Processing (NLP). It can be defined as the procedure of identifying the morphosyntactic class for each lexical unit using its structure and contextual information. POS tagging is usually done in the first steps of advanced NLP tasks such as machine translation and text categorisation.

Morphological analysis is a **more general** term that tackles different aspects of the word. It involves the identification of word segments, POS tags, lemma, and morphological features. A morphological analyser (MA) is usually a context-free tool that provides all possible morphological analyses based on a lexicon or dictionary. Morphological analysers may also include a disambiguation component: the solution set are ranked according to the context. In this case, we call such tools *taggers*. The terms POS tagger and morphological analyser are sometimes used interchangeably though. While POS taggers and morphological analysers both analyse the word form (or sometimes its morphemes), POS tagging usually is a more straightforward task that only predicts the POS tag from a set of tags.

2.3 Arabic Language

Arabic and Hebrew are the two most common examples of Semitic languages. Arabic itself contains many different dialects. Arabic is the official language of more than 20 countries, which covers most of the Middle East and North Africa.

Classical Arabic is the "liturgical" language that Muslims around the world use in religious practice. CA is also known as "Fussa" (the clearest), which Arabic

Grammarians build their rules upon. One variant of CA is Quranic Arabic (QA), which is worded from CA, but differs in the sense that it is believed by Muslims to be the direct word of Allah. As time passes, different spoken variants of classical Arabic emerged, and people needed a standard form of communication: Modern Standard Arabic (MSA). MSA is recognised as the formal and standard written Arabic. MSA is the language currently employed in media and education (Bin-Muqbil, 2006).

MSA differs from CA. MSA inherits its syntax, morphology, and phonology from CA; however, MSA's lexicon is much more modern (Habash, 2010) and its stylistics are different (Bin-Muqbil, 2006). CA is not a spoken language (neither is MSA) and is usually found in books and journals. Therefore, it is more *standardised* in the form of writing. Because it is a classical language, CA had *less attention* in the literature and is under-resourced compared with MSA, despite a significant amount of Arabic heritage of ancient books. The classical text is usually grammatically analysed with POS tagsets that are inspired from **traditional Arabic Grammar, Ia'rab** (Elhadj, 2009; Dukes and Habash, 2010; Sawalha, 2011; Elhadj, Abdelali and Ammar, 2014).

Almost all classical Arabic is low-resourced with one exception, the Quran. There are at least 5 corpora that either completely focus on the Quran or at least include it. Because of the Quran's central position in Muslim lives, it grabs more attention. However, in this research, we claim that the Quran is not a fully representative sample of classical Arabic[1]. The Quranic script (a.k.a Uthmani) has a different orthography and lacks some POS tags that normally appear in classical Arabic such as punctuation and numbers.

2.4 Arabic Morphological Analysis

Morphology in linguistics can be defined as the study of the form (internal structure) of the word (Kiraz, 2001). While there is some disagreement in the literature about its definition, this research only cares about morphological analysis in the sense of identification of some meaningful parts and aspects of word structure. Specifically, it includes identifying inflectional and lexical features of word

[1] Neither will the Sunnah corpus be a fully representative sample of classical Arabic. However, it is in the direction of filling the gap of one classical Arabic genre.

segments such as root, stem, affixes, part-of-speech, lemma, pattern, etc. For example, the word (سيضربون /syDrbwn/ (They) will hit) has four meaningful elements /s+y+Drb+wn/: /s/ indicates a future tense, /y/ is third person marker, /Drb/ is the verb "hit", /Drb/ is its root and lemma and / FaCaL/ is its pattern, and /wn/ is a plural marker of the subject. These elements (a.k.a. *morphemes*) are the smallest meaningful units of the word.

Morpheme function can be ***derivational* or *inflectional***. Morphology derivation is the procedure of building new words on the basis of an existing word, e.g. *unstable* and *stableness* are both derived from *stable*. Inflectional morphology, however, changes grammatical features of the same word, e.g. *cats* is the plural form of *cat*. Unlike English which is mostly morphologically concatenative (or linear), Arabic derivational morphology tends to be nonlinear or *templatic* (a root with some vocalism injected into a pattern to form a word), and Arabic inflectional morphology tends to be concatenative. However, there are some exceptions: for example, broken plurals (inflectional) are templatic, and the Nisba phenomenon (a derivation of relative adjective by attaching a suffix Yaa letter) is concatenative (Ryding, 2005, p. 263).

Morphology analysis includes the process of identifying each word's morphemes and extracting their *grammatical features:* including *inflectional* and *lexical* features. A *morpheme* is the minimal unit of the word that carries a meaning. The term *word* is used to represent its orthographical purpose, i.e. one unit of a sentence bounded by two whitespaces. *Tokenisation* is the process of transforming the stream of input characters into a series of words. It includes separating punctuations, grouping digits of one number or date, etc. In contrary, *segmentation* is a more specific form of tokenisation: the morphological process of separating clitics and affixes from the word according to some linguistic theory. Clitics and affixes are not the same: An affix is a morpheme of a word, such as prefixes and suffixes, that attach to a base or a stem while a clitic is a *syntactically independent* morpheme that attaches after affixes (Habash, 2010).

The term ***feature*** is sometimes misleading. In this research, it is mostly used to describe the morphological aspects or characteristics of one word or morpheme, primarily *inflectional* and *lexical* features. It is also, however, later in the thesis used in the Machine Learning sense of describing model parameters or factors.

This research follows the convention of representing the morphological features for each segment of the word, instead of the word. While feature inventories can have an extensive list of features, morphological analysers usually limit the list to *morphosyntactic* features: the features whose values are directly related to the *syntax* of the word (either in *agreement* [التبعية] (e.g. gender agreement between noun and verb) or *government* [العامل والمعمول] (e.g. case (for nouns) and mood (for verbs)).

Morphosyntactic features include:

- *Gender [الجنس]:* a lexical[2] feature for nouns, and inflectional feature for verbs, adjectives, pronouns, etc. It does not necessarily denote the sex of the entity but indicates the grammatical function. Values usually are either masculine or feminine.
- *Number [العدد]:* usually an inflectional feature (although it is sometimes derivational, e.g. broken plural nouns). It denotes the number of persons (even though it is used for non-animate nouns) and usually singular, dual, or plural. In traditional Arabic grammar, more values are defined (e.g. plural of plural) (Sawalha and Atwell, 2013).
- *Definiteness [التعريف]:* inflectional feature for nominals that determine whether they are known or unknown. Usually, definite nouns are prefixed with /Al+/. *Nunation*, the process of adding *Tanween* (/F/,/N/,/K/): a suffix for nominals that is pronounced as an */n/* sound and usually marks nominals as indefinite. However, diptotes (some specific classes of words) are restricted from nunation (Ryding, 2005).
- *Case [إعراب الأسماء]:* inflectional feature for nouns and adjective. It is related to morphology as they are usually marked by a case marker (e.g. a diacritic), and to syntax, as it indicates the role of the noun in the grammar. There are three cases in Arabic: *nominative [مرفوع]*, *genitive [مجرور]*, and *accusative [منصوب]*.
- *Tense/Aspect [نوع الفعل]:* a derivational feature of verbs that has three values: perfect (or past) [ماضي], imperfect (or present) [مضارع] and imperative [أمر]. These types are profoundly influenced by traditional Arabic grammar. Perfect verbs do not necessarily indicate the past occurrence of their actions.

[2] See discussion of lexical features below.

There are more tenses in Arabic, but they are compound tenses. The imperative feature can be describe the mood of the verb (Ryding, 2005) as imperative semantics can be expressed in other ways.

- *Person [نوع الضمير]:* inflectional feature for imperfective verbs and pronouns: first person [المتكلم], second person [المخاطب], and third person [الغائب].
- *Voice [البناء للمعلوم والمجهول]:* derivational feature for verbs and participles to indicate whether the agent of the verb is known (active [مبني للمعلوم، اسم فاعل], or passive [مبني للمجهول، اسم مفعول])
- *Mood [إعراب الأفعال]:* similar to the *case* feature of nouns, the mood is an inflectional feature to determine the mode of the verb: indicative [مرفوع], subjunctive [منصوب], jussive [مجزوم].

In addition to inflectional features, computational morphological analysis usually include identifying some ***lexical* features**. Lexical features are the set of features that describe the meaning of one word regardless of its inflexion (i.e. abstracted from the morphological analysis) but from the language's inventory *(lexicon)*. *A lemma* is a word form that represents a group of word forms that differ only among themselves (Marton, Habash and Rambow, 2010). This group of word forms are called *lexeme*. A *lexeme* is the smallest unit of language that bears some meaning. One example of a lexeme is "*describe"*, and it includes the set of word forms through inflexion: *describing*, *describes*, *described*. In English, the infinitive form of the verb and the singular form of nouns are usually picked as lemmas. So, the lemma is the central representation as it is used in a lexicon.

The most crucial lexical feature is the **core *part-of-speech,*** POS (or lexical category): the category of words in the lexicon that has similar grammatical properties. Because this grouping can be done using different linguistic theories, there is no standard set of POS tags.

Other lexical features include the pattern (either the pattern of the word or the lemma), the root of the word, number of root letters and noun finals. Arabic word roots can be defined as "a relatively invariable discontinuous bound morpheme, represented by two to five phonemes, typically three consonants in a certain order, which interlocks with a pattern to form a stem and which has lexical meaning." (Ryding, 2005, p. 47)

Diacritics or short vowels are some phonological marks that are usually underspecified (not written) in Arabic. *Diacritisation* is the process of adding those

missing marks. Some diacritics are lexical, and some are inflectional and their type usually correlates with their position (usually, last diacritic is inflectional, and others are lexical). Lexical diacritics change the lexical word meaning, while inflectional diacritics change mood, case and voice features. This process is close to morphological analysis as both processes analyse an ambiguous word based on its context and because of the effect of absence/appearance of diacritics on the morphological analysis. It is worth noting that the above Arabic morphology terms are not standardised; and they were sometimes used interchangeably in the literature (Al-Sughaiyer and Al-Kharashi, 2004; Habash, 2010).

2.5 Computational Arabic Morphological Analysis

Arabic morphology analysis is usually essential to Arabic NLP tasks. It is usually done in the first steps of advanced NLP tasks, such as machine translation and text categorisation (Jurafsky and Martin, 2008). It is considered one of the most studied topics in Arabic NLP. Arabic morphology has a high impact on computational tasks.

However, **the level of analysis needed depends on the target goal**. Tasks can be either contextual or non-contextual, analytical or/and generative, and shallow or deep. For example, it may be sufficient for information retrieval (IR) tasks to extract the stem or the lemma of the word. In contrast, traditional statistical machine translation (MT) tasks require thorough morphological analysis (e.g. morphological features play a critical rule such as the gender of the subject and the aspect of the verb).

The **traditional text analysis** pipeline includes tokenisation, POS-tagging, and parsing. The stages of analysis usually start from the surface text, and proceeds through tokenization, lexical analysis, syntactic analysis, semantic analysis and pragmatic analysis with the goal to fully grasp the speaker's intended meaning (Indurkhya and Damerau, 2010). This research is limited to the first two stages. To support the subsequent downstream analyses, it is usually more beneficial to have a fine-grained tagset than a course tagset[3].

[3] In fact, Kübler and Mohamed (2012) shows that tagging using a complex tagset then converting its result to a smaller tagset leads to a higher accuracy than directly tagging using the smaller tagset.

In this research, the task is ***contextual***: the proposed tagger must determine the most likely tag based on the context, ***analytical****:* it only cares about labelling a sequence of text with morphological annotation, and ***deep***: the level of morphological analysis cares about functional morphology and extend core tags to the set of inflectional features.

Computational Arabic morphological analysis techniques can be **classified into four categories**: table-lookup, linguistic (using finite state automaton (FSA) or traditional grammar), combinatorial and pattern-based (Al-Sughaiyer and Al-Kharashi, 2004). *Table-lookup* approaches use a massive database of lexicon and morphology. The *linguistic* approach uses hand-crafted or auto-generated rules to analyse. The *combinatorial* technique determines the morphology by checking combinations of letters against a root list. *Pattern-based* uses the word pattern to find the stem of the word. Table-lookup and linguistic methods suffer from storing and maintaining a high number of inflected forms or rules. Recent advances in the literature seem to be more towards data-driven statistical methods like *combinational* and *pattern-based*. However, these methods require creating costly annotated corpora, which are missing in the case of under-resourced languages.

Computational morphological analysis may involve some of the following **tasks**: *POS tagging*: identifying the morphosyntactic class for each lexical unit using its structure and contextual information; *morphological features prediction*: assigning each word a value of a specific morphological feature (e.g. gender); *segmentation*: finding word segments boundaries; *lemmatisation/stemming*: extracting the lexical origin (*lemma*, *stem*, *pattern*, or radical *root*) of each lexical unit; and *diacritisation:* recovering unspecified lexical and inflectional diacritics (i.e. short vowels) in each word's orthography.

In this research, we interchangeably use *morphological tagger* and *POS tagger* to refer to ***deterministic*** **analysers** that use the context to either choose the most probable tag according to the context or at least provide an ordered list of tags. *Morphological analysers* is a more general term and usually refer to ***non-deterministic*** **analysers**. One more slight but important difference: morphological analysers (non-deterministic analysers) are usually designed to be general-purpose and therefore their tagset is usually rich. Taggers are usually designed for specific purposes and use a reduced tagset for accuracy purposes.

2.6 Computational Linguistic Resources

A number of Arabic linguistic resources are available in the computational linguistic field. They are designed for different purposes: lexicography, Arabic learning, investigating Arabic compounds, language modelling, morphological/syntactic modelling, teaching, speech recognition, and much more. This research focuses on linguistic resources that are related to morphological analysis, namely: morphological modelling, morphological annotated corpora, lexicon, and orthographical annotated corpora.

POS taggers are usually trained on a *corpus. A corpus* (pl. corpora) is "A collection of pieces of language that are selected and ordered according to explicit linguistic criteria in order to be used as a sample of the language" (Sinclair and Ball, 1996, p. 27). The corpus should be *annotated* with POS tags. *Corpus annotation* is "the practice of adding interpretative, especially linguistic, information to a text corpus, by coding added to the electronic representation of the text itself" (Leech and Wilson, 1996, p. 3).

All corpora add some **meta-information (annotation)** for the collected texts. This annotation varies depending on the goal of the corpus from document-level such as marking document's source and author, sentence-level such as parallel bilingual corpora, to word-level or segment-level such as grammatical annotation, i.e. morphological annotated corpora. The more deep the level of annotation, the harder and more tedious the task, and probably the fewer and smaller corpora.

Morphological annotated corpora are usually word-level (sometimes segment-level) annotated. They are useful for several applications such as segmentation, grammatical tagging, diacritisation, lemmatisation, and disambiguation. Annotated corpora vary in the richness of the annotation itself. Usually, they are designed to be rich to give the flexibility to downstream applications. However, this comes with a cost in time and money.

The **annotation of morphological analysis** is usually performed by assigning each word (or a segment of the word) one or a set of tags that represent the different aspects of the morphological analysis. One aspect is the POS tag, and the annotation should have a set of possible POS tags in advance, a.k.a. *POS tagset*. Annotated corpora serve as training datasets for data-driven POS taggers and as evaluation datasets to measure the quality of one tagger.

Other valuable linguistic resources are **lexicons and dictionaries**. In Collins Dictionary, a lexicon is "the set of all the morphemes of a language". In other words, it is the entire inventory of the language lexemes. Dictionaries are a similar resource but are intended usually for human readers. They usually are indexed by the root word where some inflected words can be listed under the root word. Other inflected forms are not listed as the dictionary assumes that the reader has enough grammar on how they will be inflected. In contrast, lexicons are usually indexed by the lexeme where all its inflections are listed.

Note that *the root word* are hard to define, especially in templatic languages. In English dictionaries, words derived from other words are indexed separately. For example, you do not expect to have PLAY and PLAYER in the same entry. Classical Arabic dictionaries, however, usually index the entries using the three radical letters (the root of the word). This indexation abstracts not only inflection morphology but derivational morphology.

Lexicons are useful in morphological tagging, especially for determining lexical features. For example, the gender of nouns is a lexical feature, so lexicons can play a critical role in predicting a word's gender. Because the number of inflected forms in Arabic is high, morphological analysers usually encode the lexicon in different ways (e.g. finite state automaton). Lexicons, however, are not always optimal in terms of coverage, especially for under resourced languages and varieties, such as classical Arabic and dialects.

2.7 Challenges of Arabic Morphology Analysis

Due to the morphologically-rich nature of the language, its highly inflectional, non-linear morphology, and the absence of short vowels (phonological information), the morphological analysis of Arabic is not an easy task. The analysis involves handling an "exceptionally high degree of *ambiguity*" (Soudi *et al.*, 2007).

Arabic is a ***morphologically rich language (MRL)*****,** as illustrated in previous sections. It makes the interaction between syntax and morphology more complicated. As for all MRLs, the rich morphology allows the language to have a considerable degree of freedom in word order as some syntactic relations are expressed in the morphology (Tsarfaty *et al.*, 2010). This phenomenon explains why case and mood features are of central importance in traditional syntactic theory [النحو]. This is a remarkably essential difference in the computational analysis as this

free order property makes algorithms less able to model the language given the same set of examples (Heintz, 2014). Moreover, the case and mood diacritics are usually not written, making such syntactic property ambiguous. Like other MRLs, the actual usage of the free order property in Arabic is less than it could be in principle (Habash, 2010).

As an effect, Arabic is **highly inflectional**. Much of the structural information in Arabic sentences is encoded in inflectional features. Grammatical features usually inflect the word in a concatenative way (prefixes and suffixes) but sometimes in a templatic way. Inflected words do not have an orthographic marker to distinguish affixes. Consequently, the number of possible inflected forms is high, thus the vocabulary size can be enormous (leading to a *data sparseness* problem).

Since there is a high number of inflexions per word, **Arabic's tagsets are usually more extensive** than a typical tagset for English. The size of compound tagsets (that embody morphological features) in Arabic can reach an unusually high number. The Buckwalter tagset, for example, can hypothetically reach over 330,000 tags (Habash, 2010). Tagset size is critical to the process of classification.

POS tagging is typically assigned to each morpheme instead of the whole word as in English (Habash, 2007). Therefore, a pre-processing step of **morphological segmentation** is usually required in order to reduce the data sparsity. This pre-processing step leads to improvement in performance of statistical machine translation (Lee, Papineni and Roukos, 2003; Lee, 2004; Habash and Sadat, 2006).

Written Arabic is highly ambiguous because some **phonological information**, particularly short vowels (diacritics), are not usually written. The short vowels were not introduced in the Arabic orthography system until the 2nd century of prophet Mohammed’s date of migration. In fact, they are still absent in most of the printed and handwritten materials in Arabic. As a result, the same word form can correspond to different possible lexemes.

Concatenating a word with a morpheme sometimes results in the **adjustment** of the original word form (*form adjustment)*. For example, the prefix /l/ precedes the definitive particle /al/, but this resulted on dropping its first letter /a/. This makes the morphological segmentation more complex as it involves morphological awareness of the word.

Lastly, some letters like the Hamza and Yaa letters in Arabic script are **inconsistently spelt**, which increases the ambiguity and sparsity (multiple forms correspond to the same word) (Habash and Sadat, 2006). This problem is not limited to MSA but is also applicable to classical Arabic (Mohamed, 2018).

2.8 Ensemble Tagging

In Collins English Dictionary, "An ensemble of things or people is a group of things or people considered as a whole rather than as separate individuals." (from French word meaning: together). In Machine Learning, ensemble methods refer to the process of combining multiple learning methods to obtain a higher accuracy in classification prediction that was not achieved by any individual learning methods. Instead of relying on one expert decision, ensemble methods tries to make a decision based on the opinions of a collection of experts (Malmasi and Dras, 2018).

The **main goal** of combining classifiers it to have a more accurate classification decision. This comes at the expense of increased complexity. However, the question is "whether a combination of classifiers is justified" (Kuncheva, 2014, p. 101).

There are at least four approaches to building classifier ensembles (see Figure 2.1). Ensembles normally mean multiple learning algorithms trained on the same training data-set, i.e. *homogeneous* ensemble. But in the case of the combination in this thesis, the individual Arabic text taggers which are combined are **not** trained on a common data-set, but separate "black boxes"[5], where we have no control over (or even knowledge of) the training set.

[5] An alternative name for this combination might be "coalition" or "assembly", cf. Collins English Dictionary: "A coalition is a group consisting of people from different political or social groups who are co-operating to achieve a particular aim." "An assembly is a group of people gathered together for a particular purpose. ... The assembly of a machine, device, or object is the process of fitting its different parts together."

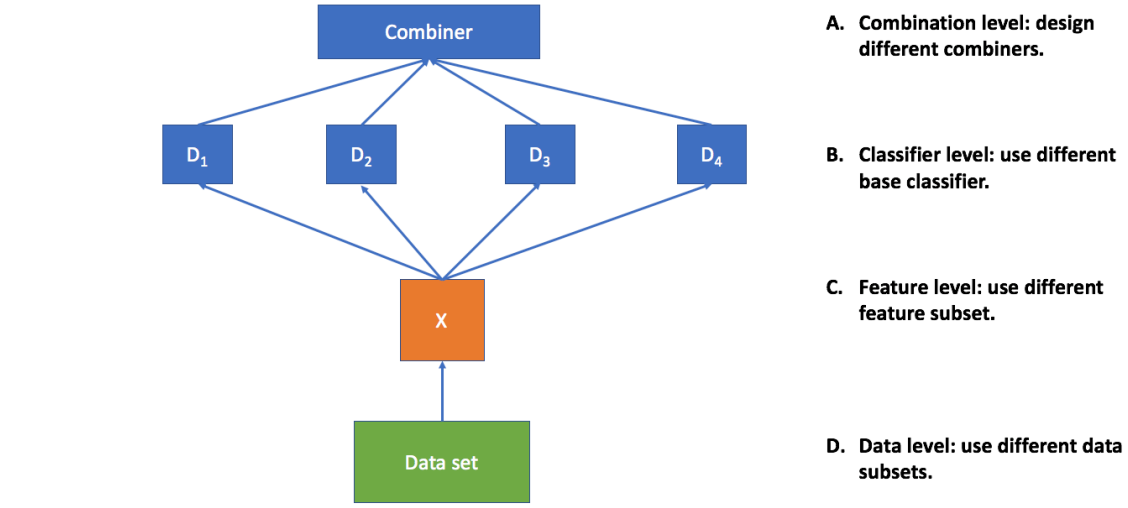


Figure 2.1 Approaches for building classifier homogeneous ensembles, reproduced from (Kuncheva, 2014).

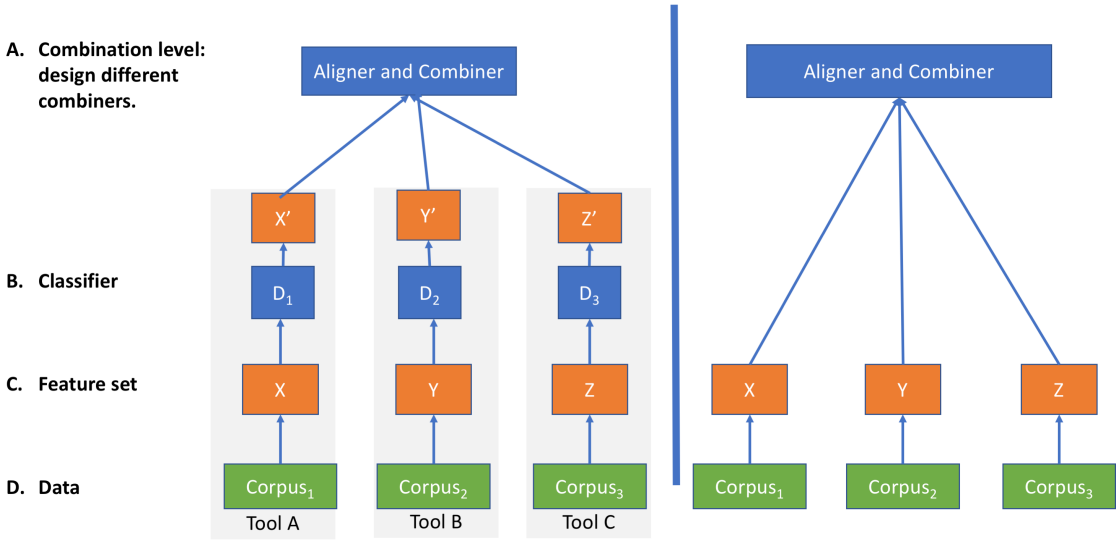


Figure 2.2 Two approaches of building heterogeneous ensembles.

Multiple types of ensemble exist in the literature, mostly for homogeneous methods like bagging (equally-weighted models trained on random subsets of the training data) and boosting (adaptive training where each new model focuses on a subset of training data that was misclassified), including others.

Heterogeneous methods require adaptation or mapping steps before/alongside combining and usually are more complex and prone to errors. Since they use different datasets, the evaluation method for these ensemble cannot be directly compared to original ones. Figure 2.2 listed two approaches to handle

heterogeneous ensembles. The right approach combines jointly adversarial corpora like (Qiu, Zhao and Huang, 2013; Chen, Zhang and Liu, 2016; Chen *et al.*, 2017), while the left approach builds a tagger for each corpus and combine these taggers. Variants of this method is implemented in this thesis and by (Zavrel and Daelemans, 2000; Alabbas and Ramsay, 2012b; Albogamy and Ramsay, 2016).

In POS tagging, different techniques are used, including knowledge-based models: (table lookup, syllable-based morphology, pattern morphology) and empirical methods: (Hidden Markov Models (HMM), Support Vector Machines (SVM), …). Each POS tagger is designed differently. However, without a full understanding of the language, no POS tagger could ensure perfect accuracy. Because of their different bases, taggers will typically produce different errors. Some combinations of POS taggers exploit these differences, and have been reported to achieve a better accuracy for several languages, including Arabic (Alabbas and Ramsay, 2012a; Aliwy, 2015; Zeroual and Lakhouaja, 2017), English (Marquez *et al.*, 1999; Halteren, Zavrel and Daelemans, 2001; Schroder, 2002), Italian (Søgaard, 2009), Icelandic (Henrich, Reuter and Loftsson, 2009), and Swedish (Sjöbergh, 2003).

Most of the combination of POS-taggers in the literature are homogeneous and based on training different models on a common training corpus. Each individual model uses the same tagset and morphological segmentation as the one on the training corpus. However, combining heterogenous black-box taggers, as the approach chosen for this thesis, involves handling different issues, such as mapping taggers' tagsets to one output tagset. The output of those taggers might need to be aligned on the different levels: document, sentence, word, and even morpheme.

2.9 Evaluating Taggers

Several evaluation measures exist for evaluating POS taggers. One of the most common and intuitive measures is the accuracy, the proportion of word forms correctly tagged. This measure requires a method to decide whether a tagging is "correct" or not. For POS tagging, it is common to use a reference corpus that is manually annotated to check the validity of a tagging:

$$Accuracy = \frac{\text{no. correct tags}}{\text{no. reference tags}}$$

However, three conditions need to be met:

- The tagger must use the same tagset used by the reference corpus, otherwise, a mapping to the reference tagset is required;
- The tagger must produce same tokenisation as the reference corpus, otherwise, a re-alignment to the tokenisation in reference corpus is required; and,
- The tagger should output only one tag per token. In case of multiple tags per token, some alternative measurements like ambiguity should be provided.

Alongside the accuracy measure, ambiguity is used to determine the average tags per word emitted by the tagger. It is common to drop reporting ambiguity if the tagger is a single-tag tagger[6], as its ambiguity is one. Ambiguity can be used to measure the difficulty of POS disambiguation:

$$Ambiguity = \frac{\text{no. produced tags}}{\text{no. reference tags}}$$

An alternative method is the use of precision and recall measures inspired by Information Retrieval. It is used for evaluating multiple-tag taggers. Both are usually combined into F1-score that can be balanced or shifted toward precision or recall. We assume here the frequent case where a single tag is assigned to each token in the reference corpus. The recall measure is the proportion of words that have one correct tag, i.e. it is the same as the accuracy. Precision can be seen as the accuracy but punished by ambiguity.

$$Precision = \frac{\text{no. correct tags}}{\text{no. produced tags}} = \frac{\text{no. correct tags}}{\text{ambiguity} * \text{ no. reference tags}} = \frac{\text{Recall}}{\text{ambiguity}}$$

$$Recall = \frac{\text{no. correct tags}}{\text{no. reference tags}}$$

$$F1_{balanced} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

In ensemble classifiers, we needed also to compare the error distribution of different taggers. Precision, recall, accuracy and even ambiguity are global

[6] In this thesis, we use the term *tagger* to refer to single-tag taggers, and *analyser* to refer to multi-tag taggers.

measurements. So, we use the known measurement for measuring the human annotation agreement: (Kappa coefficient) (Carletta, 1996).

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

where p_o is the relative observed agreement among raters (identical to accuracy), and p_e is the hypothetical probability of chance agreement. It can be computed as following:

For classes k, number of samples N and n_{ki} number of times rater *i* predicted class *k*:

$$\mathrm{p_e} = \frac{1}{N^2} \sum_k n_{k1} n_{k2}$$

Please note that this measurement requires that both taggers have the same tagset.

2.10 Conclusion

This chapter gave a brief but essential summary on different background aspects in this research with the aim to define key concepts and help the reader understand challenges specific to the Arabic language. It started by defining and explaining the problem of the research: Part of speech tagging and morphological analysis. Then, it introduced the Arabic language with a focus on the similarities and differences between classical Arabic and modern standard Arabic. We define the scope of the Arabic morphological analysis and associated computational problems and their challenges. After that, we introduce the reader to the concept of ensembles in the sense of machine learning. In the next chapter, we explore related work in four areas: classical Arabic corpora, annotation tools, morphological annotation styles, and automatic annotation methods, with a focus on classical Arabic and ensemble methods.

3 LITERATURE REVIEW

Chapter Summary:

This chapter surveys existing morphological analysis methods with a focus on ensemble methods and classical Arabic. It starts by exploring existing Arabic corpora, especially morphologically and orthographically annotated corpora, as these corpora serve as a basic requirement for data-driven morphological analysis methods. Next, it explores and evaluates existing tools for annotating corpora, with a focus on Arabic needs and requirements. Third, it investigates morphological annotation representations in the literature with a focus on adapting between different representations by methods such as mapping and alignment. Finally, it examines the computational methods for segmentation, tagging, and diacritisation with a focus on ensemble methods and how these methods are evaluated.

3.1 Introduction

The computational analysis of the Arabic language started in the 1980s. The morphological aspect of the language is an ongoing research theme, especially on low-resource variants like classical Arabic and dialects. Adapting and reusing existing resources to a new domain or language has shown advantages in many fields.

This chapter surveys existing morphological analysis methods with a focus on ensemble methods and classical Arabic. It starts by exploring existing Arabic corpora, especially morphologically and orthographically annotated corpora, as these corpora serve as basic requirement for data-driven morphological analysis methods. Next, it explores and evaluates existing tools for annotating corpora, with a focus on Arabic needs and requirements. Third, it investigates morphological annotation representations in the literature with a focus on adapting between different representations by methods like mapping and alignment. Finally, it examines the computational methods for segmentation, tagging, and diacritisation with a focus on ensemble methods and how these methods are evaluated.

Text corpora forms the basis for developing data-driven computational models of one language. These corpora are designed to be representative samples of one aspect of its language, e.g. its morphology. Treebanks, or morphologically annotated corpora, are usually annotated to the word-level with grammatical categories, lemma, and various grammatical features. Classical Arabic, in particular, faces a lack in these type of valuable resources unlike its more modern variant: MSA. The first part of this chapter presents a systematic review of the literature of these corpora.

The next section focuses on tools for manual annotation of treebanks in general and morphological annotation specifically. These tools aim at speeding up the repetitve task of annotation by reusing predefined annotations with no compromise on the quality and consistency. General methods exist, but Arabic language and highly inflectional languages in general require more features that are sometimes not implemented in general-purpose frameworks.

Computational annotation of written texts, POS tagging or morphological annotation in particular, adds a layer to the text that is valuable to many downstream processes in the field of Natural Language Understanding. This layer describes the grammatical role of words for the purpose of a better understanding of the whole

sentence. Different linguistic bases lead to different annotation schemas. Because of the expensive and tedious characteristic of annotation, several methods that try to adapt existing annotation to other languages exist. The third part surveys these Arabic annotation schemas and review efforts to map and standardise tagsets and align incompatible segmentation schemas.

The forth part of this chapter surveys existing computational systems for morphological and orthographical annotation. It pays more attention on efforts to combine or adapt several taggers, especially heterogeneous taggers. Existing open access taggers are evaluated in detail in Chapter 4, for the purpose of selecting and using the best ones in our ensemble approach.

3.2 Arabic Corpora

3.2.1 Corpora

A number of Arabic corpora are available in the computational linguistic field. They are designed for different purposes: lexicography, Arabic learning, investigating Arabic compounds, language modelling, morphological/syntactic modelling, teaching, speech recognition, and much more. This survey focuses on corpora that are related to morphological analysis, namely: morphological modelling, morphological annotated corpora, lexicon, and orthographical annotated corpora.

Most existing Arabic corpora are for written texts. There are few corpora that are designed for spoken languages, although there is a recent shift in focus in the research toward dialectics. The source of written texts is mostly newswire and the web, and most of these corpora are not open-access and not freely downloadable (Sawalha, 2011).

MSA Corpora form the majority of these corpora. Classical Arabic has recently grabbed attention in the corpus creation field, with most work on the Quranic texts. The most prominent resource for classical Arabic is the Shamela Library. Shamela (http://shamela.ws) is a freely downloadable electronic library that contains at least 5300 Arabic books in Islamic studies and has become the standard e-library of Arabic classical books. It has been used to obtain Arabic classical text in building several corpora for different purposes: language modelling corpora (Alrabiah, 2014; Belinkov *et al.*, 2016), orthographic modelling corpora (Zerrouki

and Balla, 2017; Alosaimy and Atwell, 2018), and morphological modelling corpora (Mohamed, 2012; Alosaimy and Atwell, 2017).

3.2.2 Morphologically annotated Corpora

In regards to Modern Standard Arabic, there are several existing corpora, including:

- **Khoja POS tagged corpus**, 50,000 words of newspaper text with simple POS tags, and 1700 words with detailed POS tags.(Khoja, 2001)
- **The Penn Arabic Treebank (PATB),** one million tokens annotated with part of speech (POS), gloss, diacritisation and word segmentation. (Maamouri *et al.*, 2005)
- **Prague Arabic Dependency Treebank (PADT)** (Hajic *et al.*, 2004), morphologically annotated 113,500-tokens newswire texts.
- **Columbia Arabic Treebank (CaTiB)** (Habash and Roth, 2009). 273,000 tokens annotated plus 735,000 automatically converted from PATB; collectively 1M tokens of newswire.
- **Nemlar Written Corpus** (Yaseen *et al.*, 2006)**.** Half-million words in a balanced corpus of 13 genres where the time span goes from late 1990's to 2005.
- **AQMAR dependency corpus** composed of 36,000 words of 10 Arabic Wikipedia articles tagged using CaTiB tagset (Schneider *et al.*, 2012).

One particular treebank is highly influencing the field of Arabic morphological analysis: the PATB treebank. It not only has a large amount of annotated texts, but its level of annotation is magnificent: the texts are segmented and each segment is diacritised, lemmatised and labelled with its complex POS tag. Although PADT treebank has a similar rich annotation, PATB treebank has been cited more in the literature maybe due to its larger size. However, the fact that they do not conform to one standard annotation schema limits their use in a combined corpus.

Universal Dependency (UD) project is a framework that aims to provide treebanks in different languages with cross-linguistically consistent grammatical annotation. There are three Arabic treebanks in UD: A converted PADT treebank to UD standards, NYUAD treebank that is based on the PATB treebank but converted to CaTiB annotation schema in UD format, and a newly released 20K-words Arabic-part of the Parallel UD (PUD) project. Aside from their fine-grained tagsets, the

three treebanks share the same coarse tagset (12-tag UD tagset) and morphological features. In fact, the comparative statistics of Arabic treebanks published in their website shows some differences such as the lemma definition.

As Albared, Omar and Ab Aziz (2009) pointed out: most corpora available are derived from newspapers. Moreover, each corpus used its own tagset and morphological segmentation scheme making it difficult to ensemble them into one training dataset, which could lead to a better accuracy (Banko and Brill, 2001).

Classical Arabic on the contrary is low-resourced, especially in manually annotated corpora. They are usually attached with POS tagsets that are inspired from traditional Arabic Grammar, Ia'rab (Elhadj, 2009; Dukes and Habash, 2010; Sawalha, 2011; Elhadj, Abdelali and Ammar, 2014; Zeroual and Lakhouaja, 2016; Alosaimy and Atwell, 2017).

In regards to classical Arabic corpora, there are six annotated corpora as follows:

1. The morphological analysis of the Holy Qur'an by Al-Imam University (Elhadj *et al.*, 2010)

This project provides an indexed Quran text database of morphological segmentation which has been done according to linguistic terms and rules. Each and every word of the Holy Quran has been split into a prefix, a root, a stem and a suffix, and then stored in a 4-column table. It was part of a larger project that involves a search engine for similar pronunciation of words. The project as well provides a manually verified text of the Quran that is written according to modern orthography.

2. The Quranic Arabic Corpus (QAC)[1] (Dukes, Atwell and Habash, 2013):

Developed at the University of Leeds, the QAC corpus is a morpheme-based corpus that is fine-grained annotated with grammatical and syntactical annotation. In addition to segmenting each word to its morpheme, each morpheme is annotated with its POS tag, root, lemma and a set of morphological features.

The corpus covers the whole holy book which is 77,430 words. After manually segmenting each word, the total number of segments is 128,220, where each segment is given one POS tag out of about 45 tags, and a set of lexical and grammatical features that includes each word's lemma and root (assigned to its stem) and eight grammatical features. In addition, various information about some

[1] http://corpus.quran.com/

types of words is given to define a finer group of POS like verb form, noun derivation, special groups like the verb */kaana/* and its sisters.

The process of developing this corpus is well defined. It started by analysing each word with a non-disambiguated list of possible analysis extracted from the Buckwalter morphological analyser filtered to keep only analyses that match the diacritised form. Some orthographic processes are required to convert the Quranic script to the modern script expected by the analyser. Only 87% of verbs are analysed by the morphological analyser. Then two paid annotators were assigned the task of selecting and correcting the tagging according to the morphological guidelines and tagset in two rounds. The first one completed the analysis of remaining words and corrected 13% of words incorrectly analysed, i.e. only 76% of words are correctly annotated by the morphological analyser. The second annotator reviewed the corrected version and made changes to 1.3% of the words. Users of the corpus corrected about 2.5% of words within the first six months of corpus release.

3. SALMA Annotated Quranic Text (Sawalha, 2011)

For the purpose of demonstrating the Standard Arabic Linguistics Morphological Analysis tagset (SALMA), Sawalha and Atwell (2013) developed the Gold Standard of Arabic - Quranic text (GSA-Q). They fully annotated the 29th chapter of the Quran, where each word form is annotated with its root, lemma, pattern, long stem and its morphemes tagged with its part-of-speech and sixteen morphological features: gender, number, person, inflectional morphology, case or mood, case or mood marks, definiteness, voice, emphasising, transitivity, rational, declension and conjugation, augmentation, number of root letters, verb root type and noun finals. The corpus is publicly available[2].

4. Emad's Heritage Corpus (Mohamed, 2012, 2018)

The Heritage corpus is a recently published corpus of classical Arabic that covers various genres of classical texts. The total number of annotated text increased from 27k word of religious texts in 2012 to 58k words of broad classical texts in 2018. The text covers several topics that include: the Quran, Sunnah, Islamic law, literature, philosophy, and psychology. The corpus will be publicly available.

The text is annotated with the PATB tagset, which allows the comparison of a tagger trained on the corpus and the one trained on the PATB corpus. The tagset

[2] http://www.comp.leeds.ac.uk/sawalha/

used is a complex tag (morphological features are embodied) and has 133 segment-level tags and 949 word-level compound tags.

The corpus was done in one round by the first author in an iterative way. The first 2k words are initially tagged using a tagger trained on the PATB treebank and then proofread and corrected. Then, for each proofread 2k words, they are added to the PATB to create a new more-accurate model, and so on. The initial accuracy of using only PATB to tag the corpus is 78.62%. This is due to the high rate of out of vocabulary (OOV) words (43.39%) and the domain difference (Mohamed, 2012).

The author also developed a classical Arabic tagger based on this corpus using the TiMBL toolkit, a memory-based learning toolkit. The accuracies of full automatic segmentation and POS tagging on development and test datasets are 89.8% and 87.8%, respectively.

5. Evaluation Set of Joint Tagging and Parsing (Zhang et al., 2015)

A classical Arabic dataset is mentioned in (Zhang *et al.*, 2015), where its texts are obtained from the Shamela library and segmented and tagged by a computational linguist. No clear mention of the used tagset is provided, nor its size; however, the paper claims that the dataset is available at the Farasa website, but we could not find a link to it.

The dataset size is 7.9k of words and 163 sentences. The dataset was used for testing their joint parser trained on the MSA treebank, and showed that incorporating syntactic information reduced the error rate significantly, especially for OOV words.

6. Al-Mus'haf Corpus (Zeroual and Lakhouaja, 2016)

Al-Mus'haf is a new annotated corpus of the whole Quran that focuses more on lexical features and uses a tagset that is more influenced by the traditional Arabic grammar, Ia'rab. It covers all words of the Quran (~78k words) and tags each word with a rich POS tag, lemma, stem and root. It does not specify the affixes of the root, though.

The annotation is done semi-automatically using the AlKhalil morphosyntactic analyser (Boudchiche *et al.*, 2016). Since AlKhalil is a non-deterministic analyser, a further treatment of its output is required. Experts in Arabic morphological rules verified the results and completed non-analysed cases. Alkhalil was able to tag a word with one analysis in 71% of cases. Other cases required either

correction or disambiguation. The authors did not report the number of cases where multi-analyses are incorrect.

7. Non-verified Corpora

Corpora that collect classical text usually add a layer of automatic morphological annotation. Alrabiah et al. (2014) built the publicly available[3] general-purpose King Saud University Corpus of Classical Arabic (KSUCCA). The 50-million-word corpus was designed originally for studying the distribution of lexical semantics. The corpus was automatically POS-tagged using the MADA 3.2 toolkit. The corpus combines different genres: religion 45%, literature 15%, linguistics 13%, science 12%, biography 7%, and sociology 5%. In a similar approach, Belinkov et al. (2016) developed 1-billion words of classical Arabic drawn as well from the Shamela Library with a focus on diachronic information of the texts. It has been annotated using the MADAMIRA toolkit and is available publicly but without the morphological annotation[4].

Table 3.1 summarises the annotated classical Arabic corpora. Most annotation is done to the Quranic text. Tagsets of these corpora are not the same nor the segmentation schemas, which complicates the combination of these corpora into one standard bigger corpus. With the exception of unverified corpora and Emad's work, all other works are done using tagsets that are influenced by the traditional Arabic grammar. We noticed as well that the Sunnah texts are not annotated except for a small part of Emad's work, although the Sunnah is the second major source of Islamic law and guidance. Most presented corpora are done semi-automatically, as morphological analysers usually speed up the annotation greatly. However, the number of needed corrections in these corpora can give us a measure of how well these analysers fit to the classical Arabic. The percentage of corrections ranges around ~25% of words, which is quite high, although the Quranic text is fully diacritised.

[3] https://mahaalrabiah.wordpress.com/2014/06/07/the-annotated-ksucca/

[4] https://github.com/OpenArabic/

Table 3.1 Summary of classical Arabic corpora

Name	Reference	Texts	Word #	Tagset	Downloadable	Verified
Imam	(Elhadj *et al.*, 2010)	Quran	77k	Only segmentation corpora	No	Yes
SALMA	(Sawalha, 2011)	Quran	1k	(Sawalha and Atwell, 2013)	Yes	Yes
Religious	(Mohamed, 2012)	Quran, Sunnah, Philosophy	27k	(Maamouri and Bies, 2004)	No	Yes
QAC	(Dukes, Atwell and Habash, 2013)	Quran	77k	(Dukes and Habash, 2010)	Yes	Yes
Eval Set	(Zhang *et al.*, 2015)	N/A	7.9k	(Maamouri and Bies, 2004)[1]	No	Yes
Al-Mus'haf	(Zeroual and Lakhouaja, 2016)	Quran	78k	(Zeroual, Lakhouaja and Belahbib, 2017)	Yes	Yes
Heritage[2]	(Mohamed, 2018)	Five-geners	58k	(Maamouri and Bies, 2004)	No	Yes
Alrabiah	(Alrabiah *et al.*, 2014)	General	50m	(Habash, Rambow and Roth, 2009)	Yes	No
Shamela	(Belinkov *et al.*, 2016)	General	1bn	(Pasha *et al.*, 2014)	Yes	No

[1] It is not mentioned in the paper. However, since it is used for evaluating a trained model of MSA text annotated on PATB, we assume that it is annotated using the same tagset.

[2] This is an expanded corpus of the Religious corpus developed by the same author.

3.2.3 Orthographically annotated Corpora: Diacritised Corpora

Corpora may also be annotated by adding diacritics to the word form, which is useful in reducing the ambiguity of the word in meaning and grammatical category and features. This type of annotation can be seen as one type of natural rewritings corpora, e.g. misspelling corpora; however, under-specification of word forms in Arabic is not a mistake as it is a common practice. Natural rewritings corpora are usually helpful in NLP tasks such as text correction, paraphrasing, summarisation, and text normalisation.

This section focuses on diacritised corpora as it is highly related to morphological annotation. One unique aspect of classical Arabic texts is that they are often diacritised. This added specification was not done for no reason: diacritisation should help the reader disambiguate each word by looking at its diacritics. This disambiguation is needed more for classical texts, and in particular religious texts where correct interpretation is much needed.

However, works that focus on MSA texts generally ignore the diacritisation completely. It is common to normalise the text by removing all diacritics as they only contribute to increasing the sparsity of words. In classical Arabic, this information should be integrated in the morphological analysis. This section reviews the available corpora and sources for diacritised texts.

1. PATB Treebank (Maamouri and Bies, 2004)

Rich-annotated treebanks are one source of diacritised corpora (in particular: PATB treebank) (Maamouri and Bies, 2004). The PATB treebank is one million tokens (~750k words) annotated with part of speech (POS), gloss, and word segmentation; however, not all words are fully diacritised.

In fact, the diacritisation on the PATB treebank has passed through different decisions. The first corpus was lexically diacritised, with no case marks for nouns, no voice nor mood marks for verbs. The second corpus diacritisation considered the case and voice marks and was governed through some guidelines that allow a consistent annotation schema. The third corpus added mood for verbs.

This treebank has been used in many diacritisers as a training and testing dataset. However, the dataset is only available through an expensive membership of

the Linguistic Data Consortium (LCD). Its data is newswire and all the text is in MSA.

2. OptDiac Project (Zaghouani et al., 2015)

OptDiac stands for Optimal Diacritisation Scheme for Arabic Orthographic Representation. The project aims to improve readability and comprehension rates for Arabic text through NLP. It is the only annotation project that is dedicated to diacritisation. It proposes different schemas for partial diacritisation for the purpose of achieving optimal readability scores. One contribution of the project is the annotation of the Corpus of Contemporary Arabic (CCA) (Al-Sulaiti and Atwell, 2006), a balanced 1-million words corpus of MSA texts. There is no clear mention of the availability of the annotated corpora nor the licence.

3. Tashkeela (Zerrouki and Balla, 2017)

Tashkeela is a corpus of 75 million words semi-automatically extracted from several sources. Classical Arabic constitutes about 98% to the corpus, with 97 books extracted from the Shamela Library. The estimated average number of diacritics per word is 2.05, an indicator of partially diacritised texts. The process of text selection is basic and does not ensure that all texts in that book is diacritised.

Although MSA orthography is largely standardised (Habash, Diab and Rambow, 2012), the presented corpora cannot be assumed consistent because of four reasons:

1. The level of diacritisation varies: fully (every single letter), semi-fully (except deterministic letters), and partially.
2. The schema for diacritisation may differ which affects how one letter is diacritised: e.g. position of nunation and the diacritisation of the final letter proceeded by a vowel-starting word.
3. Some corpora truncate syntactic vowels (i.e. last short vowels) for the purpose of keeping only lexical diacritics. An automatic process usually results in inconsistent results, e.g. when word is fused with a suffix.
4. Some lemmas can have multiple correct lexical diacritisations (this does not include case marks nor mood marks). For example, (*/>SbE/*, “finger”) can be diacritised in eight ways (Mandhour, 1994): */<iSobaE/, /<iSobiE/, /<iSobuE/, />aSobiE/, />aSobuE/, />uSobaE/, />uSobiE/, />uSobuE/.*

3.3 Annotation Tools

Recent research developments in, and uses of, Arabic annotated corpora were the main inspiration behind building a new tool for manual annotation. These corpora play a growing role in some linguistic and computational research areas such as part-of-speech tagging, segmentation, and diacritisation. Additionally, the need of a freely available annotated corpus of classical Arabic increases the importance, which may encourage researchers to conduct more studies in the aforementioned research areas.

Annotation tools play a critical role in the development of annotated resources. Annotation is known to be tedious; but because it is done by humans, it is prone to errors. All tools should aim to be *efficient* in terms of time and accuracy. The annotation of Arabic text is even more tedious and time-consuming than its equivalent in morphologically-poor languages, as the annotation richness is usually higher.

Morphosyntactic annotation of highly inflectional language corpora requires additional specialised functionality:

1. Segmentation of one word into a set of segments
2. Addition of orthographical accents or diacritics
3. Listing a set of solutions from a lexicon dictionary (internally or externally using a morphological analyser)
4. Consistency validation and integrating annotation guidelines (e.g. homographs).
5. Adaptive prediction based on historical tagging
6. Efficient keyboard-based navigation and labelling

In this literature review, we focus on four aspects, i.e. tools that:

1. Are open access and available to download for research purposes.
2. Are web-based: to integrate it with other systems, and to allow easier access through browsers.
3. Annotate text tokens with morpho-syntactic tags in CoNLL-U v.2 format[1].

[1] CoNLL-U format has been used in the Universal Dependencies project (Nivre *et al.*, 2017), and is described in detail on their website (http://universaldependencies.org/). The choice of this format is to constrain tools that: do not allow morpheme-based annotation, do not restore adjusted-form, and do not have POS+features representation.

4. Support right-to-left languages.

Annotation tools can be classified in two categories. General-purpose tools aim to provide a single framework to all annotation tasks of one text and support different languages. Task-specific tools aim to give specific features for the annotation of one layer, e.g. morphological annotation or to specific features of one languages, e.g. Arabic. The first usually support a variety of file formats, while the second may not. We noticed that task-specific tools are usually done in research groups to suit their needs, and are usually not available.

3.3.1 General Annotation Tools

These annotation tools are not designed for a specific language. In addition, they aim to support a range of annotation tasks. The summary of each tool's support of our set of criteria is shown in Table 3.2.

1. Brat Annotation Tool (2012)

The Brat tool (Stenetorp *et al.*, 2012) is a generic tool that has an excellent visualisation component for syntactic annotation. It has a morpho-syntactic annotation layer as well, but it suffers greatly from not supporting right-to-left languages. We can use transliterated Arabic instead, but it is still sub-optimal.

2. WebAnno (2013)

WebAnno (Yimam *et al.*, 2014) is a set of well-documented tools for multiple annotation tasks. It uses brat annotation for visualisation and supports RTL languages as well. However, it does not allow changing or inserting nodes to the basic layer and assumes that input is a gold-standard segmented text. Moreover, a number of clicks are required to just change one element's information.

3. Arborator (2013)

Arborator (Gerdes, 2013) a dependency annotation tool that supports RTL languages. One significant advantage of this is the synchronisation between the CoNLL-U and the visualisation, allowing the annotator to edit CoNLL-U text and check the result in the visualisation. It has a simple drag-and-drop interface for syntactic relations editing. However, it is not well documented, and it is more suited for syntactic annotation than morphological annotation.

4. CorA (2014)

CorA (Bollmann *et al.*, 2014) is a web-based tool publicly available[2] for morpho-syntactic annotation of non-standard texts. It offers token-based annotation of lemmatisation, POS tags and morphological features in addition to normalisation and modernisation. The modernisation layer can be used for diacritisation in our case. A significant advantage of this tool is its support of immediate retraining of taggers on newly annotated data. The tool assumes tokenised morphemes as input, and does not allow the annotator to segment on the fly.

Table 3.2 Comparative analysis of open access annotation tools.

Features	Brat	WebA	Arb	CorA
Segment one word into segments.	✓			
Diacritics		✓	✓	✓
Suggest a set of solutions from a lexicon dictionary				
Consistency validation		✓		
Adaptive predicting based on historical tagging		✓		✓
Efficient Keyboard-based navigation and labelling		✓		

3.3.2 Arabic Morphological Annotation Tools

1. Fassieh (2009)

Fassieh (Attia, Rashwan and Al-Badrashiny, 2009) is a tool used internally in the RDI company and is not available for download. It was used in the development and annotation of the NEMLAR written corpus. It aims to provide a one-for-all framework for all types of annotations including morphological, POS tagging, phonetic (diacritisation), and semantic annotation. The annotation tool is one part of the whole system which includes diacritiser, tagger and segmenter. It is a standalone application that shows the context of the sentence and provides a set of possible analyses in a tabular format. The features of annotation tools like searching, output format and consistency checking is not specified in the article. The tool seems to only work with RDI settings, including the tagset and taggers.

2. SAWT (2016)

[2] https://github.com/comphist/cora

Sequence Annotation Web Tool (Samih, Maier and Kallmeyer, 2016) is a web-based tool for the annotation of token sequences with an arbitrary set of labels (e.g. POS tags). It is simple and efficient but suffers from segmentation assumption as well. It is yet not available.

3. MADARi (2018)

MADARi (Obeid *et al.*, 2018) is a web-based annotation tool for morphological analysis with an emphasis on spelling corrections. The authors target annotating the MADAR project, a multi-Arabic dialect corpus (Bouamor *et al.*, 2018). Arabic dialects are often misspelt or at least do not conform to standards. They plan to release the tool and make it available but no specific timeline or licence is stated. The tool as well does not support CoNLL-U format. Specifically, it does not support morpheme-based annotation. Although this tool does not match our criteria, this tool is recently published (concurrently with Wasim tool (introduced in Chapter 9) in LREC 2018 conference) and have common tasks and needs with Wasim.

3.4 Morphological Annotation Representation

3.4.1 Tagsets

Arabic traditional grammar, and school textbooks, state that there are three POS tags: nouns, verbs and particles. This classification is criticised for being too coarse and confusing. It does not state how to define the three categories and how to handle borderline cases. Instead the classification sometimes was based on examples; for example, Sibawayh, the father of traditional grammar, introduced this classification by saying (translated from Arabic): “the speech consists of nouns, verbs or particles which are not nouns nor verbs. A noun is like a man, a horse, a farm” (Sibawayh, 1988, p. 12). But later scholars tried to define this classification by stating some features like only nouns can have nunation and the definitive article. Traditional scholars were not unanimous on the tripartite classification and some introduced a forth class, e.g. Abdel Qahir (Al-saqi, 1975).

The most famous modern classification is the one introduced by Tamam Hassan (Hassan, 1994) and his PhD student (Al-saqi, 1975). They proposed a two-dimensional morphological analysis system where the first dimension is POS tags, and the second dimension is the morphological features. The POS tagset has two layers and consists of seven main categories: nouns, verbs, adjectives, adverbs,

pronouns, particles and interjections. Each category has its own subcategories. This new tagset is based on two principles: word form and function, which can be seen as the word features that should be used for classifying one word. Word form considers the word's ability to have some grammatical marks (Ia'rab marks, e.g. case and mood marks), its order, possible pattern, inflections system, coupling and its orthography. Word function includes morphological and grammatical functions like naming, action, tense, dependency, and grammatical meaning. Main categories have to have a difference not only in form or in function, but in both principles. This classification is not adopted in computational linguistics; however, Al-jundi (2016) tried to map some existing resources to this tagset and claim that it is possible. Traditional classifications, and Tamam's in particular, lack the evaluation of proposed systems: e.g. there is no tagged corpus with such a classification.

Arabic tagsets in computational linguistics can be divided into two groups: traditional and "English-centric". English-based tagsets (Hajic *et al.*, 2004; Maamouri and Bies, 2004; Diab, 2007) emerge when resources for Arabic was limited, and an agreed-upon tagset is required for resource development. These tagsets use a minor modified tagset from standard English. However, as Wintner (2014) stated: "Such an adaptation is problematic for Semitic languages". For example, the distinction between adjectives and nouns is blurry (See 5.7). Unlike English, they have common morphological properties, e.g. inflection, which suggest they could be a subcategories of *nominal* (Wintner, 2014). On the other hand, traditional tagsets follow the long history of morphological studies that spans fourteen years in their names and classification. They usually are central to the explanation of grammatical marks (case and mood). Several traditional-based tagsets are proposed (Khoja, 2001; Dukes and Habash, 2010; Sawalha, Atwell and Abushariah, 2013; Elhadj, Abdelali and Ammar, 2014; Zeroual, Lakhouaja and Belahbib, 2017). However, the main challenge is the construction of a language resource, e.g. an annotated corpus, which is fundamental in computation linguistics. Since many tagsets are introduced in the literature, we will focus on tagsets used in classical corpora.

Classical corpora, in general, do not conform to a standard tagset and are relatively small. As shown in Table 3.1, almost every corpus has its own tagset and the largest annotated corpus is the Quran, ~77k words. This might be due to the lack of commercial interest and the limited uses of such annotation. However, religious

corpora, e.g. the Quranic Arabic Corpus, demonstrate their usefulness not only for NLP studies, but for end users who use the tagged corpus for the purpose of understanding the Quran.

1. **Khoja's Tagset (Khoja, 2001)**

   It is a complex hierarchical tagset that is based on traditional Arabic grammar. Figure 3.1 shows the list of the main POS tags. The hierarchical aspect of the tagset implies inheritance such that all subclasses inherit properties from parent classes. There is a 1.7k-words newswire MSA corpus that uses this tagset.

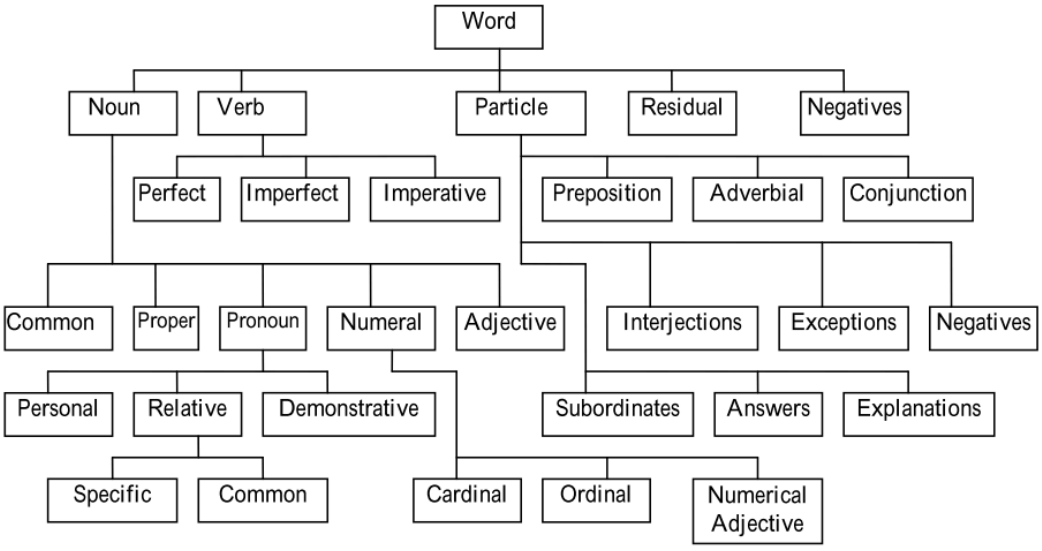


Figure 3.1 Khoja's Tagset, taken from (Aliwy, 2013).

2. **Penn Arabic Treebank (Buckwalter) Tagset (Maamouri and Bies, 2004)**

   The tagset used in PATB is the most widely used (Sawalha and Atwell, 2013), and has been recently applied to classical texts (Mohamed, 2018). The tagset, which consist of ~70 basic tags, is the untokenised version of the Buckwalter morphological analyser. Because this tagset is rich, several reduced tagsets emerged, e.g. (Diab, 2007). Figure 3.2 shows the token-based basic tagset.

ABBREV	**IV2FP**	**PART**
ADJ	**IV2FS**	**POSS_PRON**
ADJP	**IV2MP**	**PREP**
ADV	**IV2MS**	**PRON**
CASE	**IV3FD**	**PUNC**
CONJ	**IV3FP**	**PVSUFF_DO**
CVSUFF	**IV3FS**	**PVSUFF_SUBJ**

DEM	IV3MD	REL_ADV
DET	IV3MP	REL_PRON
EMPHATIC_PART	IV3MS	RESULT_CLAUSE_PARTICLE
EXCEPT_PART	IVSUFF	SUBJUNC
FUT	LATIN	SUB_CONJ
FUT_PART	NEG_PART	VERB
INTERJ	NOUN	VERB_IMPERATIVE
INTERROG_PART	NOUN_PROP	VERB_IMPERFECT
IV1P	NO_FUNC	VERB_PART
IV1S	NSUFF	VERB_PASSIVE
IV2D	NUM	VERB_PERFECT

Figure 3.2 PATB Tagset[3].

3. **Quranic Arabic Corpus Tagset** (Dukes and Habash, 2010)

   QAC tagset is two-dimensional and used to tag the Quranic texts, so the tagset is tailored to Quranic texts. It is designed to capture long-established traditional Arabic grammar, Ia'rab. The first dimension has ~45 tags: nine tags for nominals, one for verbs, 34 tags for particles and one for Quranic initials. The second dimension represent affixes and morphological features including gender, person, number, aspect, mood, verb form, state, case, derivation and voice. The tagset is published online[4]. This tagset is discussed in detail when we introduce the Sunnah Arabic corpus (see Chapter 8).

4. **SALMA tagset** (Sawalha and Atwell, 2013)

   SALMA is the most fine-grained tagset in two dimensions: the number of features (~ 15 features) and the number of possible tags of each word (~ 91 distinct tags). The POS tags has two levels where the first level is the three traditional categories (noun, verb, particle) plus two categories: affixes, and punctuations. The SALMA tagset has thirty-four possible tags for nouns, three for verbs (which matches the aspect feature in other tagsets), twenty-two for particles, twenty for others, and twelve for punctuations. It is the

[3] Each part has a slightly different tagset. This tagset is for Part 2. The original tagset (with compound tags, with morphological features) can be found in: https://catalog.ldc.upenn.edu/docs/LDC2004T02/

[4] http://corpus.quran.com/documentation/tagset.jsp

most finely-grained tagset in Arabic regarding tagset size and feature set size.

The tagset has been applied to two small corpora (~1000 words each): MSA and Quranic. While this tagset is rich and follows the traditional Arabic grammar, it does not define the characteristics of each tag. This evaluation is based on the feedback from two linguists in one experiment (see Section 5.5). This results in many borderline cases, e.g. some particles can belong to two categories like (/mn/ *from*).

3.4.2 Mapping of tagsets

Mapping between tagsets is useful in reusing and accessing existing heterogeneous annotated corpora. It is one of the first attempts to exploit the existing heterogeneous corpora and collate them into one big dataset which can increase the quality of training for statistical methods. Also, it is useful for standardising corpora with different annotation schemas. There will be no need to know and memorise each corpus tagset.

Mapping from one tagset to another tagset has been adopted in many applications. It has been adopted to achieve better accuracy by reducing tagset size (Brants, 1995; Dienes and Oravecz, 2000; Giesbrecht and Evert, 2009), to build a universal tagset (Petrov, Das and McDonald, 2012; Zhang, Reichart and Barzilay, 2012), to evaluate a proposed tagset (Sawalha and Atwell, 2013), to easily use other corpora (Atwell, Hughes and Souter, 1994), to standardise languages resources (Leech and Wilson, 1996; Schmidt *et al.*, 2006), and to merge an existing annotated corpus into a new one (Habash and Roth, 2009).

Mapping a tag to a different tagset can be seen in one of the following situations (Teufel, 1995):

- **1-to-1 mapping:** This is just renaming of the tag.
- **n-to-1 mapping:** Many tags can only be mapped to one tag in target tagset. We lose some information from the source tagset. For example, mapping perfect, imperfect and imperative verbs to V will entail losing the aspect of the verb.
- **1-to-n mapping:** The tag is ambiguous. However, the source tag will have less information than possible target tags.
- **m-to-n mapping:** This case is the most challenging one.

Most of the proposed mappings in the literature involve a reduction in the tagset size; i.e. mostly mappings are many-to-one or one-to-one mappings. We will not go into detail on those mappings, as those mappings are just “renaming” of the tagset. However, several attempts have been made to “standardize” tagsets (map tagsets to a standard one) or “cross" map existing tagsets. In the following sections, we will explore and describe each approach.

3.4.3 Cross Mapping of Tagsets

Automatic Mapping Among Lexico-Grammatical Annotation Models (AMALGAM) project (Atwell *et al.*, 2000) was a pioneer project that tried to provide a full-featured mapping. AMALGAM aimed to provide a “POS-tagset conversion” method for English annotation schemas; i.e. given a text tagged with one tagset, it outputs the text tagged with another tagset, no matter how the two tagsets differ in their formalism, size, etc.

The AMALGAM project maps the tagset A to tagset B by doing the following steps: First, it builds a POS-tagger trained on the corpus tagged with tagset B. Next, it uses the tagger to predict the tag of the word in a corpus tagged with tagset A. In other words, there are no mapping rules from tagset A to tagset B. This decision was made as the authors discovered in earlier experiments that the n-to-m and 1-to-n mapping “predominated” over the simple 1-to-1 and n-to-1 mappings.

Teufel (1995) proposed a mapping tool which maps morphosyntactic tags to a specification language. This language is typed, constraint-based, and editable. The paper did not handle multiple tags per word, i.e. words that have clitics, each with a tag, which is a very common pattern in Arabic.

Pîrvan and Tufi (2006) proposed a cross-tagging generic algorithm that allows mapping one tagset to another. The algorithm uses four components: two gold standard corpora. Each one is tagged using a tagger learned from the other corpus. The four components then get involved in a stochastic process that builds what they called *cross-mapping* by finding probabilities of the contingency table of tokens. The paper claims that it is possible to merge the two corpora confidently tagged with either of the tagsets. It claims even that gold standards can be improved. However, the algorithm seems to assume that the two corpora are aligned when constructing the contingency table. In the paper, the authors mentioned a

tokenisation inconsistency within a gold standard, but they did not mention how the corpus and its cross-tagged version were aligned.

In Arabic, most of the mappings of tagsets consider a reduction or renaming of some tagset, to match the target application such as POS tagging (Toutanova *et al.*, 2003; Diab, 2007; Habash, Rambow and Roth, 2009; Pasha *et al.*, 2014), parsing (Kulick, Gabbard and Marcus, 2006), universal representation (e.g. UD treebanks), or faster treebank production (Habash and Roth, 2009). The link between the reduced tag and the set of fine-grained tags is usually maintained. There are some cases where reduction is followed by expanding some tags using some manual correction, e.g. the QAC tagset (Dukes and Habash, 2010). Similarly, the SALMA sample of the Quran was developed using a mapping procedure from the QAC corpus (Sawalha and Atwell, 2013). To the best of our knowledge, there is not any existing work that handles the mapping of two independent tagsets without manual intervention at the word level.

3.4.4 Standardizing Tagsets

One appealing solution to the diversity of tagsets is the standardisation of annotation schemes. The most famous example of this approach is the EAGLES[5] initiative, which aimed to build standards for large-scale language resources. One of the outcomes of the initiative is the EAGLES meta scheme (Leech and Wilson, 1996) which provides three levels of constraint (obligatory, recommended and optional) and in each constraint, a set of attributes and their possible values are defined. For example, it is 'recommended' to have an attribute *number* for *noun* tagging, and its value can be *singular* or *plural.* These guidelines urge that tagsets should be mappable to the provided framework, i.e. the tag should be mappable to one or more attribute/value pairs.

In collaboration with EAGLES, the Multext-East Project built a similar project for central and eastern European languages (Dimitrova *et al.*, 1998). The project built parallel and comparable corpora POS tagged and aligned to the English version of the original text.

While these two frameworks provide a detailed set of standard morphosyntactic terminology, they are only applicable to Indo-European languages.

[5] The Expert Advisory Group on Language Engineering Standards.

For example, Arabic nouns can be dual but in EAGLES they are either singular or plural. Additionally, EAGLES aims to increase tagging comparability of taggers, but a tagger must map its tagset to EAGLES's course tagset. This mapping would reduce the quality of such comparison. EAGLES does not constitute an interlingua tagset for translating between existing tagsets, as it is not resolving the problem of tokenisation (Hughes, Souter and Atwell, 1995). Applying EAGLES standards to Arabic in a tagset will "seem alien to Arabic linguists and grammarians" (Atwell, 2008, p. 517).

Similarly, in a joint project between three research centres in Germany, (Schmidt *et al.*, 2006) presented a new initiative to "standardize" existing linguistic resources. They addressed the diversity in the language resources and proposed a solution to the integration of linguistic terminology. In contrast to the EAGLES project, they propose a "terminological backbone" that is well-defined, does not integrate language-specific tags, is not limited to European languages, and is a more thorough terminological resource.

Additionally, Petrov, Das and McDonald (2012) developed a mapping of twenty-five different treebanks tagsets from twenty-five languages (including Arabic) to a single universal tagset, initially for unsupervised part of speech tagging. The tagset is a course annotation scheme that has twelve tags: ADJ, ADP, ADV, CONJ, DET, NOUN, NUM, PRON, PRT, VERB, X denoting others, and DOT (.), denoting punctuation marks. These tags were chosen to be the most useful tags to exist among different languages. The mapping was done manually by a high level analysis of tagset. This mapping does not solve one-to-many mapping cases; they map to a courser tagset, whereas the majority of treebanks are "very fine-grained".

However, this project has since been adopted as a widely used standard for mapping diverse tagsets to a common standard. It has been used later as a standard tagset in the famous Universal Dependencies Project along with other projects like the Interset (Zeman, 2008), a tool for converting morphosyntactic tagsets of different languages. We used this universal tagset as a course tagset version of taggers' fine-grained tagsets on several occasions.

3.4.5 Segmentation Schemas

Different schemes in POS tagging assume a different tokenisation of the input text. This tokenisation varies from tagging compound names with one tag to

tagging each affix of a word. In the same manner, some taggers do not tag some of the text, punctuation, dates, numbers, etc.

In the Arabic language, Habash and Sadat (2006) defined three ordered degrees of segmentation structure: [CONJ+ [PART+ [Al+ BASE +PRON]]]. The degrees are ordered, which means that CONJ+ cannot appear after a PART+. The authors constructed several schemes, amongst which are: **D1** which separates CONJ+ from the BASE, **D2** which separates CONJ+ and PART+ from the BASE, and **D3** which separates CONJ+, PART+ and Al+ from the BASE, respectively. The ST scheme is the baseline, where a word is tokenised by splitting off punctuations, numbers and diacritics. More schemes can be defined, as Arabic is highly inflectional. As a consequence, taggers can have varied tokenisation schemes.

Taggers differ, however, in some more details. Some taggers segment not only clitics but affixes as well. For example, the first character on imperative verbs is segmented and tagged with some tags that indicate the person and number of that verb: *>u/IV1S + bAliy/VERB_IMPERFECT.* Some taggers segment proclitic pronouns that indicate the subject and some do not. Traditionally, Arabic grammar segments these pronouns in the "Ia'rab" system: *'aAma/V + n~aA/PRON.* More details are presented in 4.5.

3.4.6 Segmentation Alignment

Because of different schemes of segmentation, it is necessary to align the results of those taggers for proper evaluation and voting. However, this alignment is "quite sophisticated" (Atwell *et al.*, 2000). Segmentation alignment, in general, is a requirement for evaluating different taggers that assume different tokenisation/segmentation. Combination techniques vote between aligned tokens as well.

Morphological alignment can be defined as a sequence alignment problem. Any algorithm that tries to solve the problem will compute a score of similarity (a.k.a. distance) between the two sequences and tries to minimise that distance. The output of that algorithm is an alignment, a series of operations (e.g. addition, deletion, substitution) where each operation has a cost with the goal of transforming one sequence into the other. The optimal alignment is the alignment that has minimum cost.

In the GRACE evaluation task (Adda *et al.*, 1998), the organisers used a reference corpus that uses text tokenisation different than the one returned by a participant. They give the participants complete freedom in the tokenisation scheme. The total number of tokens returned from participants after processing the test dataset varied from 416,193 to 463,596 tokens. The "realignment" was done using a token-level comparison using each token's lexical form. Specifically, they used the UNIX command *diff* (after putting each token in a new line) which finds the difference between two files. The number of reference tokens is always larger than or equal to the number of tokens of any participant; i.e. reference tokenisation is the most fine-grained one. The result was then realigned (substring matching in two runs), by first adding "ghost" characters then rebuilding the original tokens. In case a token could not be realigned, it is omitted from the evaluation (Adda *et al.*, 1998). This alignment assumes that there are limited changes to the word when it has been tokenized; however, it is very common in Arabic to have orthographical changes when tokenizing the word: such as words with final *Taa Marbutah /p/* which is converted to *Taa /t/* (N. Y. Habash, 2010, p. 60).

In contrast, the AMALGAM project used a neutral tokenisation scheme. In order to simplify comparisons, they used just one tokeniser for all schemes. This produced errors in tokenizing and tagging negatives (*aren't*), enclitics (*where's*), and expressions like (*for example*, *have to*, *set up*…) (Atwell *et al.*, 2000). This is only possible because they built a Brill tagger for each participating corpus. It is, however, not possible if the POS-tagger has an integrated tokeniser with no option to configure it.

To the best of the author's knowledge, there is no work that described the problem of morphological alignment in Arabic or suggested a systematic mapping from one to another. However, several studies presented the biword alignment of bilingual parallel corpus (Lee, 2004; Elming and Habash, 2007; Nguyen and Vogel, 2008). Those studies inspired us to develop and learn the alignment from a multi-tagged corpus.

3.4.7 Word Form Similarity

The distance between two text strings can be measured using one of the string metrics (or string distance measurements). One of the most commonly used metrics is the edit distance (also called Levenshtein Distance) (Levenshtein, 1966)

which counts the number of deletions, additions, and substitution operations required to transform one string to the another. A smaller number indicates greater similarity between strings. We are using string metrics for the alignment purposes on the level of word and morpheme between different taggers and/or reference corpora.

Damerau (1964) extended Levenshtein distance algorithm to include the transposition operation. Many other string metrics exist including Longest Common substring, Jaro and Jaro-Winkler (Cohen, Ravikumar and Fienberg, 2003; Gomaa and Fahmy, 2013). However, those string matchings are character-based algorithms and treat all letters with an equal weight.

Arabic words can be optionally diacritised, and therefore a simple string metric is not a perfect metric for comparing two Arabic words. Freeman *et al.* (2006) extended the Levenshtein algorithm for the purpose of matching Arabic Romanised names by mapping possible English equivalence class to Arabic letters. Abdel Ghafour *et al.* (2011) proposed a string matching algorithm for the purpose of name matching by defining different levels of similarity scores. In each level, groups of letters are categorised based on their phonetic similarity. Similarly, it defines groups based on letter form similarity. Finally, a function is defined to report the keyboard distance between two characters. The algorithm then computes the similarity based on the three criteria.

However, none of the algorithms appears to solve the problem of comparing two partially voweled strings. Diacritisation systems (even human-based ones) use a variety of conventions for diacritizing certain letters, e.g. the final *Alef* letter with nunation. While some put the nunation vowels (ً, *Fatha nunation*) on the Alef letter, some put it in the previous letter. Similarly, a letter preceding an Alef letter is always vocalised with a (َ, *Fatha*) vowel, and linguists debate whether it should be written or not.

3.5 Automatic Annotation

3.5.1 Taggers

Several previous studies surveyed the linguistic resources available for researchers in the field of Arabic NLP. In these surveys, the aim is to come up with a list of morphological analysers that one can use for downstream applications. Atwell et al. (2004) conducted a survey on the Arabic MAs and came up with a list

of 10 different analysers. They concluded their survey pointing out that most of those analysers are not freely available or they are hard to use. Maegaard (2004) surveyed the state-of-the-art language resources including MAs and POS taggers. The Basic Language Resource Kit (BLARK) project in 2010 listed 7 MAs, three of which are commercial software. Sawalha (2011) listed 6 MAs with his proposal of a new fine-grained morphological analyser, three of which are freely available. A noticeable inconsistency can be seen in the literature, maybe due to the lack of a regularly updated directory of NLP tools, or due to the fact that some tools become unavailable. In a survey of the literature on POS tagging techniques, Albared (2009) surveyed the "POS tagging" techniques with a focus on Arabic: MSA and dialects. None were explicitly designed for classical Arabic. Those techniques were criticised as assuming closed-vocabulary and low generalisation with OOV words which is the major challenge with domain adaptation to classical Arabic.

However, because our goal is to combine different taggers, these taggers should be freely available to be included in our ensemble. Chapter 4 surveys Arabic POS taggers and morphological analysers that matches four constraints: availability, generality, credibility and normality (i.e. designed for standard Arabic). It evaluates surveyed taggers on a classical Arabic dataset. We refer the reader to this chapter for more details.

3.5.2 Domain Adaptation

Supervised tagging and segmentation methods usually score the best accuracies. However, these methods are usually hard to port to other languages or variants since they requires a training dataset that is usually costly and expensive. There exist several other methods in the literature to tackle this problem, which includes bootstrapping training datasets which assumes partially tagged training corpus, and unsupervised methods (Freeman, 2001; Clark, 2007; Albared, Omar and Ab Aziz, 2009).

Another approach is exploit existing tagged corpora and adapt these corpora to different languages. One example is lemmatization using translation from a second language. Another example is the reproduction of morphological analysis through exploiting existing taggers designed for a related language, which is discussed in this thesis.

Experiments that used these MSA-based taggers for classical Arabic reported a significant drop in the accuracy. Even though the morphology of classical Arabic is the father of MSA, some studies showed that CA texts are not compatible with MSA taggers.

Alrabiah (2014) compared two MSA-based taggers both designed for MSA to annotate the KSUCCA classical Arabic corpus. Using five samples from different genres of classical Arabic, an evaluation of these two systems showed a drop in their accuracy by 10-15%.

In addition, the semi-annotation of the **QAC corpus** (Dukes, Atwell and Habash, 2013) used an MSA morphological analyser (Buckwalter analyser), but the manual verification step made corrections to at least 24% of words, nearly a quarter of text words, although the text is fully diacritised.

The Heritage Corpus (Mohamed, 2018) used a tagger trained on the PATB treebank to tag one part of his corpus (2000 words). The accuracy achieved is only 78.62% and referred to the high percentage of out of vocabulary words (43.39%).

This finding is not limited to POS tagging. For example, tag-based text compression using Prediction-By-Partial Matching exploits the morphological analysis to improve the compression performance. The method tagged the text to be compressed, and the tagged files then compressed and compared against baseline character-based compression. In (Alkhazi and Teahan, 2017), the tag-based compression (using MSA tagger) shows improvements in MSA texts over the character-based compression. However, for classical Arabic texts it does not show a similar pattern, which suggests that “the quality of tagging of classical Arabic has dropped”.

3.5.3 Combining Taggers

In Machine Learning, ensemble methods refer to the process of combining multiple learning methods to obtain a higher accuracy in classification prediction that was not achieved by any of individual learning methods. Multiple types of ensemble exist in the literature, including bagging (equally-weighted models trained on random subsets of the training data) and boosting (adaptive training where each new model focus on a subset of training data that were misclassified). Many other combination techniques are available.

In POS tagging, different techniques were used, including: knowledge-based models– table lookup, syllable-based morphology, pattern morphology; and empirical methods– Hidden Markov Models (HMM), Support Vector Machines (SVM). Each POS tagger is designed differently; however, without a full understanding of the language, no POS tagger could ensure perfect accuracy. Because of their different knowledge bases and diverse inference methods, taggers will typically produce different errors (Halteren, Zavrel and Daelemans, 2001).

The combination of *heterogeneous* POS taggers exploits these differences, and it is reported to achieve a better accuracy for several languages, including English (Marquez *et al.*, 1999; Halteren, Zavrel and Daelemans, 2001; Schroder, 2002), Italian (Søgaard, 2009), Icelandic (Henrich, Reuter and Loftsson, 2009), Polish (Śniatowski and Piasecki, 2012; Kobyliński, 2014), and Swedish (Sjöbergh, 2003) and even for Arabic (Zribi, Torjmen and Ahmed, 2007; Aliwy, 2013; Albared and Hazaa, 2015; Zeroual and Lakhouaja, 2017).

Most of the combination of POS-taggers are based on training different models inferred from *one* training corpus, i.e. homogeneously annotated texts. Therefore, each model uses the same or reduced tagset and morphological segmentation as the one on the training corpus. However, heterogeneous combination of black-box taggers or heterogeneous corpora involves handling different issues (see Section 5.2).

3.5.4 Exploiting Heterogenous Resources

For word segmentation in Arabic and POS labelling, supervised learning has become a dominant model. Its progress is due to the development of annotated corpora and NLP techniques. Although many corpora are released in the literature, obtaining sufficient amounts of high-quality training data remains a major obstacle, especially for morphologically rich languages. Annotation schemes for corpora are adversarial since underlying linguistic theories differ. Sadly, although there are multiple resources, it is not possible to merely collate such data for training systems, since almost all existing NLP systems assume a homogeneous annotation. Therefore, it is essential to consider how to use and exploit heterogeneous resources to improve Arabic word annotation and segmentation.

A second related problem is that existing corpora are usually drawn from some specific domains, e.g. newswire data. Adapting these corpora to a new

domain, e.g. classical Arabic, usually is not trivial. The second problem is well-studied, e.g. adapting MSA to dialectal Arabic (Monroe, Green and Manning, 2014; Albogamy and Ramsay, 2016). However, we agree with Jiang *et al.* (2009) and argue that the two approaches are related but the underlying problem is different. Domain adaptation assumes that the annotation guidelines are the same in terms of tagging and segmentation and only the *distribution* is different (Jiang, Huang and Liu, 2009). Contrarily, annotation-style adaption, as defined by Jiang *et al.*, assumes the guidelines themselves are different and tries to exploit the shared knowledge, and the distribution might be the same or different.

3.5.4.1 Annotation-style Adaptation: combining heterogeneous corpora

Exploiting heterogenous resources in annotation-style can be done by developing a tagger from heterogeneous corpora or by combining heterogeneous black-box taggers. They both exploit the annotation which is costly in terms of time and money. However, there are some differences. First, the evaluation of black-box taggers is not always possible because of the lack of evaluation datasets. Second, these taggers are not always tuneable, as they come pretrained on a specific dataset with specific guidelines.

There are a growing number of efforts that address the reusability of heterogeneous corpora, especially in Chinese[6]. Most of these are done toward integrating different corpora instead of adversarial taggers. There are two main approaches: stacking and multi-view learning. Stacking methods, e.g. (Jiang, Huang and Liu, 2009), pile up independently trained models where each model is trained based on the predicted values of the previous model. These methods suffer from an error propagation problem. Recent works shift to Multi-view models, which, in contrast, model the problem jointly by sharing common feature representations and treat the problem as a multi-class problem.

Qiu *et al.* (2013) uses a multi-view model and trains two homogeneous corpora simultaneously, using a manually-extracted loose mapping between the two

[6] The Chinese language shares some features with Arabic, namely the need to segment the text sequences to reduce the word form sparsity. Chinese words comprise several characters, up to four or five characters (Teahan *et al.*, 2000). In Arabic, a surface word form comprises several morphemes, up to four morphemes, and five-morpheme words exist but are rare. The segmentation is not standard in both languages, which results in corpora annotated with adversarial segmentation schemas.

corpora tagsets. Chen *et al.* (2017) recently published an ACL award-winning paper that proposes an adversarial method to exploit heterogeneous segmentations in Chinese corpora using deep neural models. The integration of shared knowledge from different segmentation schema is done by regarding the problem as a multi-task learning problem. A shared layer is used to extract shared features (using a custom adversarial objective function), and a private layer is used to extract segmentation-specific features. The two methods are shown to be effective in improving the performance of Chinese word segmentation.

3.5.4.2 Annotation-style Adaptation: Reusing Adversarial Taggers

Building an ensemble tagger from heterogenous corpora might be more appealing since it does not restrict the ensemble to the taggers' constraints, e.g. how they expect the input. However, taggers might employ sophisticated techniques like the use of external resources (e.g. lexicons), which the state-of-the-art Arabic taggers do. It is worth exploiting Arabic heterogenous corpora, especially with the growing number of classical annotated corpora, but we have left it for future work.

Most works in Arabic ensemble segmentation and tagging used homogeneous settings (Zribi, Torjmen and Ahmed, 2007; Aliwy, 2013; Albared and Hazaa, 2015; Zeroual and Lakhouaja, 2017). To the best of the author's knowledge, four works in Arabic utilize adversarial and homogeneous tagging and domain adaptation from MSA to classical texts. The following detailed review will only discuss studies that combined heterogeneous ensembles in Arabic language analysis.

1. **Alabbas and Ramsay (2014)**

Alabbas and Ramsay (2012a, 2014) performed a simple method for combining three Arabic taggers: MADA, AMIRA and a simple home-made maximum likelihood tagger (MXL). They examined five strategies of combining the results: three strategies of majority voting (with backoff to MADA, AMIRA or MXL), majority voting with backoff to the most confident, and most confident as the primary strategy. To define the most confident tagger, they examined how likely a tagger is correct when tagging with one particular POS tag (e.g. noun), e.g. MADA is 95% correct when it is tagging as a noun. The most-confident strategy achieved the highest accuracy with 0.995 with a coarse-grained tagset and 0.96 with a fine-grained tagset.

To recover from the mismatches between the reference corpus used (PATB Part 1 v. 3) and AMIRA tagset, the authors used transformation-based retagging

(TBR) which improves the score from 90% to 95%. However, AMIRA and MADA tokenise sometimes differently. To solve this problem, the PATB was translated to a coarser version of AMIRA's tagset, and compared with AMIRA's output: the output is used if it is compatible with the translated tag; otherwise, the translated tag is used. This edit ensured that AMIRA and MADA will use the same token number as the PATB. The accuracy of this study is encouraging. The combination of taggers boosts the accuracy by 2-4%.

Although this work is encouraging to our research, testing on a subset of the ATB is problematic as the individual tools are trained on the ATB– "due to its limited lexical diversity and the similarity between the training and test sets" (Darwish and Mubarak, 2016, p. 1070). This generally leads to results that are often artificially high.

The technique does not propose a systematic method for homogeneous segmentation schemas. In fact, the AMIRA toolkit uses a reduced tagset from the PATB tagset, which means that there is a direct link between the two outputs[7]. The handling of segmentation differences makes this technique inapplicable to unseen text as it relies on a pre-processing step on the tagged corpus (to enforce same tokenisation).

2. Albogamy and Ramsay (2016)

This work does not introduce an ensemble tagger; however, it uses heterogenous taggers for the purpose of improving POS taggers using agreement-based bootstrapping on noisy microblogging texts (Twitter). Using three off-the-shelf Arabic taggers (Stanford, MADA, AMIRA) for such text reports leads to a drop in performance; their accuracies range from 49-65%. The best approach to improve their performance was to retrain on a small twitter dataset, pre- and post-process texts, add MSA annotated corpora and use agreement-based bootstrapping.

The novel agreement-based bootstrapping aims to increase the training dataset size by adding words that the three taggers agree upon. However, since the three taggers use different tagsets, their tagsets were reduced to a collapsed tagset, which is used to evaluate the predicted outputs of the three taggers instead of the original ones.

---

[7] POS tags are from "the collapsed set of tags included in the Arabic treebank distribution (known as the Extended Reduced Tag Set or ERTS)" [README file in AMIRA package].

This work is related to our research as it utilizes heterogenous taggers and adapts them to a new domain, microblogging. However, this work lacks more details about the segmentation alignment between the three taggers. Its evaluation is word-based but there is no clear mention of handling heterogeneous clitics. In addition, the agreement-based addition of words will introduce incomplete or ill-formed sentences, which might affect the final tagging performance. Constraining to only sentences with full agreement is not practical; as the agreement between tagger is low (60.4%) according to the authors. The chance of a 5-word sentence to have a full agreement is very low: 7%.

3.5.5 Classical Arabic Tagging

1. **Rabiee (2011): Adapting from QAC to MSA**

Rabiee (2011) tried to adopt several taggers by training them on the QAC and then applying the learned model on tagging an MSA sample. He used BAMA as a morphological analyser and used TreeTagger to train a model from the QAC. Tagging then was constrained by the solutions of BAMA. The tagset of BAMA was reduced to only a 9-tag tagset that was comparable with the QAC tagset. However, his mapping encountered one-to-many cases (e.g. mapping ADV tag). In that case, he chose to map to the most common tag. The accuracy achieved in tagging a 66-word MSA sample was 76%.

This tagging can be seen as a novel sequential tagging scheme as it uses the output of BAMA to constrain TreeTagger. The coarse mapping of the tagset is justified as the author needs to compare taggers with different tagsets. However, errors were raised from this mapping: *LOC* is about 38% of *ADV* cases, and the mapping of *ADV* to the other more common tag *T* constrains the TreeTagger to an incorrect tag. Additionally, the author used a test sample with only 66 words, which does not count as a representative sample of the MSA. The sample's origin, genre, and how it was annotated were not even clear. The author used an earlier version of QAC which has word-based annotation, and thus the morphological alignment was not an issue.

2. **Alrabiah (2014): Adapting from MSA to Classical Arabic**

This work compared two MSA-based taggers both designed for MSA – Alkhalil and MADA - to annotate the KSUCCA corpus. Ten samples were randomly extracted from KSUCCA from different genres, and each sample is of 100

words. Seven annotations are used to evaluate each tagger: root, pattern, lemma, stem, POS tag, number and gender, only two of which is common between the two taggers. Because Alkhalil does not disambiguate between proposed solutions, Alrabiah proofread all of them and whenever one of them was correct it was marked as a success. The evaluation of these two systems showed a drop in their reported accuracy by 10%-15%.

A comprehensive experiment in Chapter 4 tends to confirm similar findings. These studies show that current taggers might need to be adapted for classical Arabic and their dictionaries need to include classical lexicon.

Table 3.3 The accuracy of MADA and Alkhlalil (Alrabiah, 2014)

Tool	Stemming	POS tagging
AlKhalil v.1	75.1%	77.6%
MADA v3.2	84.9%	83.40%

3. Alashqar (2012): Various Taggers on Quranic Arabic

Alashqar conducted a comparison between POS techniques using the Quranic Arabic Corpus. He compared four techniques: N-Gram, Brill, HMM, and TnT taggers. The experiments were done on two versions: diacritised and undiacritised Quranic text using NLTK toolkit.

After pre-processing a diacritised QAC file to match NLTK format, an undiacritised file was generated. Next, the author mapped the tags into the 9-tag simplified tagset, resulting in four cases of the experiment: diacritised vs. undiacritised and 9-tag vs. 33-tag tagsets. He trained several models using 97% of the corpus and reported the accuracy of each model in POS tagging the remaining 3% of the text.

According to the authors, N-Gram (particularity Bi-Gram) taggers outperform other taggers. The best model accuracy is 83.2%, a Brill tagger on the undiacritised version. Tagging undiacritised text also outperforms diacritised ones, especially in the case of Brill Tagger. The mapping to 9-tag tagset increased the accuracy for all experimented taggers, with the exception of TnT tagger.

This experiment shows that off-the-shelf taggers are not always capable of handling Arabic specific problems: diacritisation and high inflection. Diacritisation causes the word sparsity to be high, and for the Brill tagger, for example, accuracy on the diacritised version is as low as 38.6%. In addition, the segmentation problem

is not discussed in the paper. The author formalized the problem as a word-based tagging POS tag, i.e. no prior segmentation is required.

3.6 Conclusion

In this chapter, we first provided a survey of available corpora with a focus on morphologically-annotated corpora. We show that there are several open access manually annotated corpora of classical Arabic, but most of them are on Quranic texts and none for Hadith. There is a need for manually-annotated corpora of general classical Arabic as well. We showed the need for an open access annotation tool that is designed for Arabic specifications.

The survey of mapping tagsets reveals that most approaches are reductive. Although there are several initiatives to standardise PoS tagsets for Indo-European languages, Arabic tagsets are still highly incompatible.

We surveyed the literature for methods that exploits heterogeneous corpora and taggers. A few works are done in Arabic, although almost every corpus is tagged using its own tagset. In the next chapter, we survey the open access Arabic taggers and evaluate them on tagging classical Arabic excerpts.

4 MORPHOSYNTACTIC TAGGING OF CLASSICAL ARABIC

Chapter Summary[8]:

Focusing on classical Arabic, this chapter in its first part surveys morphological analysers and POS taggers that are open access, are designed for Modern Standard Arabic (MSA) or classical Arabic (CA), can analyse all forms of words, and from a credible academic research group. This chapter lists and compares the supported features of each tool, and how they differ in the format of the output, segmentation, Part-of-Speech (POS) tags and morphological features. A sample output of each analyser is used to demonstrate the differences using one CA fully-vowelised sentence. This part serves as a guide in choosing the best tool that suits research needs.

The second part reports the accuracy and coverage of tagging a set of classical Arabic vocabulary extracted from classical texts. The results show a drop in the accuracy and coverage and suggest an ensemble method might increase accuracy and coverage for classical Arabic.

[8] Some parts of this chapter are based on:

Alosaimy, A. and Atwell, E. (2017) 'Tagging Classical Arabic Text using Available Morphological Analysers and Part of Speech Taggers', *Journal for Language Technology and Computational Linguistics*. German Society for Computational Linguistics & Language Technology (GSCL), 32(1), pp. 1–26.

4.1 Introduction

Arabic morphological analysis is essential to Arabic NLP tasks, and part-of-speech (POS) tagging is usually done as one of the first steps of advanced NLP tasks such as statistical machine translation and text categorisation. It derives its importance as its accuracy impacts other subsequent tasks. Arabic morphology is one of the most studied topics in Arabic NLP. Due to the nature of the language, being highly inflectional, and the lack of short vowels, morphological analysis of Arabic is not an easy task. The analysis involves handling a high degree of ambiguity.

POS tagging usually uses the information provided by the morphological analyser. A morphological analyser (MA) is a context-independent tagger that provides all possible solutions based on a lexicon or dictionary. While POS taggers and MAs label the word morphosyntactically, some POS taggers use the context to either choose the most probable tag according to the context or at least provide an ordered list of tags.

Surveys of the literature show that multiple morphological analysers and POS taggers exist. The accuracy and features of those taggers vary, and errors are generated for every tagger. No tagger shows a perfect performance, and no tagger has been adopted as a standard. Therefore, choosing between available taggers can be challenging.

Even though the morphology of MSA is inherited from CA, two studies showed that classical Arabic is not compatible with MSA taggers and vice versa. Rabiee (2011) tried to adopt several taggers by training them on a classical Arabic Corpus: the Quranic Arabic Corpus (QAC), and then tested them on MSA. The accuracy achieved in tagging a 66-word MSA sample was “not impressive”–73% was achieved. Alrabiah *et al.* (2014) compared MADA and AlKhalil (both designed for MSA) in order to annotate the KSUCCA corpus. Using five samples from different genres of classical Arabic, an evaluation of these two systems showed a drop in their accuracy by 10-15%. It shows that current taggers need to be adapted for classical Arabic and their dictionaries need to include more classical vocabulary. This evaluation is extended to examine the coverage and accuracy of the surveyed tools.

The next section discusses the survey design and criteria. The third and fourth sections list surveyed POS taggers and MAs in detail. The fifth section

compares those tools by their features and demonstrates such differences on one tagged sentence. The last section reports the accuracy and coverage on a collection of classical vocabulary.

4.2 Survey Methodology and Criteria

Focusing on *open access* MAs and POS taggers, we performed a comprehensive search, which adds to previous surveys, an in-depth literature review of available MAs and POS taggers. We limited the search to MAs and POS taggers that:

- **are designed for MSA or CA,** i.e. either designed for Arabic but not intended for dialectal Arabic or has a model for MSA or CA;
- **are able to analyse all forms of words,** i.e. not designed for verb only for example;
- **are open access,** i.e. available freely for research purposes; and
- **have a credible academic establishment,** i.e. either has at least one published academic paper or published by a well-known research group.

The result of this survey includes seven MAs and eight POS taggers listed in Table 4.1.

For the sake of completeness, Table 4.2 lists some tools that do not conform with the availability condition. However, as other researchers have used them, they might have been available, and someone may get hold of them in future. However, we contacted their owners to receive a copy for research purposes, but we did not get any response.

4.3 Survey of Open Access Morphological Analysers

Seven morphological analysers (MA) match our criteria. MA differs than POS tagger in that they do not perform any disambiguation; therefore, they provide a list of analyses with no order. MA typically do not consider the context in the analysis.

One common phenomenon is the lack of proper documentation that does not only include installation guides but a technical documentation of tagset and tools features. When the tagset is listed, it is sometimes not comprehensible, as tags are always shortened for representation purposes.

Table 4.1 The list of MAs and POS Taggers that have been studied

Name	Code	Main Category	Sub-category	Tagset Size[9]
Mada	MD	POS-tagger	knowledge-based: BAMA. SVM using SVMTools for disambiguation	36
AMIRA	AM	POS-tagger	data-driven: SVM using YAMCHA	25
MadaAmira	MX	POS-tagger	knowledge-based: BAMA. SVM for disambiguation	36
Stanford	ST	POS-tagger	Data-driven:	25
ATKS' POS Tagger	MP	POS-tagger	Data-driven: SVM with CCA features	N/A
MarMoT	MR	POS-tagger	Data-driven: CRF	25
SAPA	WP	POS-tagger	Data-driven: CRF	24
Farasa	FA	POS-tagger	Data-driven: Joint prediction with syntax	16
AraComLex	AR	MA	FST	14
ElixirFM	EX	MA	Haskell, functional programming	23
BAMA (AraMorph)	BP	MA	Dictionary	70
Almorgeana	AL	MA	Dictionary	36
ATKS' Sarf	MS	MA	N/A	70
AlKhalil	KH	MA	Dictionary	> 118
Qutuf	QT	MA	Dictionary	N/A

Table 4.2 The list of MAs and POS Taggers that have been excluded.

Name	Main Category	Sub-category	Excluded
MORPH2	MA	knowledge-based: XML lexicon	Yes[10]
Khoja ArabicTagger	POS-tagger	Hybrid: Statistical and Rule-based. Vetrabi for disambiguation	Yes[10]
SAMA	MA	Dictionary	Yes[11]
SALMA	MA	N/A	Yes[10]
Xerox	MA	FST	Yes[12]

[9] Tagset size might be different from published numbers. This is the output of the process of finding *core* tags (or *basic* tags), removing embedded inflectional features and splitting compound tags. *Complex* tags refer to tags with embedded inflectional features. *Compound* tags refer to tags that aggregate all morphemes tags in a single tag. For example, *NNS and VBZ* are a complex tag, while *NN* is a core tag. *DTNN* (an article and a noun) is a compound tag.

[10] Authors did not respond to our request for their system.

[11] Only available to LDC members.

4.3.1 AraMorph (BP)

AraMorph (a.k.a BAMA, stands for Buckwalter Arabic Morphological Analyser) is free GNU-licenced software initially written in Perl by Tim Buckwalter in 2002 and published in www.qamus.org. The software was later optimised by Jon Dehdari in 2005 to support UTF-8 encoding and speed up the processing time. AraMorph has been ported to Java by Pierrick Brihaye and published on www.nongnu.org. AraMorph received further work in 2012 by Hulden and Samih (2012)[13] that converts original table-based procedural AraMorph software into a finite-state transducer (FST) parser using his Foma Software (Hulden, 2009)[14]. The authors claim that it is faster and more flexible, i.e. a more extensive range of applications can use the FST such as spell checkers. Tim Buckwalter released BAMA 2 and later SAMA 3, but they need the Linguistic Data Consortium (LDC) licence to be used; therefore, they have been excluded from our list.

AraMorph views the Arabic word as a concatenation of prefix+stem+suffix, where prefix and suffix can be null. It has a lexicon where each lexeme is assigned a category (in addition to its POS tag and gloss). This categorisation is the most important part in the analyser and it embodied all morphological decisions. For example, some categories allow the addition of Taa Marbouta to mark feminine noun, but some do not. The analysis is straightforward: using the list of possible prefixes, suffixes, and a compatible table, it extracts all possible compatible substrings that match these affixes and returns all matched candidates.

However, infixes are common in Arabic, and thus it fails to identify them correctly (e.g. identify the plurality of a “broken” plural noun). BAMA does not make use of partially diacritics inputs (Hulden and Samih, 2012).

TAGSET: Tags are mixed with morphological features to form complex tags such as *IV_PASS* (imperfective passive verb). The tagset has about 70 basic tags (Habash, 2010).

---

[12] The demo website is working but its web service produces 501 error which makes it impractical to annotate large corpora.

[13] https://code.google.com/p/buckwalter-fst/

[14] Foma is software for constructing finite-state automata and transducers for multiple purposes. https://code.google.com/p/foma/

4.3.2 AlKhalil (KH)

The AlKhalil Morphological Analyser (Boudchiche *et al.*, 2016) is a morphosyntactic analyser of MSA shipped with a broad set of lexicon and rules. It is free open-source software written in Java and in Perl. The latest version 2 was released in 2016[15] which improved the lexicon and added lemma and its pattern to the list of features. The standard way to interact with AlKhalil is using its graphical user interface that accepts raw text in UTF8 encoding. El-Haj and Koulali (2013) reported that AlKhalil (v1.1) reached an accuracy of 96%.

OUTPUT: The system results can be either shown in the browser or saved as a comma-separated file. Given one word, AlKhalil returns a list of solutions of possible tags of the stem with features. Noun features are its nature, root and pattern in addition to functional features of a noun: gender and number. Verb features are aspect, form and voice in addition to syntactic features: form, root, permittivity[16], transitivity and conjugation's gender, person and number. For every solution, the system determines its voweled form, and its prefix and suffix whenever those exist.

TAGSET: AlKhalil is not consistent in identifying the possible tags of the word, and its results are not in readily reusable form: morphological and grammatical features are embedded within a plain text that describes the analysis. To the best of our knowledge, AlKhalil does not have a predefined set of tags. For example, for some functional words that have different possible analyses, it returns one analysis with a description such as: “conditional or negative particle”, instead of returning two analyses: “conditional particle” and “negative particle”. The estimate number of the possible tags for the base form of the word is at least 118 basic tags.

4.3.3 AraComLex (AR)

AraComLex (Attia, 2006) is a morphological analyser and generator that uses finite state technology shipped with a contemporary dataset of news articles. It uses the rule-based approach with the stem as the base form in its lexicon. The last version published is 2.1[17]. The analyser uses Foma (Hulden, 2009) to construct a model and then lookup for matches.

---

[15] http://oujda-nlp-team.net/?p=1299&lang=en

[16] Verbs are traditionally classified into two categories: "primitive" which all of its characters are primitive and "derived" where one or more characters have been added to the original primitive verb

[17] sourceforge.net/projects/aracomlex/

A distinguishing feature in AraComLex is the identification of multi-word expressions. However, since AraComLex assumes a tokenised input provided by author's tokeniser which was not working[18], we could not find a suitable tokeniser that makes it able to detect and identify multi-word expressions.

INPUT: With the lack of technical documentation and after some trial-and-error: AraComLex expects undiacritised UTF8-encoded text with each word in a line. The system fails to find proper analysis if diacritics are present.

OUTPUT: The output of AraComLex is a set of solutions for every input word in a custom format as can be seen in Appendix B. No description of the tagset is provided: “past” tag, for example, is not lucid (tense or aspect feature). The tagset size is about 14 basic tags.

4.3.4 ALMORGEANA (AL)

ALMORGEANA (Habash, 2007) is a lexeme-based morphological analyser and generator. It uses Buckwalter's lexicon with a different engine that can additionally generate the proper inflected word given a feature-set. In the analysis task, it differs from AraMorph in the output lexeme-and-feature representation. In addition, it has a back-off step where it looks for compatible substrings of prefix and suffix, and if found, the stem is considered a degenerate lexeme.

ALMORGEANA is used in MADA and MADAMIRA toolkits to generate all possible morphological analysis of a given text. This step follows the preprocessing step of normalisation. ALMORGEANA can be used with either the Buckwalter Arabic Morphological Analyser (BAMA) or the Standard Arabic Morphological Analyser (SAMA). The latter is only available to LDC members, so BAMA is used instead. MADA authors reported that using BAMA instead of SAMA will result in a slight drop (2-4%) in word disambiguation.

[18] The author also published a set of relevant tools in his web page including a guesser and a tokeniser in a compiled format for Mac and Windows. However, they did not work on current operating systems (at least on MAC OSX 10.10). One tool is Arabic Morphological Guesser, with the back-off feature; that is, if a word is not found in the lexicon, it guesses a correct morphology rather than returning none.

4.3.5 Elixir Functional Morphology (EX)

Elixir Functional Morphology (Smrz, 2007) is an analyser and generator tool that reuses and extends the functional morphology library for Haskell. Elixir has two interfaces to the core Haskell system written in Perl and Python. Its lexicon is designed to be abstracted from the actual program which allows an easy addition to the lexicon. It was initially derived from the form-based Buckwalter dictionary, but it has been enriched with syntactic annotations from Prague Arabic Dependency Treebank (PADT) and adapted to support function-based morphology.

TAGSET: Elixir uses the same tagset as PADT (23 basic tags). The tags consist of a 10-position string with first two characters reserved for POS tag and the remaining eight includes morphological and grammatical features like gender, person, case and mood.

4.3.6 SARF from Arabic Toolkit Service (MS)

Microsoft Research Lab in Cairo has developed a set of linguistic tools targeting the Arabic language. Among eight tools, they provide free of charge access to a morphological analyser (SARF) and a POS tagger for academic researchers, professors and students only. We could not find an academic paper that describes the morphological analyser methodology. The toolkit can be accessed using the SOAP web service.

The morphological analyser (SARF) provides all possible analyses of an input word: affixes, stem, diacritised form and morphological features such as gender. One distinguishing feature of SARF is that it ranks its solutions based on the actual language usage of each analysis.

TAGSET: The tagset contains 109 possible complex tags, making it the second largest tagset. The tagset has some combination of morphological features in it. For example, pronouns can be suffixed with *_MOTAKALLEM* to denote a first-person. The tagset has about 70 basic tags.

4.3.7 Qutuf (QT)

Altabba (2010) proposed an NLP framework written in Python that has a morphological analysis component. The latest version of Qutuf is 1.01. Qutuf used the Alkhalil dictionary after enriching it. Qutuf extends Alkhalil by making the output more easily reusable and by assigning each solution with a probability.

TAGSET: A tag has 10 slots separated by a comma that represents the base POS tag and some morphological and syntactical features. Some slots serve different meanings depending on the main POS tag. For example, slot 2 represents the punctuation mark (if the main POS is "other"), particle (if "particle") type or gender (if "verb" or "noun").

Table 4.3 The features of each of the morphological analysers for each given word/segment.

Name	AR	EX	BP	AL	MS	KH	XE	QT
Base POS tag	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Aspect	Yes[19]	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Person	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Gender	Yes	Yes	Yes	Yes	Yes	Yes[20]	Yes	Yes
Number	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Transitivity	Yes	-	-	-	-	Yes	-	Yes
Voice	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
State	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Mood	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Case	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Pattern	-	Yes	-	-	Yes	Yes	Yes	-
Root	Yes	Yes	-	-	Yes	Yes	Yes	-
Stem	-	Yes	Yes	Yes	Yes	Yes	-	-
Lemma	-	-	Yes	Yes	-	Yes	-	-
Diacritisation	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Glossing	-	Yes	Yes	Yes	-	-	Yes	-
Tokenisation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Segment-based[21]	-	Yes	-	-	-	-	Yes	Yes

4.4 Survey of Open Access POS taggers

POS taggers assign one POS tag to every word-form or every word's segments. Unlike MAs, POS taggers assign a tag that is contextually suitable. Some POS taggers return only one tag, a ranked list of possible POS tags or a list with each tag assigned with a probability. Some POS taggers use MAs as a preprocessing

[19] Tense (past, present, and future) is used instead of the aspect of the verb, but they are highly related.

[20] Only for nominals.

[21] Whether morphosyntactic features are for each morpheme or not. See Section 6.9 for examples.

step (e.g. MD, MX, MR, etc.) and thus they disambiguate and rank different proposed analyses. Some POS taggers use MAs even in the tokenisation process, e.g. MADA and MADAMIRA.

While some POS taggers do word-based tagging, e.g. (Mohamed *et al.*, 2010), all POS taggers in our list do morpheme-based tagging. Because of Arabic's rich morphology, word sparsity is high, and consequently, word segmentation becomes essential. Studies have shown that word segmentation lowers data sparseness and achieves better performance (Diab, Hacioglu and Jurafsky, 2004; Benajiba and Zitouni, 2010). A POS tagger usually has a component that does the segmentation or relies on the user to provide segmented input. However, this segmentation increases the ambiguity as a word may be segmented into multiple candidate sets of segments.

4.4.1 MADA+TOKAN suite (MD)

MADA (Habash, Rambow and Roth, 2009) is a popular suite that has multiple tools for Arabic NLP. MADA processes raw Arabic text to provide a list of applications: POS tagging, diacritisation, lemmatisation, stemming and glossing. MADA is written in Perl and uses Support Vector Machines (SVM) trained on Penn Arabic Treebank (PATB) to select a proper analysis from the list provided by Buckwalter Arabic Morphological Analyser (BAMA). MADA uses 19 features, 14 of which are morphological features, to rank the list of possible analyses. The reported accuracy of predicting the correct POS tag is 96.1% (Pasha *et al.*, 2014).

A remarkable feature of MADA is how it models the problem. The prediction is word-based: it predicts its clitics by predicting the value of four features: article, preposition, conjunction and question proclitics. It assumes that no two proclitics of one type can co-occur in one word. Predicting the value of each type will result in the word segmentation. In addition, clitics POS tags are complex and embodies some morphological features. This modelling allows the full analysis to be done in “one fell swoop”. No segmentation is required in advance.

TAGSET: MADA “targets the finest possible POS tagset” (Habash, Rambow and Roth, 2009). It supports the mapping to four different possible tagsets: ALMORGEANA, CATiB, Reduced PATB, or Buckwalter. The default tagset has a size of 36 tags for tagging the base of the word. Five, eighteen, seven, and two tags are dedicated to article, preposition, conjunction and questions *proclitics*

respectively; and twenty-two tags for *enclitics*. The tagset used by MADA is well documented in the manual shipped with the suite.

4.4.2 AMIRA (AM)

AMIRA (Diab, 2009) is a toolkit of three main tools: tokeniser, POS tagger, and base phrase chunker. The POS tagger uses YamChi toolkit, an SVM-based sequence classification toolkit. The toolkit does not depend on in-depth morphology information; instead, it learns from the surface data. AMIRA was trained on PATB. The reported accuracy of predicting the correct POS tag using default tagset is 96% (Diab, 2009).

TAGSET: AMIRA can output the tags in one of three tagsets: RTS, Extended RTS, Extended RTS with the 'person' information. Extended RTS has about 72 complex tags, and those tags encode gender, number and definiteness. After removing features from the tag, the tagset is about 25 basic tags.

4.4.3 MADAMIRA suite (MX)

MADAMIRA (Pasha *et al.*, 2014) is a suite that combines two previously mentioned systems: MADA and AMIRA. MADAMIRA ported the two systems into the Java programming language allowing it to be portable, extensible and even faster. MADAMIRA supports MSA and Egyptian Arabic. One added feature to MADAMIRA is the server mode feature, which allows the user to run MADAMIRA in the background and then send HTTP requests for different tasks. While the accuracy has not improved, the speed of tagging has improved over MADA substantially (16-21 times faster). The reported accuracy of predicting the correct POS tag is 95.9% (Pasha *et al.*, 2014).

TAGSET: The tagset used by MADAMIRA extends the MADA tagset by having some tags for Egyptian Arabic processing.

4.4.4 Stanford POS tagger and Segmenter (ST)

Stanford NLP group released a list of Arabic NLP tools including a POS tagger (Toutanova *et al.*, 2003) and Arabic word segmenter (Diab *et al.*, 2013). The POS tagger is shipped with a model for Arabic trained on the Penn Arabic Treebank (PATB). It uses the Maximum Entropy approach to assign a POS tag to a segmented text (using Stanford Arabic Word Segmenter). The Stanford Arabic Word Segmenter uses the Conditional Random Fields (CRF) classifier to normalise the

text and split off clitics from base words in a similar segmentation schema to one used in the PATB. El-haj (El-haj and Koulali, 2013) reported that the Stanford Tagger reached an accuracy of 96.5%.

TAGSET: This tagset is the (augmented) Bies tagset of 25 basic tags. Authors augmented the tagset by adding DT (determiner) to the beginning of nominal tags.

4.4.5 MarMoT (MR)

MarMoT (Mueller, Schmid and Schütze, 2013) is a generic CRF morphological tagger written in Java. MarMoT provides a pre-trained model that was trained on the PATB provided by SPMRL2013 shared task. MarMoT does backwards-forward computations by incrementally increased order to prune the size of possible morphological analyses. MarMoT is efficient in training high order CRF classifiers even with large tagsets and does some approximation using coarse-to-fine decoding. MarMoT assumes a transliterated and tokenised input according to the PATB transliteration and tokenisation. We used the TOKAN segmentation tool to pre-process the input. The reported accuracy of predicting the correct POS tag is 96.43%.

TAGSET: This tagset is the 25-tag RTS tagset. Additionally, MarMoT provides morphological features identical to AraMorph.

4.4.6 Arabic Toolkit Service POS Tagger (MT)

The Arabic Toolkit Service (ATKS) also have a POS tagger (Kim, Snyder and Sarikaya, 2015) that identifies the part-of-speech of each word in a text. A distinguishing feature in this tagger is the use of the Canonical Correlation Analysis method to find a multi-lingual word representation in the prediction of the POS tag. They do not state the use of their morphological analyser (SARF) in the process of tagging. This tool identifies the grammatical features like mood and case; also, it resolves the nunation, the addition of nun sound that indicates a noun's indefinite case. Instead of normalising, the tagger uses a spelling corrector as a preprocessing step which helps in decreasing the ambiguity caused by normalising Hamza and Alif letters.

TAGSET: This tagset has a complex compound tagset: (>3000 tags[22]). Each particle has its own tag (Laam particle is tagged Laam). Without official documentation and because of the limited usage quota, it is hard to estimate the number of core tags.

4.4.7 Segmenter and Part-of-speech tagger for Arabic (WP)

Segmenter and Part-of-speech tagger for Arabic (Gahbiche-Braham *et al.*, 2012) is a tool that uses a CRF model trained on PATB using the Wapiti toolkit[23]. The tool has two components: one to predict the POS tag and the second is to split the enclitics. The reported accuracy of predicting the correct POS tag is 96.38%.

TAGSET: WP used the list of main 24 POS tags of PATB, with 3, 6, and 2 for conjunctions, prepositions, and determiner prefixes respectively.

4.4.8 Farasa POS tagger (FA)

Farasa (Zhang *et al.*, 2015) is a toolkit for segmentation/ tokenisation module, POS tagger, Arabic text diacritiser, and dependency parser. Farasa is different from other POS taggers as it can jointly segment, POS-tag, and parse the text which avoids error propagation in the pipelined structure and should exploits syntactic information for POS tagging. It is particularly useful for tagging CA as CA is different in vocabulary from MSA, but it shares similar syntax. The reported accuracy of predicting the correct POS tag of MSA is 97.43% and of CA is 84.44%.

TAGSET: Farasa has a tagset of 16 basic tags.

4.5 Discussion

While POS taggers and morphological analysers predict the main POS tag, they vary in fine-graininess of tagset and segmentation. They differ in many aspects: tagset used, output format, the method used, and tokenisation. Most taggers adopt their own tagset, and they subsequently assume their tokenisation scheme. Table 4.3 and Table 4.4 lists supported features by each morphological analysers and POS tagger. Most taggers produce their results in their customised format as shown in Appendix B.

---

[22] https://www.microsoft.com/en-us/research/project/part-of-speech-pos-tagger/

[23] https://wapiti.limsi.fr/

Table 4.4 The result of POS taggers, for each input word.

Name	MD	AM	MX	ST	MT	MR	WP	FA
Base POS tag	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Glossary	Yes	-	Yes	-	-	-	-	-
Aspect	Yes	Yes	Yes	Yes[24]	Yes	-	-	-
Person	Yes	Yes	Yes	-	Yes	-	-	-
Gender	Yes	Yes	Yes	-	Yes	-	-	Yes[25]
Number	Yes	Yes	Yes	Yes[26]	Yes	-	-	Yes[25]
Transitivity	-	-	-	-	-	-	-	-
Voice	Yes	Yes	Yes	Yes	Yes	-	-	-
State	Yes	-	Yes	-	Yes	-	-	-
Mood	Yes	-	Yes	-	Yes	-	-	-
Case	Yes	-	Yes	-	Yes	-	-	-
Pattern	-	-	-	-	-	-	-	-
Root	-	-	-	-	-	-	-	-
Stem	Yes	-	Yes	-	-	-	-	-
Lemma	Yes	-	Yes	-	-	-	-	-

To show the differences in context, Appendix A presents one Hadith (an utterance attributed to prophet Mohammed often called "prophet sayings") sentence annotated by each tagger. The sentence was extracted from the prophet Mohammed sayings (classical Arabic): لا يُؤْمِنُ أَحَدُكُمْ حَتَّى يَكُونَ هَوَاهُ تَبَعًا لِمَا جِئْتُ بِهِ, */lA yu&ominu >aHadukumo Hat~aY yakuwna hawaAhu tabaEFA limaA ji}otu bihi/* (None of you [truly] believes until his desires are subservient to that which I have brought). The sentence is fully vowelized, including the ending vowel. However, some taggers (ST, MR, AR, BP, KH) performed better when vowels are completely removed, probably because they were trained on undiacritised texts or the ending vowel is not expected.

We used a slightly edited version of CoNLL-U format to represent the tagged sentence using MAs and POS taggers. We added one column (the 1st) to represents the tagger name and dropped the 3rd, 7th, 8th, and 9th irrelevant columns.

[24] Yes unless it is passive: verb mood cannot be determined.

[25] only for nominals.

[26] Number is either singular or plural.

Since MAs do not disambiguate, we manually picked the most-correct analysis. The last column shows the selected analysis and the number of alternative analyses.

This conversion is not straightforward. We had to deal with a number of different output formats. In addition, the morphological features values were unified for a direct comparison. We had to deal with different transliterations and representations: e.g. we extracted clitics from word-based taggers, we extracted morphological features from compound-tag (e.g. word #5 and `IV3MS`) taggers. The converter software to CoNLL-U format, XML and JSON is freely available and open-source[27].

In the following excerpts, a simpler format is used that highlights only the morphological analysis aspect in the discussion, using a list of word form and POS tag separated by a slash. The plus sign at the beginning/end of word form indicates a proclitic/enclitic.

The results presented on Appendix A shows that:

i. Not only POS tags are different, but the word segmentation as well (word #2).

One Segment	Two Segment
AL: yu&omin/verb AR: >Amn/verb	BP: yu+/IV3MS &omin/VERB_IMPERFECT MS: _+/PREFIX_YA2_ANAIT_MA3LOOM_MAGHOOL yu&omin/FE3L_MODARE3_MAZEED

ii. Word #10 shows that the definition of the main segment is not standard: is it the PREP or the PRON? This can cause problems when evaluating different lemmatisers/stemmers for example.

Two proclitic	Stem + Enclitic
FA: b+/PREP +h/PRON ST: b+/NN +h/NN AM: b+/IN +h/PRP_MS3	BP: bi/PREP +hi/PRON_3MS MD: bihi/prep +/3ms_pron

iii. Some taggers do not recover the word's clitics. Instead, it reports the POS tag of such clitics. Some others try to recover the original form of the word before concatenation. Aligning such taggers with others cannot be done intuitively.

[27] http://github.com/aosaimy/sawaref-web

No Form	MT: hawaAhu/Ed -/N -/Poss MX: hawAh/noun -/3ms_poss
Form Segmentation	ST: hwA/NN +h/PRP$ FA: hwA/NOUN +h/PRON
Form Restoration	AM: hwY/NN +h/PRP MR: hwy/NN +h/PRP EX: hawaY/N- hu/SP

iv. Two tokens sometimes are given one tag (KH analysis of word #10) even though the tag explains the two tokens: "a preposition and its pronoun".

Two POS tags	KH: bihi/jAr_wmjrwr
Single Tag/Segment	EX: bi/P- hi/SP FA: hwA/NOUN +h/PRON

v. Some segmentations are for affixes, not clitics (word #7). INDEF tag is related to the first segment though.

Affix-based	FA: tbE/NOUN-MS +A/CASE BP: tabaE/ADV AF/NSUFF
Clitic-based	AM: tbEA/NN MX: tabaEAF/noun_(CASE=ACC)

vi. The convention of diacritisation is not standard. For example, look at short vowels before long vowels (word #1) and *tanween* location (before or after Alif letter) (word #2). Normalisation is required if a comparison is to be performed.

Long vowels	BP: lA/NEG_PART	EX: laA/F-
Tanween	AL: tabaEAF/adv BP: tabaE/ADV AF/NSUFF	EX: tabaEFA/N- MS: tiboEFA/Asm_jAmd

vii. Features and POS tags are not always consistent between different taggers. For example, the morphosyntactic features of verbs and its subject may be segmented and not.

Single	AL: ji}ota/verb_(Gender=M|Number=S|Mood=IND| Aspect=PERF|Voice=ACT|Person=2) EX: ji}tu/VP_(Gender=M|Number=S|Aspect=PERF|Voice=ACT|Person=1) AR: jA'/verb_(Aspect=PERF|Voice=ACT|Person=1)
Segmented	ABP: ji}/VERB_PERFECT_(Aspect=PERF) tu/PVSUFF_SUBJ:1S_(Number=S|Voice=ACT|Person=1) MS: ji}otu/FE3L_MADI_MOGARRAD_(Aspect=PERF) _/SUFFIX_TA2_FA3EL_MOTAKALLEM_(Person=1) MT: ji}out/V_(Number=S|Aspect=PERF|Voice=ACT) _/Subj_(Number=S|Person=1)

viii. Some diacritics are dropped in the morphological analyser. It is not due to input normalisation at the beginning of the morphological process but in the

tool's processing of the word form. The processing includes some spelling changes in short vowels, Hamza letters, and Alif/Yaa Maqsourah. This inconsistency complicates the comparison of outputs using lexical forms.

Word Form		
	AM: j}t	MR: jt
	FA: j} +t	ST: j}t
	MX: ji}otu	WP: ji'tu
	MD: ji}otu	MT: ji}out

We noticed that in many cases, the first suggested analysis is the correct one: this is because of some ways MAs sort alternative analyses. However, this should not be confused with POS taggers as POS taggers use the *context* to rank alternative analyses.

4.6 Tagging Classical Texts

In optimal cases, evaluating a list of POS taggers requires a gold standard test dataset, that has a standard tagset and segmentation (tagger's output should be mapped, otherwise). However, the reported accuracies of taggers fail to adhere to these three conditions. Besides, most surveyed tools are designed primarily for MSA, including their test datasets. The commonly-used dataset for testing is parts of PATB, which has most of its content is news articles. In this thesis, the performance of these tools is analysed in classical Arabic. The goal is to compare more taggers on a sample of CA concerning accuracy and coverage. A direct automatic evaluation is not possible (Paroubek, 2007).

4.6.1 Methodology

As mentioned earlier, Alrabia (2014) showed that CA has a worse POS tagging accuracy for MD and KH tools. They overcome the issue of different tagsets by learning each tagset and validating each tagger against its own tagset. Therefore, the reported accuracy can be compared to their published accuracies. However, the reported accuracies should be taken with caution when comparing taggers to each other as they adhere to different linguistic schemas.

Their work is limited to only two taggers. In this thesis, more taggers are included. The approach is similar to their approach, but with a smaller data set. Our approach focuses on a subset of words that looks classical. This decision is to minimise the effort and improve the quality of the analysis. A word is assumed to

be classical if it appeared on a classical Arabic corpus but not a contemporary corpus. To formalise this assumption, let C be the set of words in a classical corpus, and M be the set of words in an MSA corpus, the set of classical words are $W = C - M$.

This methodology makes performance measures intentionally biased to classical Arabic and not necessarily comparable to their previously published work. For example, frequent words (usually not (out of vocabulary) OOV) contributes to good accuracy, but these words are excluded in our case. This methodology should as well give some insights into the similarity between classical Arabic and MSA and the richness of Arabic lexicon of classical Arabic.

Since the word list is extracted with no context, their POS tags are not determined. It is common for one word in Arabic to have a list of possible tags which is required for reporting accuracy as it is a contextual measure. The accuracy measure is defined by the average prediction accuracy of the POS tag of the word in 10 occurrences, i.e. 10 concordances are extracted from the classical corpus subset, and checked if the proper POS tag is given correctly by the analyser.

4.6.2 Data

The classical corpus used for C is a *subset* of OpenArabic Corpus (Dmitriev, 2016). It categorised classic books into centuries and provided word frequencies for each book with and without normalisation. The subset is conditioned books that are written in the first seven centuries (1075 books). The contemporary corpus used for M is the Corpus of Contemporary Arabic (Al-Sulaiti and Atwell, 2006). W is capped to the top 500 words.

The final list of words have some issues: 30% of the words are proper nouns which may suggest the need for gazetteers for classical Arabic proper nouns. It is particularly useful because Proper nouns in Arabic are not marked (i.e. they are not capitalised). Unlike common nouns, grammatical features of proper nouns are sometimes lexical.

The word frequencies reported by OpenArabic is a simple word frequency, instead of the term frequency-inverse document frequency (TF/IDF). This choice raised some words that are highly frequent but only on certain books (e.g. dictionaries like (بضم /*bDm/* with a Dammaah vowel), prophet sayings like (ثنا /*vnA/* he reported), bibliography like some proper nouns).

One drawback of this methodology is the incapability of handling different inflexions, especially with highly inflectional languages like Arabic. Some contemporary words are found in the final list, as they appear in inflected forms that did not appear in the contemporary corpus.

4.6.3 Evaluation

In this experiment, we report the performance in two folds: the performance of morphological analysers and POS taggers. In morphological analysers, we compare the accuracy and coverage of these analysers, while in POS tagger we only report the accuracy. POS taggers tag each word even if it is OOV, so no coverage is reported. The OOV rate is not available due to the unavailability of their training dataset.

In the morphological analysers, the accuracy of tagging these words is reported, in addition to the rate of out of vocabulary (OOV) words, analysis time measured in seconds, average number of analyses per word, and the average number of lemmas per word. See Table 4.5. AL used *backoff* strategy when no analysis was found in the dictionary (so OOV rate is zero). QT does not provide lemmatisation.

Table 4.5 The rate of out of vocabulary (OOV), accuracy, analysis time, average number of analyses/lemmas of analysing 50 common classical words.

Tool	AR	AL	KH	EX	BP	MS	QT
OOV rate	0.228	0	0.058	0.076	0.084	0.052	0.82
Accuracy	56%	88%	90%	84%	88%	82%	N/A
Analysis Time (in secs)	0.255	4.324	3.453	177.465	1.061	N/A[28]	0.766
Avg. Analysis/Word	2.06	7.32	14.25	17.89	2.44	1.86	4.27
Avg. Lemmas/Word	1.5	2.53	4.51	2.61	2	1.53	1

The second fold is evaluating the performance of POS taggers. Because of the high appearance of proper nouns, the accuracy of tagging this specific tag is reported. Table 4.6 shows the overall and proper nouns accuracies.

Proper nouns were rarely tagged correctly by MAs. Alkhalil seems to have a list of classical proper nouns (gazetteers) as it performed the best in this matter. The identification of personal names is challenging for several reasons: the absence of a proper mark, nominals acting as proper nouns (adjectives, nouns, participles, and

[28] Not available as it is a web-based service.

even inflected verbs), and phrasal names (teknonymics, patronymics, matronymics)[29] (Ryding, 2005).

Since each tagger has its own labelling schema, marking the tag as either correct or not is not easy, as it requires a thorough understanding of the tagset. The marking was done by the author of this thesis, who manually checked each tagger's output. A tagger has to identify all clitics correctly and assign each clitic its proper POS tag. No other morphological analysis is included in this experiment.

Some sources of incorrect tagging were as follows:

- **Obsolete forms:** One adverb was only tagged correctly by one analyser, as this adverb is obsolete. Some patterns as well were not identified as the broken plural pattern is obsolete (like القرأة *Alqr>p* (the readers))

- **Normalisation:** e.g. Converting Yaa Maqsourah to Yaa, a proper noun was not tagged properly.

- **Orthography and spelling:** e.g. Classical tokenisation of يا أيها /*yA >yhA/* (O (mankind)) was written jointly unlike it usually is in MSA.

Table 4.6 gives evidence that one POS tagger performs better in some tags than the other. The MADAMIRA toolkit (MX) performed poorly with classical proper nouns; however, it outperforms other taggers in tagging other words. On the contrary, the Stanford POS tagger (ST) (and Alkhalil tagger, KH) performed better in proper nouns. These different tag-specific accuracies suggest that an ensemble POS tagger could increase the accuracy of POS tagging, maybe with some attention to tagger's strengths.

Table 4.6 The accuracy of POS taggers of tagging 50 classical words within three sentences per word extracted from classical books.

Tool	MD	MX	ST	MR	WP	AM	MT	FA
Overall [30]	69.6%	70.6%	78.4%	66.7%	68.6%	79.4%	67.6%	74.5%
No Prop Nouns (57%)	80.0%	78.5%	71.4%	52.8%	58.5%	74.2%	87.1%	74.2%
Prop. Nouns (43%)	46.8%	53.1%	93.7%	96.8%	90.6%	90.6%	25.0%	75.0%
Reported Accuracy	96.1%	95.9%	96.5%	96.43%	96.38%	96%	N/A	97.43%[31]

[29] It is not uncommon in Arabic to have proper names derived from mother's child name (matronymics), father's child given name (patronymics), or father's given name (teknonymics).

[30] This accuracy can be seen as the OOV accuracy, as our methodology limits the test dataset to words that have not appear in a contemporary corpus. Therefore, it should not be directly compared to reported accuracies, but to their OOV accuracies, which are not reported for most of these tools.

4.7 Conclusion

POS taggers and morphological analysers differ in many aspects. While they both predict the main part of speech tag, they vary on what morphological and word features they also predict. Most taggers adopt their own tagset, and they subsequently assume its tokenisation scheme. With a focus on tagging classical Arabic, the accuracy and coverage have dropped to a low score. The average drop is at least 20%. As a result, annotation of classical Arabic text should either adopt its own new morphological analyser or improve current ones to support classical Arabic. One potential solution is to combine those taggers into one system which should increase the coverage and accuracy levels.

[31] FA was tested on a classical Arabic corpus as well and the reported accuracy is 84.44%.

Part II

Ensemble Morphosyntactic Tagger for Classical Arabic

5 ENSEMBLE TAGGER DESIGN FOR CLASSICAL ARABIC

Chapter Summary[1]:

In Modern Standard Arabic text (MSA), there are several morphological resources, but none is designed and tuned primarily for classical Arabic. The goal of our language resource is to build a freely accessible multi-component toolkit (named SAWAREF[2]) for part-of-speech tagging and morphological analysis that can provide an easy interface for several taggers, compare and evaluate between them, standardise their outputs of each component, combine different solutions, and analyse and vote for the best candidates. We illustrate the use of SAWAREF in tagging adjectives of classical Arabic. This chapter describes the research method and design and discusses the critical issues and obstacles.

---

[1] Some parts of this chapter are based on:

Alosaimy, A. and Atwell, E. (2015) 'A Review of Morphosyntactic Analysers and Tag-Sets for Arabic Corpus Linguistics', in *Eighth International Corpus Linguistics conference (CL2015)*, pp. 16–19.

Alosaimy, A. and Atwell, E. (2016) 'Ensemble Morphosyntactic Analyser for Classical Arabic', in *Second International Conference on Arabic Computational Linguistics*. Konya, Turkey.

Alosaimy, A. and Atwell, E. (2018) 'Diacritisation of a Highly Cited Text: A Classical Arabic Book as a Case', in *2nd IEEE International Workshop on Arabic and derived Script Analysis and Recognition (ASAR 2018)*. London, UK.

[2] SAWAREF toolkit: sawaref.al-osaimy.com.

5.1 Introduction

The Arabic language has several variants where each has its own characteristics in morphology, lexicon and syntax. Classical Arabic, Modern Standard Arabic (MSA) and Dialectal Arabic have been written in different genres and media: from social networks to newspapers to journals. Researchers tend to build POS taggers for specific variant or dialects. Adapting one or several existing taggers to another domain/genre saves time and effort. While several POS taggers for MSA exist, none exist for classical Arabic to the best of the author's knowledge. Moreover, many of them are incompatible: incompatible tokenisation and various tagsets. The ultimate goal of our system is to build a methodology of combining black-box POS taggers; hence, a more robust tagger.

The outline of this chapter is as following. First, we formally define the problem and propose a general design and methodology of a black-box ensemble system for transferring the knowledge to a low-resource variant of Arabic: e.g. classical Arabic. Then, we start this chapter by describing the challenges that faced the development of the ensemble system (Section 5.3). Section 5.4 describes each stage in more detail.

Then, we report the results of three experimental studies. In Section 4.5, we report and analyse the results of mapping one tagset to another. In Section 5.6, we take a closer look at the approach of one stage: Diacritisation. Next, one potential use of the system (comparative evaluation of taggers) is illustrated by evaluating the case study of tagging adjectives (Section 5.7).

5.2 Problem Definition and System Overview

The tasks in this thesis can be divided into high-level and low-level categories. The high-level, i.e. the final system outcomes, are: the prediction of the word segments (or **segmentation)**, and various predictions of POS tags and morphological features (or generally **tagging)**.

The **segmentation** problem can be seen as either *boundary identification* or *word segments restoration*. The boundary identification problem is a classification problem where the task is to mark the first letter of each segment. For example, the position of the first segmental letters of *cannot* are the underlined first and fourth letters. However, word segment restoration recovers the word segments; e.g. the word *don't* is recovered into two segments: *do* and *not*. In this thesis, segmentation is

referred to the latter definition. However, it is worth mentioning that most taggers use the former definition. In the former word *boundary identification*, the problem is a binary supervised sequence labelling. Given a sequence of characters, $c = \{c_0 \dots c_i \dots c_n\}$ where n is word length, the task is to predict a sequence of labels with length n with the label set $L = \{0,1\}$. The latter definition is more complicated: the task is to predict an unknown-length set of unknown-length sequences of characters. Similar to translating one sentence to another, it translates the lexical form of one word to its original word form. Some work such as Darwish and Mubarak (2016) formed the problem as a classification problem: the task is to rank and select the most probable segmentation from a list of possible segmentations. The list can be edited to help restore the original segments.

The **tagging** problem is a set of predictions on the *segments* of the word, i.e. segmentation problem outcomes are pipelined in to the tagging problem. Although this problem could be performed on the word level (some tools already do that), we define the tagging problem as a supervised multioutput-multiclass labelling problem of each *segment.*

The two problems can be done simultaneously, i.e. joint segmentation and tagging by defining the problem as a character-based classification task of character position and label. Each character is tagged according to its corresponding *morpheme label* in addition to a *boundary tag* that indicates its relative position. More details will be discussed in form-based ensemble (Section 6.5). This method, however, does not recover adjusted word form.

In both high-level tasks, the feature selection (in the sense of machine learning) can vary according to the design of the model. In our ensemble problem, all the outputs of individual taggers may be used, including segmentation and tagging. In this chapter, the overall design of this model is described. Some adaptation to the model is proposed in the following chapters.

High-level tasks involve several low-level tasks including the alignment problem, diacritics restoration, word form-based similarity measurement, and tagset mapping. These tasks will be defined later to put them in context.

The framework that combines all individual taggers is called SAWAREF[3]. SAWAREF has an interface web-based system that can run seven morphological analysers, namely:

- AlKhalil (KH) (Boudlal *et al.*, 2010),
- Buckwalter (BJ) (Buckwalter, 2002b),
- Elixir-FM (EX) (Smrz, 2007),
- Microsoft ATKS Sarf (MS),
- ALMORGEANA (AL)(Habash, 2007),
- AraComLex (AR)(Attia, Pecina and Toral, 2011), and
- Xerox (XE) (Beesley, 1998).

Also, it can run seven POS taggers, namely:

- Madamira (MX) (Pasha *et al.*, 2014), MADA (MD) (Habash, Rambow and Roth, 2009),
- AMIRA (AM) (Diab, 2009),
- Stanford POS tagger (ST) (Toutanova *et al.*, 2003),
- Microsoft ATKS POS Tagger (MT) (Kim, Snyder and Sarikaya, 2015),
- Farasa (FA) (Zhang *et al.*, 2015),
- MarMoT (MR) (Mueller, Schmid and Schütze, 2013), and
- Wapiti Arabic Model (WP) (Gahbiche-Braham *et al.*, 2012).

The framework provides a simple convenient interface for comparing between taggers and evaluating them. It is not meant to be compared with those taggers: instead, it provides a range of useful tools to compare them against each other. The toolkit contains several tools:

- *a parser*[4] *tool* that reads the different formats of these taggers,
- *a standardiser component* that converts them to a standard morphological representation using mapping rules,

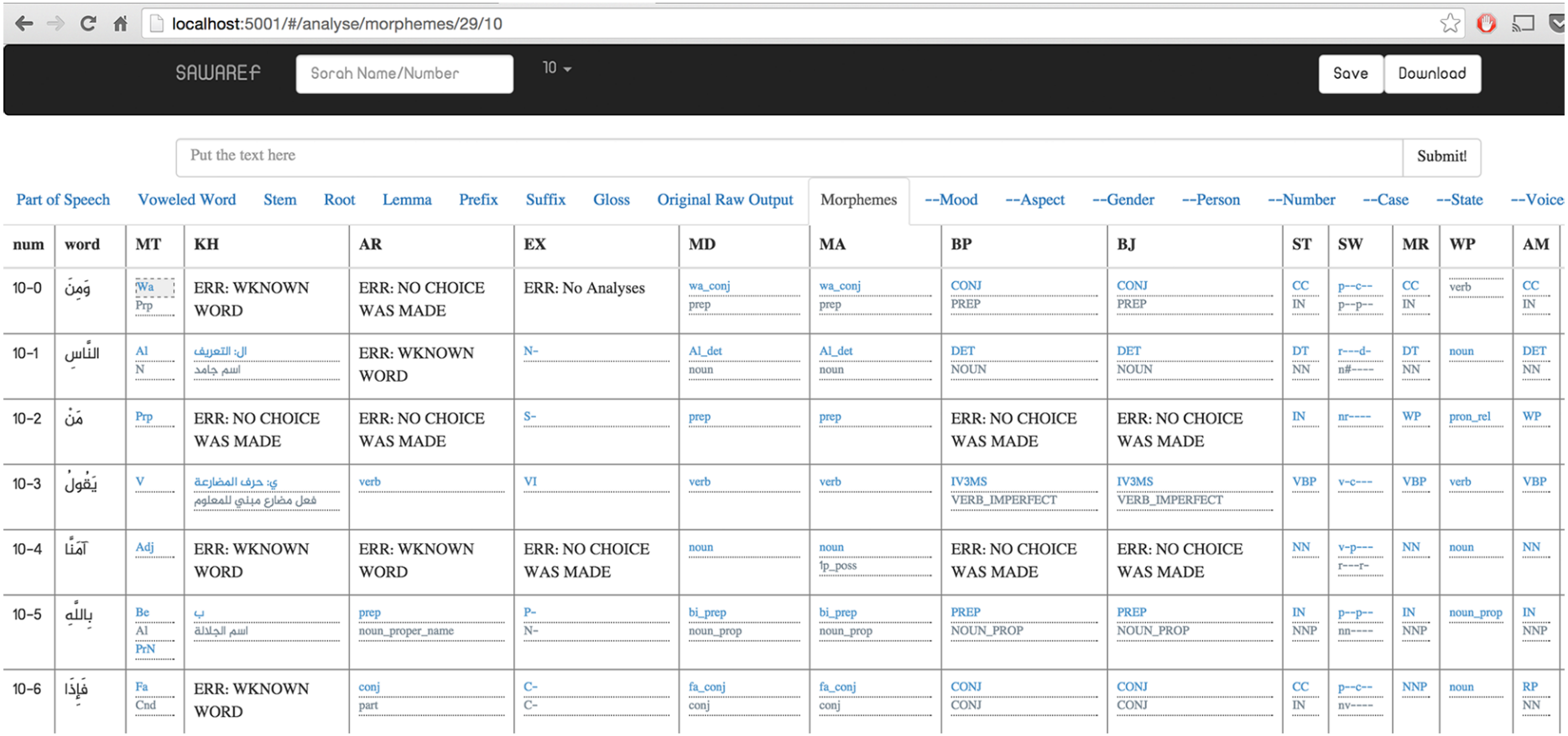
---

[3] The name is not an acronym. It is a transliteration of the Arabic word صوارف, (distractor). Morphology in Arabic is called صرف/sarf/ and its plural form is /soruof/, although both share the same root. The name is meant to show how the *pattern* of the word plays a critical role in the comprehension of Arabic words.

[4] *Parsing* and *parser* should not be confused with the linguistic meaning of syntax analysis. Syntax analysis is not out of the scope of this thesis. Parsing refers to the computational process of converting raw text outputs from one tagger into a machine-readable format.

- *a Mapper web-based interface* for mapping rules creation,
- a CoNLL-U format converter,
- *a word alignment tool* that preserves the same number of words from each tool,
- *a morphological alignment tool* that tries to map a series of morphemes to their equivalent on another tagger,
- *a disambiguation tool,* or *the ensemble tagger* that predicts the proper analyses given the taggers' analyses, and
- finally *a web-based viewer* to compare and check results interactively (see Figure 5.1).

These tools are written to be used independently following the Unix tools philosophy. Each tool is designed to perform a specific task, and one tool output can be pipelined in to another tool. This philosophy allows the task to be developed and tested independently and its output to be examined easily.



num	word	MT	KH	AR	EX	MD	MA	BP	BJ	ST	SW	MR	WP	AM
10-0	وَمِنَ	Wa Prp	ERR: WKNOWN WORD	ERR: NO CHOICE WAS MADE	ERR: No Analyses	wa_conj prep	wa_conj prep	CONJ PREP	CONJ PREP	CC IN	p--c-- p--p--	CC IN	verb	CC IN
10-1	النَّاسِ	Al N	ال: التعريف اسم جامد	ERR: WKNOWN WORD	N-	Al_det noun	Al_det noun	DET NOUN	DET NOUN	DT NN	r---d- n#----	DT NN	noun	DET NN
10-2	مَنْ	Prp	ERR: NO CHOICE WAS MADE	ERR: NO CHOICE WAS MADE	S-	prep	prep	ERR: NO CHOICE WAS MADE	ERR: NO CHOICE WAS MADE	IN	nr----	WP	pron_rel	WP
10-3	يَقُولُ	V	ي: حرف المضارعة فعل مضارع مبني للمعلوم	verb	VI	verb	verb	IV3MS VERB_IMPERFECT	IV3MS VERB_IMPERFECT	VBP	v-c---	VBP	verb	VBP
10-4	آمَنَّا	Adj	ERR: WKNOWN WORD	ERR: WKNOWN WORD	ERR: NO CHOICE WAS MADE	noun	noun 1p_poss	ERR: NO CHOICE WAS MADE	ERR: NO CHOICE WAS MADE	NN	v-p--- r---r-	NN	noun	NN
10-5	بِاللَّهِ	Be Al PrN	ب اسم الجلالة	prep noun_proper_name	P- N-	bi_prep noun_prop	bi_prep noun_prop	PREP NOUN_PROP	PREP NOUN_PROP	IN NNP	p--p-- nn----	IN NNP	noun_prop	IN NNP
10-6	فَإِذَا	Fa Cnd	ERR: WKNOWN WORD	conj part	C- C-	fa_conj conj	fa_conj conj	CONJ CONJ	CONJ CONJ	CC IN	p--c-- nv----	NNP	noun	RP NN

Figure 5.1 A screenshot of the SAWAREF web-based interface. The top bar is for navigation through documents and running analysers on a given text. The tabs represents different outputs of the analysers. The tabular view shows how each analyser is analyzing the sentence presented on a vertical mode.

Figure 5.2 The overall process of the ensemble system: SAWAREF.

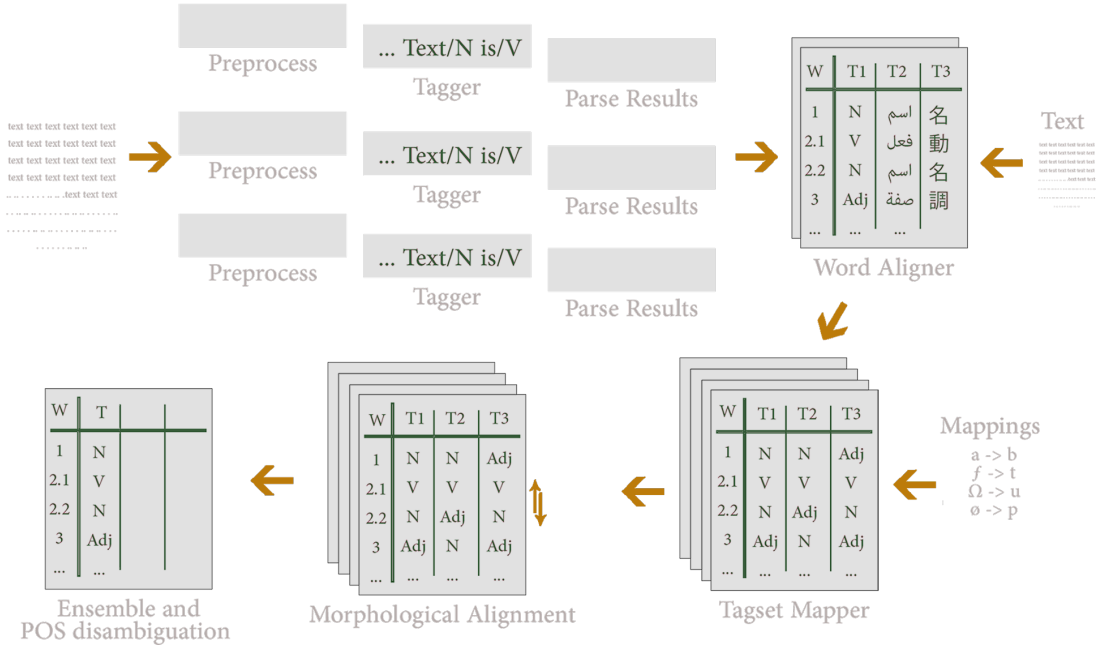


Figure 5.2 illustrates the overall process of the ensemble system. The process starts with the text to be tagged being sent to a pre-processing component for each participating tagger. The results are parsed using *the parser tool* and then sent to a *word-aligner* tool that aligns the results at the word level. Next, the system may use the mapping list to standardise the outputs. Each solution is then optionally aligned with other solutions using the *morphological aligner* tool. Finally, we use different ensemble methods to produce the most confident tagging and segmentation.

The framework can be useful for other applications in different stages. For example, it provides a high-end interface to individual taggers, which can be used to perform evaluation of taggers and ease the choice of a tagger for specific research needs.

5.3 Challenges

Any heterogeneous ensemble faces the problem of projecting input (or individual) components into one standard schema. Specifically, ensemble morphological analysers face problems due to the variance in spelling and orthography, labelling standards and segmentation schemas.

5.3.1 Diverse Output Format

Almost every tool has its own format of the output. Some tools use popular machine-readable formats like table-like CSV (Alkhalil), XML (Qutuf,

MADAMIRA), or JSON (Xerox). MADA returns a sequence of *feature:value* pairs. However, some tools have a more complex output like BAMA and AraComLex. We needed to build a custom parser explicitly designed for their outputs.

The parser component of SAWAREF translates the custom outputs of each tool to an open standard format: JSON and CONLL-U. This standard format eases the exchange of the output of these tools with other downstream products. As a consequence, the infrastructure needs to be updated every time one of the tools changes its output scheme.

5.3.2 Tools and Resources Availability

Although many researchers published papers about their morphology tools, many of these are either not available, require a licence or are limited to specific uses or bandwidth. For example, although the MADA toolkit is freely available, it requires lexicon tables that are only available with membership of the LDC until version 3.2 is released. Besides, some web services such as Xerox and Microsoft toolkits are limited to some usage quotas.

5.3.3 Different Data Distributions

Although CA is considered the father of MSA, MSA and classical Arabic have different data distributions. Many lexical words and phrases that were used in classical Arabic are no longer used in MSA. Because of that, the ensemble case in this thesis considers another aspect of adaptation: domain adaptation. Please note that some taggers are provided as black-box taggers and are not retrainable on a new training dataset. Some others are, as well, limited to specific annotation style, because they integrate external lexicon and morphological analysers.

5.3.4 Different Word Segmentation

For a valid comparison, words need to be similarly segmented. One approach is to segment the input in advance and supply the POS tagger with a segmented input. However, most tools jointly segment and tag the input, and therefore they cannot accept a segmented text. Even in cases where a segmented input is expected (e.g. Stanford POS tagger), the input has to conform to a specific segmentation schema.

5.3.5 Different Labelling Systems

Although there are many suggested tagsets in the literature, e.g. (Khoja, 2001; Sawalha and Atwell, 2013; Elhadj, Abdelali and Ammar, 2014; Zeroual, Lakhouaja and Belahbib, 2017), Arabic POS taggers suffer from not having a standard tagset. One reason is that researchers have different intentions and different views of the rich morphological nature of the language. The differences of heterogenous tagsets are derived from four aspects: their representation, comprehension, size, and convention.

First, tagsets are different in their representations. They can be classified into two categories: *pos+features* tagsets (e.g. CONLL-U tagset) and complex one-word tagsets (e.g. Buckwalter tagset). In *pos+features,* tagsets are explicitly distinguished from morphological features (which is named explicitly): *noun, Gender=Masc*. In complex tagsets, the tag encodes multiple information with no predictable format: *NSUFF_FEM_SG*.

Second, The non-standard tagsets introduces a challenge of understanding each one. Each tagset is developed using some underlying linguistic theory. However, tagsets usually do not name nor explain this theory. See Section 5.7 for an example of different definitions of Arabic adjectives.

Third, tagsets vary wildly in their sizes. The Buckwalter tagset, for example, can hypothetically reach over 330,000 tags (Habash, 2010), while the Stanford tagger used a reduced Bies tagset that has around 20+ tags.

Lastly, tagsets usually have implicit conventions. Tags tend to be short for presentation purposes, and sometimes they are misleading or incomprehensible, e.g. the “NSUFF” tag which stands for a nominal suffix. Fully understanding one tagset requires a good documentation.

Some tagsets use the notion of default value for compactness purposes, e.g. “NN” stands for *singular* common nouns, which may confuse users with other situations where the number is not applicable. Another example: the PRON_2D tag for Arabic (a second-person dual pronoun) is missing the gender feature which might be assumed to be masculine. However, the gender feature is not applicable in this dual case due to the nature of Arabic. These sometimes are not mentioned in the documentation, which makes the mapping between tagsets or standardising them a challenge. Some tagsets are improved or developed over time, and the published tagset in an academic article is incomplete.

5.3.6 Converting Complex POS Tags

Although some tools do not explicitly present some essential features such as gender, number and person, these features can be extracted from their complex one-word POS tagset; however, this extraction process needs a careful *understanding* of the POS tags.

Complex tags usually do not name the feature, which makes tags less comprehensible. For example, *V.past* could refer to a past tense verb or a perfect aspect verb (called past in traditional Arabic).

5.3.7 Different Possible Configurations

Some tools have different possible hyperparameters for different stages of morphological analysis, e.g. MADA input can be preprocessed in three different ways. Different configurations lead to different tokenisation, and therefore different analysing and performance. Although these configurations are documented, the different combination of configuration values may have some impact on the ensemble analyser. However, this increases the hyperparameter space to a high degree. We choose to use the default settings and leave comparing different configurations for future work.

5.3.8 Expectancy of Input

While some tools expect unvoweled text data (AraComLex), some accept wholly or partially voweled data such as AlKhalil. ATKS uses these short vowels to filter the best analyses if it fits or the diacritics will be ignored. Mada expects the input text to be text-only one sentence per line with no tags or metadata. AraComLex expects every word to be in a single line. The Stanford tagger expects tokenised words.

5.3.9 Different Transliteration Schemes

Different tools encode the results in either ASCII or UTF-8. Some use a one-to-one transliteration scheme like Buckwalter transliteration (which has received several extensions, and determining which extension can be tricky). Other tools like Elixir uses ArabTex encoding whose transliteration is governed by a set of complicated rules.

5.3.10 Different Spelling Schema

There are some differences in the processing of the spelling of the input, due to different standards in processing Hamza, Taa Marbouta and diacritics. The spelling inconsistency complicates the matching between their output. A post-processing normalisation step is sometimes required. For example, the convention of diacritizing /F/ when it is attached to /A/. More details are in Section 8.9.

5.4 Tagging Stages

5.4.1 Diacritisation

One optional preprocessing step of the input text to all taggers is improving the phonological information of the text, i.e. diacritizing the text by adding short vowels to its non-diacritised letters.

In this stage, we do not aim at automatic diacritisation; instead, we aim to raise the diacritisation coverage level by "borrowing" diacritisation from similar contexts with high confidence of accuracy. Raising diacritisation level reduces the word ambiguity level, which improves taggers accuracy (See Section 5.6 for experiment results).

5.4.2 Pre-processing

Most of the time, each component does the required pre-processing step on its own. That is, it transliterates, normalises, spell corrects, and tokenises the input text in the format suitable for the component's needs.

However, after a series of tests to maximise the accuracy, we found that some poorly-documented taggers assume input in certain conditions. Some components work better when diacritics, digits, or punctuations are deleted, the text is normalised, or text is transliterated. In general, we followed the documentation requirements, if such existed, and pre-processed the input the way it achieves maximal accuracy (by iterative random samples evaluated manually).

TOKENISATION: Tokenisation is well-known to be difficult in Arabic because writers often omit word spaces next to non-joining letters. Tokenisation on whitespace and punctuation, therefore, introduces many errors on all but the most carefully written texts. However, our system assumes that every tool has its own word and morpheme tokenisation. One tool–MarMoT–required the input to be tokenised and we used the AMIRA word tokeniser. Some adaptions are required: we

deleted signs that indicate affix type. The Stanford POS tagger requires the text to be tokenised using the Stanford Word Segmenter (Monroe, Green and Manning, 2014)[1]. AraComLex assumed the text to be tokenised–each word in a line.

TRANSLITERATION: We transliterate the input if the tagger does not support the UTF-8 format (e.g. MarMoT and BAMA) using the two-way table-lookup transliteration system based on the Buckwalter convention.

5.4.3 Component Manipulation

Running: Most of the tools are runnable through the command line. Some components have an API (e.g. Madamira and Stanford Segmenter) that allows them to be integrated into the developer's code. One component (Alkhalil) is only runnable through a Graphical User Interface (GUI). To integrate into the SAWAREF system, we added the functionality to permit it to be run from the command line without interfering with the analysis code.

Wrap-To-Service: Since we plan to allow the usage of these tools from the web, we wrap each component in a service. The goal here is to speed up the processing of texts by having the morphology model loaded and ready for each subsequent request. We build a web service for each tagger. It accepts HTTP requests and returns component output while maintaining dictionaries in memory.

Special Modifications: In the Alkhalil morphological analyser, if a word reappears in the text, it will be ignored, and no analyses will be given. We modified the Alkhalil toolkit source code to print the analyses of each word on every occasion, allowing us to align the analyses with other components' results.

Besides, the word's type and POS tag in the Alkhalil toolkit are printed in free text as it is meant to be easily read for Arabic linguistics. Free text is converted into a structured format by carefully examining the source code and some pattern lookups.

5.4.4 Standardizing Results and Extracting Morphological Features

Every component has its own format of output (Appendix B). We built several parsers that extract analysis for each tagger and transform them to a standard JavaScript Object Notation (JSON) object. This representation can be converted into

[1] The Stanford Word Segmenter processes raw text input according to the Penn Arabic Treebank standard (Diab *et al.*, 2013).

comma-separated-values (CSV), CoNLL-U, and XML formats. The goal is to standardise the format so that they can be reused for evaluation and ensemble tagging purposes.

For each morpheme, SAWAREF maintains the following outputs, whenever they exist:

Morpheme-based Basic POS tag: The part of speech tag XPOS (given by the analyser) and its matching universal POS tag (UPOS).

Morphological Features: Person, gender, number, aspect, definiteness, state, voice, mood and case.

Morphological Segmentation: How the word has been segmented.

Word-level Analysis: Root, Stem and Lemma.

Since the outcomes of each tagger are standardised, we were able to show them in a convenient side-by-side way on the web interface that allows any researcher to study these taggers and see their features (what features they are extracting), the accuracy of POS disambiguation, its tokenisation scheme, and more.

Within this step, the result of taggers with a one-word complex tagset is translated into the *pos+features* representation. Since our reference corpora (SALMA (Sawalha and Atwell, 2013) and QAC (Dukes, Atwell and Habash, 2013)) use the lemma-plus-features representation, we extract those morphological features and map the complex tag to its base tag. For example, AMIRA has a tag NNS_MD that represents a masculine dual noun. We mapped this to NN and assigned morphological features (gender, number) as appropriate (see Table 5.2 and Table 5.3). The goal of this transformation is twofold: to compare morphological features with other taggers, and to reduce the sparsity in the POS tagging. It should as well ease the mapping between the tagsets and improve the quality of the evaluation of those taggers.

5.4.5 Word and Morphological Alignment

It is apparent that we must align (morphologically and by token) the output of participating taggers before their tagging can be compared. However, what is not apparent is how this can be done, especially since we have diverse tagsets and a word is sometimes altered when segmented. In other words, there is no apparent *link* between the morphemes of different taggers.

The alignment problem here is a low-level task: given two sequences of words, the alignment task is to produce a series of links between the elements of two sequences. The result is a bipartite graph where the vertices of each partite are the words of each sequence, and the edges are the links.

Alignment should be done in multiple levels: document, paragraph, sentence, word and morpheme (or segment). The first three levels are controlled from the input to the tagger. Taggers are fed with a sentence, so the first three levels are maintained.

Alignment at the level of the word is a relatively easy job. Taggers' output is usually aligned: they rarely span a tag over two words. No single case is encountered in which two words were tagged with a single tag, as opposed to English, where "sometimes compound names or idiomatic phrases are given a single wordtag" (Atwell *et al.*, 2000, p. 11). However, some taggers drop punctuation marks from their analyses or split words without marking it as a clitic. Therefore, a word aligner module is required. It checks against the input text to align it correctly. It is a simple aligner that assumes an alignment window of three words, that is, the analysis should correspond to either the current word, the previous or the next word. It aligns the word with the one with the most similar form.

Morphological alignment is a harder problem as the link between morphemes is *not clear*. The link between morphemes can be the morpheme form or the tagging features. However, using these links is not straightforward. The morpheme form, for example, is in some tools (with *compound* tags) missing or altered. The POS tag or morphological features can be used; however, these feature labels are not standard. In Chapter 6, the problem and four experimental alignment methods are described and evaluated. See Table 5.1 for an example of the desired output.

For supervised morphological alignment (i.e. alignment of the morphemes of a single word), there is a need for training and evaluation datasets. They should have each word tagged by some taggers, i.e. "multi-tagged corpus". We developed a new multi-tagged corpus which is tagged by several taggers and manually aligned and proofread (any incorrect solution is marked) (See Sections 6.4.2 and 6.4.3).

Table 5.1 Aligned morphemes of the word ولقد walqd tagged by several taggers

MT	KH	AR	EX	MD & MA	BP	ST	SW	MR	AM
Wa	وَلَ حرف العطف	conj	C-	wa_conj	CONJ	CC	p–c–	PUNC	CC
—	حرف الابتداء	—	F-	—	—	—	p–z–	—	—
Lqd	حَرف تَحقِيق وتَقرِيب	part	F-	part_verb	FUNC_WORD	RP	p–b–	RP	RP

5.4.6 Voting and Final Prediction

The final stage is voting between aligned candidates. The voting problem is a multioutput-multiclass classification problem that aims to predict the target segmentation, POS tag, and morphological features. The given input to this stage is different according to the different configuration of the previous stages.

This problem can be modelled in different ways: voting vs. prediction, sequence labelling vs. independent-variable labelling, multi-output vs. single-output, multiview vs. stacking, or one-to-one sequence vs. sequence-to-sequence labelling.

Voting vs. prediction: Mapping the input tagset to a standard tagset is necessary for voting. Mapping should allow having a higher weight for common tags between taggers outputs. Without mapping, the problem should be named prediction not voting, and the outputs of individual taggers is considered as features (in the sense of ML).

Sequence labelling vs. independent-variable labelling: Since the individual taggers have already encoded the sequence (or the context), the problem arguably does not have to be expressed as a *sequence* classification problem. It could predict the required output given a set of features aligned at the morpheme level. However, contextual information may be used in the prediction, i.e. a sequence labelling problem.

Multi-output vs. single-output: One specific problem in morphological analysis, in general, is the prediction of correlated multiple outputs: segmentation, POS tagging, and each morphological feature. This problem can be modelled such that segmentation prediction is independent of the output of tagging. This may result in an output with mismatching number of segments and tags. The required outputs can be as well encoded into a single complex tag, but this makes the classification problem more complex due to the large size tagset.

Multiview vs. stacking: Stacking methods pile up independently trained models where each model is trained based on the predicted values of previous model. These methods suffers from the error propagation problem. Multi-view

models, in contrast, model the problem jointly by sharing common feature representations.

One-to-one sequence vs. sequence-to-sequence labelling: Since the segmentation and tagging are heterogeneous, two ways of modelling the problem exist. The first approach is the pipelined approach where the outputs of individual taggers are pre-processed to ensure they are aligned, then each morpheme is tagged. The second approach jointly aligns and tags the individual tagger's output by encoding each tagger's output and concatenating all features, then decoding the concatenated vector to the desired output (see Figure 5.3).

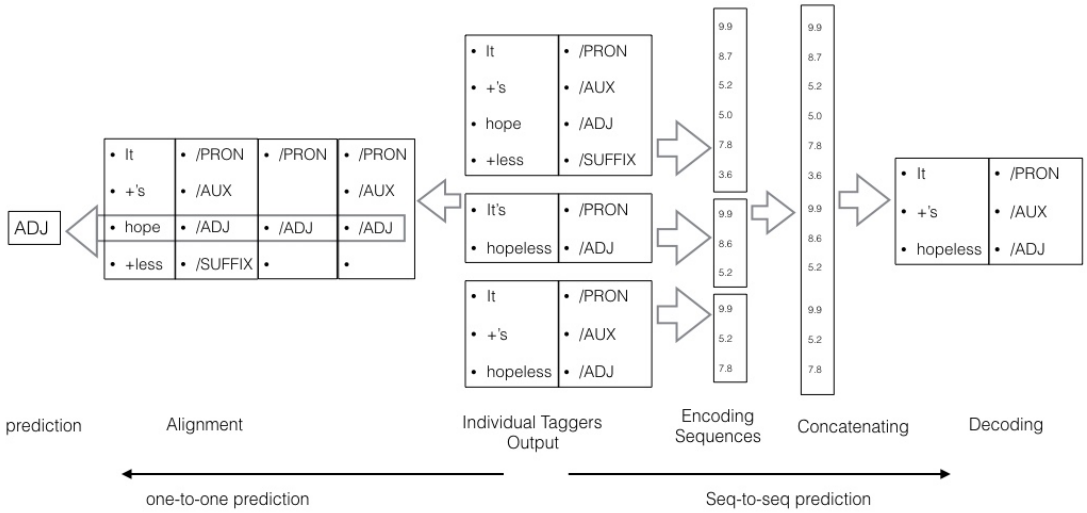


Figure 5.3 Sequence-to-sequence prediction

Since the black-box taggers are *systematically* heterogenous, there should be a systematic method that exploits the shared information between these taggers. In this thesis, we experiment with different ways for the voting stage. Unlike Alabbas and Ramsay (2014), we define the problem as a *prediction* problem (see Section 5.5 for reasons and mapping results). In Chapter 6, we experiment with a systematic *pipelined* design that defines the problem as a *prediction*, *multi-view, independent-variable*, *single-output* (only POS tag is evaluated), *one-to-one* labelling problem. This definition required a prior explicit alignment at the morpheme level which is discussed in detail. In Chapter 7, the problem is defined as an *end-to-end joint prediction, multi-view, sequence-labelling, multi-output, sequence-to-sequence* problem.

In the following sections, two attempts are discussed in detail to improve the overall robustness of the ensemble tagger. The first attempt is to map tagsets into a standard tagset, and the second is to enrich the input text with diacritics.

5.5 Experimental Study for Mapping Two Tagsets

Mapping means the conversion from one format or value described in the source tagger to the standardised or target format. Mapping can be formalised as another alignment problem where each side is a tagset and the goal is to find a link between the two sides. The link is, however, less apparent in the mapping problem, especially with different linguistic theories and segmentation schema.

The mapping stage is optional. Ensemble taggers may not require a standard tagset unless it involves some comparisons (i.e. voting). Other applications may need the mapping process such as evaluating taggers to one ground truth. Some methods in the morpheme-based pipelined ensemble approach, proposed in Chapter 6, require mapping to resolve the morphological alignment problem.

The next section defines the reference tagsets. Sections 4.5.2 and 4.5.3 define the methodology of the mapping of morphological features and core POS tags. Section 4.5.4 shows and discusses the lessons learned from an experiment of mapping one tagset to another.

5.5.1 Tagsets

Two tagsets are chosen for mapping: the SALMA tagset (Sawalha and Atwell, 2013) and the MADAMIRA tagset. SALMA is the most fine-grained tagset and is proposed to be a standard tagset in the literature. The MADAMIRA tagset is as well the most fine-grained possible POS tagset in participating taggers. Two reasons for choosing the two tagsets are as follows: they are well documented (thus, easily grasped by mapping annotators), and they are fine-grained.

The SALMA tagset is two-dimensional and is fine-grained in two aspects: its number of features (~ 15 features) and the possible tags of each word (~ 91 distinct tags). The SALMA tagset has thirty-four possible tags for nouns, one for verbs[2], twenty-two for particles, twenty for others, and twelve for punctuations. Unlike the MADAMIRA tagset, this tagset is designed to capture long-established traditional Arabic grammar, I'rab (إعراب /<ErAb/ *morphology*).

The default tagset of MADAMIRA is used which has 36 tags for tagging the base of the word. In addition, five, eighteen, seven, and two tags are dedicated to

[2] Originally three values that represents the aspect of the verb: perfect, imperfect, and imperative, but we decided to consider them as a morphological feature.

article, preposition, conjunction and questions proclitics respectively; and twenty-two tags for enclitics. The tagset used by MADA is well documented in the manual shipped with the suite.

5.5.2 Mapping Morphological Features

The mapping involves two components: Morphological features and POS tags. Morphological features are mapped to the values of the SALMA tagset. Although the naming of morphological features is heterogeneous, this mapping is straightforward and is mostly a one-to-one renaming, e.g. mapping from *gender=male* to *gender=m*. The mapping between ALL taggers and SALMA tagset is done by the author.

We made some necessary modifications to the SALMA tagset. In addition to the typical three values of the *number* feature: *singular*, *dual* and *plural*, the SALMA tagset, for example, has six more possible values (i.e. *sound plural*, *broken*, etc.). These additions are removed, and a single value for all plurals is used: "p". For the full mapping rules of morphological features, please see Table 5.2 and Table 5.3.

Table 5.2 The first part of mapping rules of morphological features from all participating taggers to the SALMA convention. The table is divided into five parts: Mood, Gender, Case, Voice and State columns. Rows in each part are trios: the tool's label, the tool acronym, the equivalent label in SALMA.

Mood	Tool	SW	Gender	Tool	SW	Case	Tool	SW	Voice	Tool	SW	State	Tool	SW
i	MA	n	masc	AR	m	2	EX	a	active	AR	a	IsDefinitiveAL	MS	d
s	MA	a	fem	AR	f	4	EX	g	pass	AR	p	-IsDefinitiveAL	MS	i
j	MA	j	Masc	MS	m	1	EX	n	a	QT	a	DET	AM	d
u	MA	-	Fem	MS	f	Nom	MS	a	p	QT	p	DT*	ST	d
na	MA	-	Masc	XE	m	Acc	MS	g	Active	XE	a	d	QT	d
I	EX	n	Fem	XE	f	Gen	MS	n	Passive	XE	p	a	QT	-
E	EX	j	_F??	AM	f	n	QT	n	VB	AM	-	i	QT	i
J	EX	j	_M??	AM	m	a	QT	a	VBD	AM	a	i	MA	i
S	EX	a	m	QT	m	g	QT	g	VBP	AM	a	d	MA	d
Ind	MT	n	f	QT	f	na	MA	-	VBN	AM	p	c	MA	-
Jus	MT	j	x	QT	-	u	MA	-	VB	ST	-	na	MA	-
Sub	MT	a	m	MA	m	a	MA	a	VBD	ST	a	u	MA	-
Eng	MT	j	f	MA	f	g	MA	g	VBP	ST	a	i	MD	i
Indicative	XE	n	na	MA	-	n	MA	n	VBN	ST	p	d	MD	d
Subjunctive	XE	a	m	MD	m	na	MD	-	a	MA	a	c	MD	-
Jussive	XE	j	f	MD	f	u	MD	-	p	MA	p	na	MD	-
Energetic	XE	j	na	MD	-	a	MD	a	na	MA	-	u	MD	-
n	QT	n	M	EX	m	g	MD	g	u	MA	-	defArt	AR	d
a	QT	a	F	EX	f	n	MD	n	a	MD	a	indef	AR	i
g	QT	j	-	EX	-	accgen	AR	-	p	MD	p	I	EX	i
i	MD	n	NSUFF_MASC	BP	m	acc	AR	a	na	MD	-	D	EX	d
s	MD	a	NSUFF_FEM	BP	f	NSUFF_*_ACCGEN	BP	-	u	MD	-	R	EX	-
j	MD	j	IV?M?	BP	m	NSUFF_*_ACC	BP	a	A	EX	a	A	EX	-
u	MD	-	IV?F?	BP	f	NSUFF_*_NOM	BP	n	P	EX	p	C	EX	d
na	MD	-	IVSUFF_SUBJ:M	BP	m	ACC	QA	a	PV	BP	a	L	EX	i
acc	AR	a	IVSUFF_SUBJ:F	BP	f	GEN	QA	g	IV	BP	a	NSUFF_*_INDEF	BP	i
_MOOD:I	BP	n	PVSUFF_SUBJ:M	BP	m	NOM	QA	n	CV	BP	a	-NSUFF_*_INDEF	BP	d
_MOOD:SJ	BP	-	PVSUFF_SUBJ:F	BP	f				PV_PASS	BP	p	INDEF	QA	i
_MOOD:S	BP	a	M	QA	m				IV_PASS	BP	p	DEF	QA	d
_MOOD:J	BP	j	F	QA	f				ACT	QA	a			
IND	QA	n							PASS	QA	p			
ENG	QA	j												
JUS	QA	j												
SUBJ	QA	a												

Table 5.3 The second part of mapping rules of morphological features from all participating taggers to the SALMA convention. The table is divided into three parts: Aspect, Person, and Number columns. Rows in each part are trios: the tool's label, the tool acronym, the equivalent label in SALMA.

Aspect	Tool	SW	Person	Tool	SW	Number	Tool	SW
pres	AR	c	1	MS	f	sg	AR	s
past	AR	p	2	MS	s	pl	AR	p
imp	AR	i	3	MS	t	dual	AR	d
imperfect	BP	c	1stPer	XE	f	Sing	MS	s
imperative	BP	i	2ndPer	XE	s	Plu	MS	p
perfect	BP	p	3rdPer	XE	t	Dual	MS	d
Pst	MS	p	_??1	AM	f	Sing	XE	s
Prs	MS	c	_??2	AM	s	Dual	XE	d
Imp	MS	i	_??3	AM	t	Plur	XE	p
Perfect	XE	p	f	QT	f	_?S?	AM	s
Imperfect	XE	c	s	QT	s	_?D?	AM	d
Imperative	XE	i	t	QT	t	_?P?	AM	p
VB	AM	i	1	MA	f	NNS	ST	p
VBD	AM	p	2	MA	s	NNS	ST	s
VBP	AM	c	3	MA	t	NNP	ST	s
VBN	AM	-	na	MA	-	NNPS	ST	p
VB	ST	i	1	MD	f	s	QT	s
VBD	ST	p	2	MD	s	d	QT	d
VBP	ST	c	3	MD	t	p	QT	p
VBN	ST	-	na	MD	-	s	MA	s
IV	BP	c	1pers	AR	f	d	MA	d
PV	BP	p	2pers	AR	s	p	MA	p
CV	BP	i	3pers	AR	t	na	MA	-
p	QT	p	1	EX	f	u	MA	-
c	QT	c	2	EX	s	s	MD	s
i	QT	i	3	EX	t	d	MD	d
i	MA	c	IV1??	BP	f	p	MD	p
c	MA	i	IV2??	BP	s	na	MD	-
p	MA	p	IV3??	BP	t	u	MD	-
na	MA	-	IVSUFF_SUBJ:1	BP	f	S	EX	s
i	MD	c	IVSUFF_SUBJ:2	BP	s	D	EX	d
c	MD	i	IVSUFF_SUBJ:3	BP	t	P	EX	p
p	MD	p	PVSUFF_SUBJ:1	BP	f	IV??S	BP	s
na	MD	-	PVSUFF_SUBJ:2	BP	s	IV??D	BP	d
VI	EX	c	PVSUFF_SUBJ:3	BP	t	IV??P	BP	p
VC	EX	i	IVSUFF_DO:1	BP	f	NSUFF_?_SG	BP	s
VP	EX	p	IVSUFF_DO:2	BP	s	NSUFF_?_DU	BP	d
IMPF	QA	c	IVSUFF_DO:3	BP	t	NSUFF_?_PL	BP	p
IMPV	QA	i	PVSUFF_DO:1	BP	f	IVSUFF_SUBJ:*S	BP	s
PERF	QA	p	PVSUFF_DO:2	BP	s	IVSUFF_SUBJ:D	BP	d
			PVSUFF_DO:3	BP	t	IVSUFF_SUBJ:*P	BP	p
			1	QA	f	PVSUFF_SUBJ:*S	BP	s
			2	QA	s	PVSUFF_SUBJ:D	BP	d
			3	QA	t	PVSUFF_SUBJ:*P	BP	p
						IVSUFF_DO:*S	BP	s
						IVSUFF_DO:D	BP	d
						IVSUFF_DO:*P	BP	p
						PVSUFF_DO:*S	BP	s
						PVSUFF_DO:D	BP	d
						PVSUFF_DO:*P	BP	p
						S	QA	s
						D	QA	d
						P	QA	p

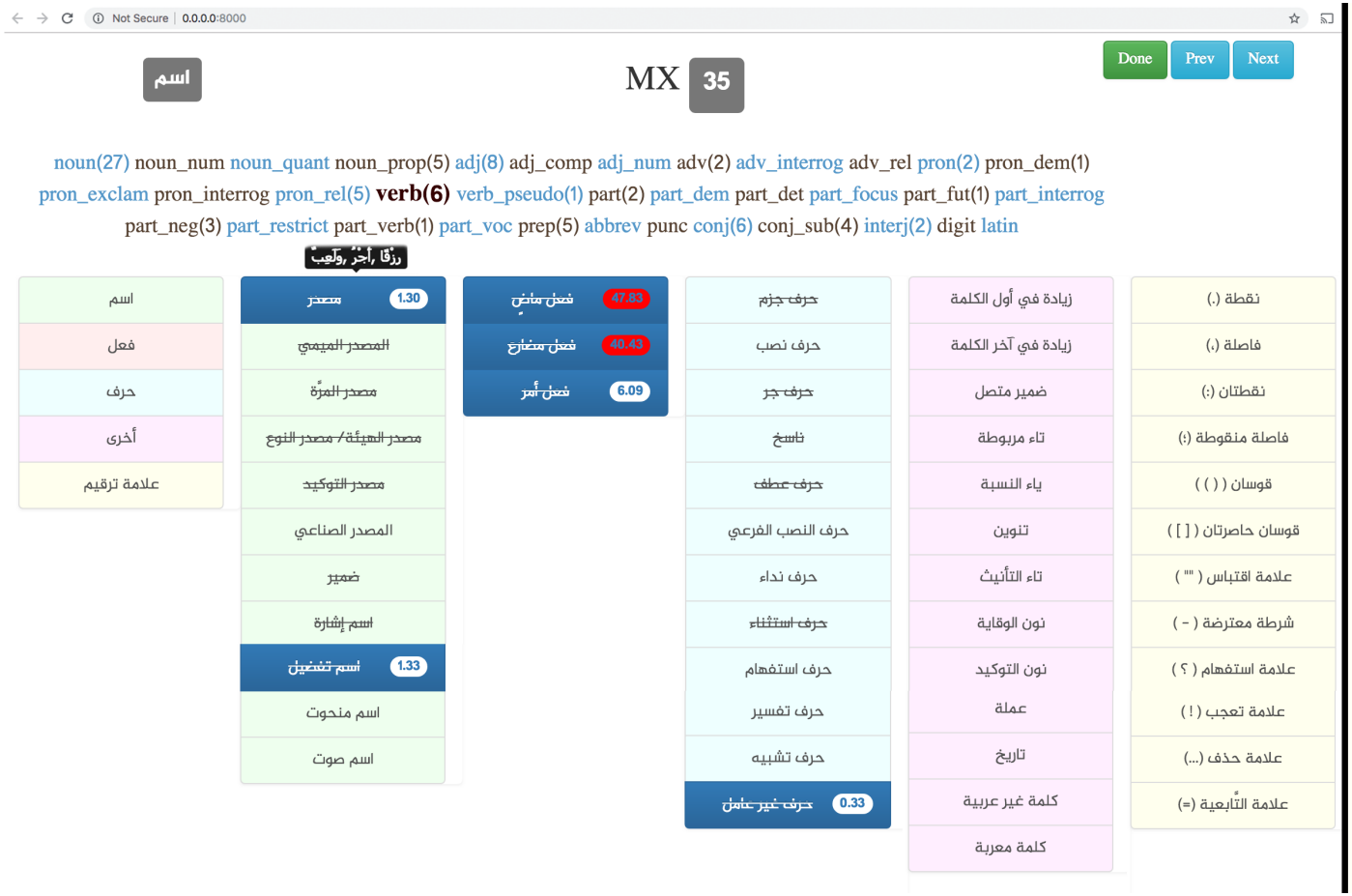
5.5.3 Mapping POS tags

The second mapping is the mapping of core POS tagsets. While many mappings in the literature involve reducing the tagset size, this experiment is designed to find all possible links between the two tagsets.

We chose not to *reduce* the tagset because it will cause a loss of information. Reducing tagset size maybe is mostly straightforward, even though it requires the understanding of both tagsets. However, when tagset size is reduced, the **full tagging** performance of the tagger will not be evaluated and exploited. Also, such mapping would force our ensemble tagger to use its reduced tagset which contradict with the stated fine-grained goal.

This mapping process can be divided into two stages: building a helper tool and manually mapping tagsets. The first stage should help the linguists in the second stage to see the tags in context. It helps as well to see how likely they co-occur in one word.

In the **first** stage, a list of co-occurrences is constructed. The MADAMIRA tagger is asked to tag the SALMA corpus to build the list. For each word, a pair of its MADAMIRA tag and its SALMA tag is defined. From this long list of tag pairs, pairs that do not past a certain threshold are deleted. Correlation statistics are computed from the rest of the list. A set of examples are maintained for every mapping pair (to help later in decisions). This list is fed into the SAWAREF mapper tool, a web-based graphical interface that eases the mapping process, where the **second** stage involves manually choosing target tags that are most appropriate. Figure 5.4 illustrates the main components of the Mapper screen layout.



Toolbar: Which shows the current source tagset: MX; its size: 35; and a description of currenly highlighted tag: (in Arabic)

Source Tagset: the tags of *source* tagset. The current tag is in bold case: verb. Numbers in parentheses refer to the number of mappings of each tag.

Target Tagset: Which shows the all possible tags of target tagset: SALMA; which is divided into five categories (illustrated in first column): nouns, verbs, particles, others, punctuations.

Tags in blue background are chosen as a possible mapping from the current tag in the source tagset (e.g. noun). Next to each tag is the possiblity of each tag derived from the training corpus. Red possibilities refer to high probable mappings.

The tooltip above the hovered tag shows some examples about the tag from the training corpus.

Figure 5.4 A screenshot of the mapper tool. The tool consists of three parts: the first part is the top bar which shows the current tagger (MA)

5.5.4 Ambiguity in Mapping Experiment

This experiment examined the possibility of mapping one tagset to another for the ensemble voting component between taggers' output.

Two volunteer linguists mapped the two tagsets. They have a background in teaching Arabic as a second language and pursuing a PhD degree in computational linguistics. Mapping one tagset to another tagset requires a thorough understanding of both tagsets. They used the 'Mapper' tool from the SAWAREF toolkit (see Figure 5.4), which was designed especially for this mapping experiment. For each tag in the SALMA tagset, the linguists were asked to select all possible tags in the SALMA tagset to map to.

They had the following in hand:

1. a description of each tag (extracted from the manual or the paper of the tagset),
2. in-context examples of the tag, and
3. some statistical correlation information about the target tag (no. of inward maps, the probability of such tagging).

Among possible mappings from the MADAMIRA tagset (59 tags) to the SALMA tagset (77 tags) (theoretically 4543 possible mappings), 228 (5%) were selected: 130 by both, 33 and 65 by each linguist. The average number of mappings for one tag in MADAMIRA is 1.88-2.57, in SALMA is 1.98-2.15 for each linguist respectively.

This experiment indicates that the mapping between the two tagsets is mostly *n-n* mapping. Although the linguistic theory of the two tagsets are different, it is surprising to see that the average number of SALMA tags from one tag in MADAMIRA range from 1.88 to 2.57. The SALMA tagset was assumed to be a much finer grained tagset. Some tags in MADAMIRA were not mapped to a single tag in SALMA. Linguists by mistake did not map some tags (e.g. date, currency, and not-separated affixes like Taa Marbouta, feminine suffix).

Because SALMA is the finer tagset, we wanted the mapping to only have one-to-one and one-to-many situations. That is, a tag can be mapped to one or many tags in the reference tagset (SALMA), but no tag on the reference tagset can originate from two tags. If a *congestion* is found on one tag (many-to-one), the reference tagset should be extended to break this congestion. For example, the two tags (from QAC tagset): *EXP* and *RES* tags (exception and restriction particles) map

to one tag in SALMA (`p---x-` exceptive particle), therefore, the SALMA tagset was extended to maintain our reference tagsets being the most fine-grained. However, this experiment showed that this methodology is not practical. The n-to-n mapping does not mean that MADAMIRA is finer than SALMA at some tags; instead, it is because they adhere to different underlying linguistic theories which prevent us from having an "extended" version of SALMA.

Our method expands the solution set of each tagger and increased ambiguity significantly. The goal was to maintain the level of granularity which could make for fairer voting between taggers. However, with this level of added ambiguity, the high variance between the two mappings, and the error rate, the mapping between tagsets might increase the error rate which will be propagated to subsequent stages. Therefore, we decided to not pursue the mapping of taggers. However, we found that these links is helpful in morphological alignment for the similarity measure of outputs, as will be shown later.

5.6 Experimental Study of Reducing Ambiguity through Diacritisation

In the Arabic language, a high amount of phonological information is missing such as *short vowels*, *Shaddah*, *tanween*, *Maddah*, and sometimes *hamzah*[1] as well (see Table 5.4 for details). They (collectively called *diacritics*) are not usually written. It is common as well in NLP to normalise them to reduce the sparseness of the data. As a result, the ambiguity at the word level is high in Arabic. There is an average of 11.5 diacritisations/word (Debili and Achour, 1998). For example, a vowelised form of the word فهم (fhm) can be one of the following "non-comprehensive" list (Figure 5.5):

1. فَهَمَ /fahama/ (v.) to understand
2. فَهَّمَ /fahhama/ (v.) to teach
3. فَهُمْ /fa+humo/ (conj. + pron.) and they
4. فَهَمَّ /faham~a/ (conj. + v.) and (he) intend
5. فَهْمَ /fihom/ (n.) understanding

[1] In cases where Hamza is considered a diacritic, only different shapes of Hamza on Alif is considered.

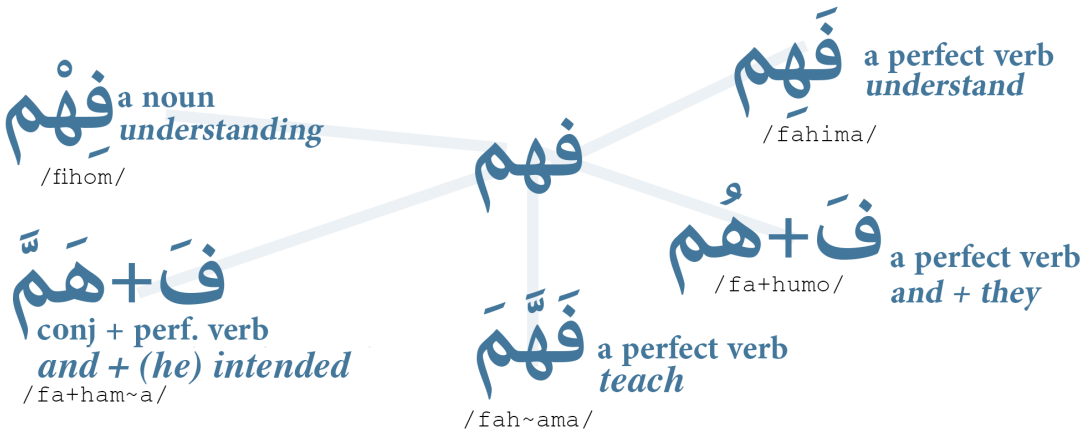


Figure 5.5 Ambiguity of one Arabic word.

Arabic *diacritisation* is the computational process of recovering missing diacritics to the orthographic word. This process is known for improving readability (e.g. children books and educational textbooks), automatic speech recognition (ASR) (Vergyri and Kirchhoff, 2004), text to speech (TTS) (Ungurean *et al.*, 2008), information retrieval (IR), and morphological annotation (Habash, Shahrour and Al-Khalil, 2016).

Words can be *fully* diacritised, where diacritics for all letter are specified, or *partially,* where diacritics for part of the letters are specified. Texts are usually fully diacritised for children's educational purposes, or when the great precision of pronunciation is required e.g. the Quran. (Hermena *et al.*, 2015). On the other hand, the text is mostly partly or completely unwritten, due to three reasons: to speed up the reading speed (Hermena *et al.*, 2015), not to strain the eyes, and to speed up the typing by one third (required for typing diacritics).

A special type is the *minimal* where some diacritics are specified in which these specifications are enough to avoid word's ambiguity. But the sufficient level of the diacritisation is ambiguous, and the minimal level depends on the audience (e.g. reader's level of education) and target; for morphological annotation in Natural Language Processing (NLP), a minimal diacritisation is the minimal partial diacritisation that is sufficient to eliminate other possible diacritisations produced by a lexicon or morphological analyser.

Table 5.4 Diacritics

Group	Diacritic	Buckwalter	Arabic	Notes
Short vowels	Fatha	/D/ /a/	ضَ	
	Dhammah	/D/ /u/	ضُ	
	Kasrah	/D/ /i/	ضِ	Optionally written for Hamzah Maksorah
No vowel	Sokun	/D/ /o/	ضْ	All letters. Indicates that the consonant is not followed by a vowel.
Shaddah (emphasis, geminate)	Shaddah	/D/ /~/	ضّ	All letters except the beginning word. Marks a long consonant. Equivalent to writing the constant twice (first is .
Tanween (Nunation)	Tanween Fatha	/D/ /F/	ضًا / ضاً	
	Tanween Dhammah	/D/ /N/	ضٌ	
	Tanween Kasrah	/D/ /K/	ضٍ	
No diacritic		/D/	ض	1. The letter preceding long vowels. 2. Long vowels 3. On the lam of the definite article. 4. When the letter is Hamzah Maksorah *Otherwise,* it indicates unspecified vowel.
Hamzah (glottal stop)	Hamzah Up	/>/	أ	Can have any short vowel. If it starts a word and has a Kasrah, Hamzah Down is used.
	Hamzah Down	/</	إ	Can only be at the beginning of has an obvious short vowel Kasrah
	Hamzah Madd	/|/	آ	Indicates glottal stop, followed by a long Alif. Cannot appear at the end of a word (its components will be written separately).
	Hamzah Wasl	/{/	ٱ	Not available in Standard Arabic keyboards. It indicates explicitly the special type of beginning Alif which is not pronounced as a glottal stop when connected to previous word. Usually written as normal Alif.

Arabic diacritisation has grabbed the attention of Arabic NLP researchers, and much work has already been done. Previous approaches have focused on improving the quality of automatic diacritisation to produce a fully diacritised version of the text, either using a *rule-based* approach (El-Imam, 2004), *statistical* approaches using, for example, recurrent networks (Abandah *et al.*, 2015), n-gram model (Hifny, 2012), *hybrid* approaches which usually perform the best (Rashwan *et al.*, 2009; Pasha *et al.*, 2014; Darwish, Mubarak and Abdelali, 2017) or using the prominent deep learning approaches (Al Sallab *et al.*, 2014; Abandah *et al.*, 2015; Rashwan *et al.*, 2015).

Diacritisation in this experimental study focuses on diacritizing text with high quality (near gold standard quality) for the purpose of manual annotation later. That is, the diacritisation approach seeks a high accuracy in diacritisation but is not necessary to diacritise the full text. Habash *et al.* (2016) exploits diacritizing to improve morphological annotation. In their work, they re-rank the solution set from the morphological analyser based on the similarity of the input diacritisation and the solution predicted diacritised form. In a similar approach, SAWAREF toolkit filters the solution set based on the input diacritised form. Additionally, the SAWAREF preprocesses the input text to standardise its diacritisation and might borrow and merge diacritisation from similar contexts. In this section, we present a robust and accurate diacritisation method of highly cited texts by automatically "borrowing" diacritisation from similar contexts.

Since the text in classical Arabic is highly cited and quoted in successive texts, we were motivated to increase its text diacritisation level, by automatically "borrowing" diacritisation from other books within the same genre.

As part of the Sawaref toolkit, we developed an open-source *diacritiser*[1] that matches the undiacritised version of one word with its equivalent in other books using their word n-gram concordance.

5.6.1 Methodology

The diacritisation of each word in the *target* corpus is done simply by searching for all locations of similar n-gram words in the *target* corpus. Then, these locations are merged to form a single diacritisation of the centric word. In the

[1] Available freely at http://github.com/aosaimy/arabic-vowelizer

extended version of this method, we asked a morphological analyser if it can help. Finally, we replaced the word with the new diacritised word.

Algorithm 1 describes the method formally:

1. $G^{train} = nWGram(x, n)$ `for` $x \in W_{train}$

   The first step is to convert the training text W_{train} into a list of word n-grams, with reference to its locations in the text, diacritised and undiacritised versions of the centre word. Documents are read in the training corpora in parallel to speed up the development of the lexicon data.

2. $M\left(w_i^{target}\right) \subset G^{train}$ `where` $g_k^{train} = nWGram\left(w_i^{target}, n\right)$

   For each n-gram g_k^{train} that is on our list (after normalisation), it builds a list of matching word-ngrams $M\left(w_i^{target}\right)$ from the training corpus where each element g_k^{train} has the same n-gram $nWGram\left(w_i^{target}, n\right)$.

3. $D_i = \{\dots v(x) \dots\}, \forall \; nWGram(x, n) \in M\left(w_i^{target}\right)$

   For matching n-grams $M\left(w_i^{target}\right)$, it extracts all found diacritisations of the centre word $v(x)$ and counts the number of occurrences of that diacritisation.

4. $D_i = sort(D_i)$

   Once finished, variants are sorted by the number of occurrences in descending order. The goal of this sorting is to prevent infrequent diacritisation from bubbling up to the surface diacritisation in the next step.

5. `while` $i < |D_i|$ `; do`

   $$v(w_i) = merge\left(v(w_i), v(d_i)\right)$$

   `od`

   Centre words diacritisation variants $v(d_i)$ are merged recursively: the merge procedure (Algorithm 2) is done letter by letter. For every letter, only candidate diacritics that do not contradict with one existing are merged.

6. $v(w_i) = MA_0(w_i)\; iff\; |MA(w_i)| = 1$

This step only applicable to *extended* version which uses the morphological analyser (MA) to improve the results if possible. Merged centre words are replaced by a more thorough diacritisation $MA_0(w_i)$ (if it exists) by consulting a morphological analyser if and only if it matches one candidate diacritisation $|MA(w_i)| = 1$.

7. The centre word's locations in the text are replaced with the new diacritised version.

This methodology assumes the following:

1. The diacritisation of the source corpora is done manually, i.e. not artificially,
2. Diacritisation of both target and source is standard,
3. Word diacritisation is only based on a window of *n*,
4. The target text is quoted or reused in the source corpora, and
5. There is no other diacritised form if morphological analyser says so (only applicable in the extended version)

As stated before, the goal is to fully diacritise words in a classical Arabic text to increase the robustness of the morphological annotation of the corpus. In the next subsections, we show how these assumptions are valid for our case.

Algorithm 1. BorrowBasedDiacritise

DEFINE:

$W = \{w_1, w_2, ..\}$ is a series of words w.

$l(w)$ is a series of letters l_i of word w.

$v(w) = \{v_1, v_2, ..\}$ where v_i is a series of diacritics of letter l_i and $|v(w)| = |l(w)|$.

$nWGram(w_i, n) = \{w_{i-n}, w_{i-n+1}, \dots, w_i, \dots, w_{i+n-1}, w_{i+n}\}$

$MA(w)$ is a series of $v(w)$ from a morphological analyser.

INPUT: W_{train}, W_{target}, n

OUTPUT: $v`(w)$ for all $w \in W_{target}$ such that $|v_i| \leq |v`_i|$ for all i.

1. $G^{train} = nWGram(x, n)$ for $x \in W_{train}$
2. $M\left(w_i^{target}\right) \subset G^{train}$ where $g_k^{train} = nWGram\left(w_i^{target}, n\right)$
3. $D_i = \{\dots v(x) \dots\}, \forall\ nWGram(x, n) \in M(w_i)$
4. $D_i = sort(D_i)$
5. while $i < |D_i|$; do
   $v(w_i) = merge(v(w_i), v(d\ ))$
   od
6. $v(w_i) = MA_0(w_i)\ iff\ |MA(w_i)| = 1$

Algorithm 2. Merge

INPUT: $v(w_1), v(w_2)$ where $l(w_1) = l(w_2)$

OUTPUT: $v`(w_1)$ such that $\sum|v_i| \leq \sum|v`_i|$.

$v_i(w_1) := v_i(w_2)$ iff $v_i(w_1) \leq v_i(w_2)$

end;

5.6.2 Assumption #1: Non-Artificial Diacritics in Source Corpora

For the first assumption, no sign of automatic diacritisation could be found in the Shamela Library. Moreover, some diacritised corpora like (Zerrouki and Balla, 2017) used some of its books as a source for verified diacritisation.

5.6.3 Assumption #2: Diacritics Standardisation

To enforce the same standard in source and target, we perform diacritisation normalisation as illustrated in Table 5.5. The terminology in the second column is in ‘regex' notation. Regex is a search pattern that is translated later by a regular expression engine into a non-deterministic finite automaton. We use the notion of regular expressions, as it is commonly used and quite efficient for text substitutions. For example, Fatha Tanween (Rule number 5) should always be before Alif and Alif Maqsorah, so the regular expression search for `AF` followed by a space `\s` and replace it with `FA` instead. The `(?=)` symbol makes sure that spaces are not captured, so it is not substituted.

Table 5.5 Normalisation of diacritisation Rules

Rule	Find Pattern	Replace With	Example	
			From	To
1.Remove starting diacritics	/(?=\s)([aiuoFKN~]+)/g	None	/**a**mkAnA/	/mkAnA/
2.Remove space-surrounded diacritics	/ [aiuoFKN~]*(?=\s)/g	None	/**a**/	//
3.Add Sokun diacritic on the long vowel Alif	/aA/g	aAo	/mkAnA/	/mkA**o**nA/
4.Remove duplicates of the same diacritic	/([aiuoFKN~]){2,}/g	$1	/m**aa**kAnA/	/m**a**kAnA/
5.Tanween then end	/AF(?=\s)/g	FA	/mkAn**AF**/	/mkAn**FA**/
	/YF(?=\s)/g	FY		
6.Shaddah should always be before other diacritics	/([aiuFKN])~/g	~$1	/vm**a~**/	/vm**~a**/
7.Remove incompatible diacritics	/([aiuFKN])[aiuFKN]+/g	$1	/v**Na**m/	/vm/
8.Tanween not at the end of word	/[FKN]([^][^])/g	$1	/mk**F**AnA/	/mkAnA/
9.Shaddah at the beginning of a word	/ ~/g	None	/~mkAnA/	/mkAnA/
10.Bottom Hamza	/<[auFKN]/g	<i	/**<**srA'/	/**<i**srA'/
11. Bottom Hamza not the beginning	/<([^][^])/	>i	/AlxT**<**/	/AlxT**>i**/

5.6.4 Assumption #3: Word diacritisation is the same for *n* surrounding words

Changing one final diacritic from a full sentence might change its meaning completely (Azmi and Almajed, 2015). While this contradicts our assumption, we examine the quantity of these cases in the full corpus.

To validate prior assumptions (mainly the last), we extracted word five-grams that have variant diacritisation of its centre word. Then, we examined the top of the list (top 100), ranked based on the number of variants in descending order. Table 5.6 lists a sample of top 5-grams.

All variants did not show a sign of artificial diacritic, nor show a non-standard diacritisation. The centre word has no conflicting diacritisation for 98% of the top 100 on the list. Conflicting diacritisation is due to different pronunciation of proper nouns, misspelt diacritics, or improper last diacritic.

Table 5.6 The possibility to merge diacritisations of variants forms.

Word	Possibilities	Context	Can be Merged?
*r	*r, *r~, *rK, *arK, *ar	wEn >aby *r rDy Allh	Y
lnby	Alnaby~, Alnby, Alnbyi, Alnby~	wEnh En Alnaby~ SlY Allh	Y
w>n	w>na, wa>na, wa>n~	<lAa Allh w>na muHmadFA rswlu	Y
wrhbp	warahbapF, warahobapF, warhobapF	<layoka ragbapF warahbapF <layoka lA	Y

5.6.5 Assumption #4: The similarity between the source and target corpora

The reliability of the optional diacritisation step depends widely on the availability of another similar context. As such, this assumption highly depends on the text to be analysed. However, classical texts, especially the Quran and the Sunnah, are quoted more often than modern texts.

In the case study of the Sunnah Arabic Corpus, SAC is mostly a collection of religious text which is widely quoted. Several authors have explained its narrations, which increases the chance that its text has been quoted. The results of our experiment show that at least 84.34% of the corpus word n-grams has been found in the source corpora.

5.6.6 Assumption #5: The morphological analyser covers all diacritised forms

Using the SAWAREF toolkit (Alosaimy and Atwell, 2016), we run four morphological analysers, namely Elixir Functional Morphology (EX) (Smrz, 2007), ALMORGEANA (included in MADA toolkit) (AL) (Habash, Rambow and Roth, 2009), AraMorph (BP) (Buckwalter, 2002a), and AlKhalil (KH) (Boudchiche *et al.*, 2016), on the lexicon of Riyadh Asslaheen (17600 distinct words). The average number of possible diacritised forms is shown in Table 5.7.

We used four morphological analysers to increase the diacritisation coverage for our corpus. By merging the output of analysing each word, we built a list of possible diacritisation of each word. After close examination of the results, their level of diacritisation is different. The diacritised format is not usually full. Table 5.7 showed the diacritisation coverage for each analyser. While merging analysers' results should increase the word coverage, similar words do not merge together as taggers' diacritisation is homogeneous. As a result, we have more than one form of diacritisation when in fact there should only be one. This explains the jump in the number of possible diacritisation from 10.38 (at maximum) to 17.42.

Table 5.7 Possible Diacritisation Statistics Per Morphological Analyser.

MA	Max	Mean	Median	Coverage
EX	124	8.46	6	67.46%
KH	96	10.38	7	80.64%
BP	20	2.38	2	47.67%
AL	23	3.69	3	42.65%

We only use MA diacritisation if it matches only one form. Using a random sample (of 100 enhanced words), we could not spot a single error in the enhanced diacritisation. It suggests that it is safe to assume there is no other diacritised form if the morphological analyser says so.

5.6.7 Evaluation

Our evaluation uses two metrics: accuracy, and coverage, both in terms of character level. Accuracy is measured by Diacritic Error Rate (DER), i.e. the fraction of letters that do not have the same diacritics in the original text. Coverage measures the fraction of letters that has at least one diacritic.

$$DER = \frac{\text{no. incorrect diacritcs}}{\text{no. diacritics}}$$

$$Coverage = \frac{\text{no. diacritised letters}}{\text{no. letters}}$$

In addition, we introduce an ambiguity measure defined as the practical average of the possible number of diacritisations per word. In theory, if a word of three letters has no diacritics, there are at least eight possible diacritisation for each letter (final letter can have more). However, we report the practical number of diacritisations only, extracted from a lexicon (or in our case, morphological analysers). In case a partially diacritised word is provided, the morphological analyser will only return the subset of possible diacritisations of that word with respect to the given diacritisation. If the word is not in the lexicon, we exclude that word from the average.

$$ambiguity(w) = \text{no. analyses returned by MA}$$

$$ambiguity = \frac{\sum_w ambiguity(w)}{no.words}$$

We test on the part of the text that is already diacritised. In other words, we used our models to diacritise a completely undiacritised version of Riyad, and later test the accuracy and coverage of our assumption on the diacritised version. However, since this method does not diacritise the full text, we only evaluate based on the subset of letters that has a diacritic. We do not consider Hamza nor Maddah as a diacritic, because in classical Arabic they are usually written according to the standards. Hamza in Modern Standard Arabic is misspelt or omitted in many cases. Similarly, Maddah is omitted in some frequent words. We only count short vowels including Shaddah and Tanween.

Table 5.8 reports the coverage, diacritic error rate, and average word diacritisation ambiguity of baseline, three n-gram models (3,5,7-grams) with/without help from morphological analyser. The baseline is the original form of the text.

We can see that accuracy improves when the word's context is broader, but on the other hand, the coverage drops. Word ambiguity does not change after using MAs, as MAs' diacritisation is not used unless word diacritisation only matches one candidate. The accuracy increased very slightly (about 0.0001) when using MAs; however, the coverage increased by ~0.2%.

Table 5.8 Evaluation of N-gram Diacritisation Models.

Model	Coverage	DER	Ambiguity
Undiacritised	0	N/A	17.42
Baseline	48.66%	N/A	4.83
3-gram	80.32%	0.007	1.56
3-gram+MA	81.26%	0.007	1.56
5-gram	76.41%	0.004	1.91
5-gram+MA	77.70%	0.004	1.91
7-gram	73.97%	0.003	2.13
7-gram+MA	75.59%	0.003	2.13

In terms of word-level, the source of Riyad is about 47.1% fully diacritised, and after borrowing diacritisation, the percentage jumps to 87.1%. However, this measure is not precise in our case, because of the different definition of the *fully-diacritised word*.

Additionally, we compare our results to two major open access diacritisers: MADAMIRA (Pasha *et al.*, 2014) and FARASA (Darwish, Mubarak and Abdelali, 2017). Diacritisation is normalised for both toolkits. Our 5-gram model slightly surpasses both tools, and FARASA scored an error rate of 0.006 while MADAMIRA was not performing well–0.214, which is because MADAMIRA removes original diacritics before processing the text. For a fair comparison, we re-compute the error rate given the undiacritised version. The FARASA error rate jumped to 0.263, and the DER of our 5-gram model increased slightly to 0.008.

While the two tools are expected to diacritise the text thoroughly, we found that MADAMIRA only diacritised 61.73% of letters, and FARASA only diacritised 65.36%, and 67.68% for undiacritised, and diacritised input text respectively. Using our method, the 5-gram model diacritised 71.81% of letters, due to diacritisation standards of final letter, article AL and long vowels in addition to the fact that our measure does not tolerate letters with obvious diacritics (such as Alif Madd (آ), Alif (إ) and Lower Hamza (ٕ)). Even the Quran text (extracted from Tanzil Project), which is known to have a full diacritised form, covers only 77.83% of letters. Table 5.9 summarises these findings.

Table 5.9 Comparison with major off-the-shelf diacritisers.

Tool	Coverage	DER	Input Text
MADAMIRA	N/A	N/A	Diacritised
	61.73%	0.214	Undiacritied
FARASA	67.68%	0.006	Diacritised
	65.36%	0.263	Undiacritied
5-gram	76.41%	0.004	Diacritised
	71.81%	0.008	Undiacritied

Interestingly, using Riyad itself as the only source for diacritisation, we found different diacritisation of the same n-grams. 2330 word 5-grams has different diacritisation of its centre word. The diacritisation coverage increased from 48.66% to 58.48% using the same text as a source for diacritisation.

5.7 Experimental study: Tagging Adjectives

While the ensemble of morpho-syntactically taggers aims to provide a robust way of tagging text, it is useful for some other purposes: e.g. linguistic comparaison of input taggers. This study aims to highlight the differences in the underlying theories of tagset.

Adjectives are commonly mistagged as nouns. The cause of this confusion is the definition of adjectives in Arabic. In traditional Arabic grammar, an adjective is marked when it qualifies its preceding corresponding noun, i.e. attributive adjective. In this case, attributive adjectives agree with the definiteness, number, case and gender of their corresponding noun. For example, (رجل طويل */rjl Twyl/ a tall man*). Taggers agree mostly on tagging "tall" as an adjective. However, taggers often vary in tagging "tall" in predicative adjectives: (هذا الرجل طويل */h*A Alrjl Twyl/ This man is tall*).

Table 5.10 The agreement of tagging adjective morphemes between two manually annotated corpora. Recall = 0.38, Precision=0.85.

		SALMA	
		nj----	**Others**
QAC	**ADJ**	11	2
	N	18	N/A

Using the parallel annotated corpus (PAC) (See Section 6.3.5), we evaluate and analyse each tagger and the two corpora in the sense of tagging adjectives. Surprisingly, the two manually annotated corpora were not in agreement in tagging adjectives. Table 5.10 shows the confusion matrix of tagging adjectives. In only 11 out of 31 cases, the two manually annotated corpora agree on the tagging. In the other 18 cases, QAC tagged them as NOUN. One reason behind this low recall and precision is the *incompatibility* of tagsets: QAC's definition of adjectives is "syntax"-driven while SALMA is morphologically driven.

Table 5.11 One sentence shows how linguists do not agree on tagging predicative adjectives.

Word	Transliteration	QAC	SALMA	Translation
إِنَّهُ	/<n~ahu/	<n~a/ACC+hu/PRON	<n~a/pa+hu/rr	Indeed,
هُو	/huw/	huw/PRON	huw/np	He is
العزيز	/AlEzyz/	Al/DEF+Ezyz/N	Al/rd+Ezyz/nj	The Exalter in Might
الحكيم	/AlHkym/	Al/DEF+Hkym/ADJ	Al/rd+Hkym/nj	The Wise

Table 5.11 illustrates the difference in tagging adjectives of one verse (29:26). QAC tagged the word *alaziz* as a *noun* as it is acting as a predicate (called *khabar* in Arabic traditional grammar). SALMA tagged it however as an *adjective*. However, QAC is not always consistent in this matter; verse 29:19 says: { إن ذلك على الله يسير} "that for Allah is easy/ADJ" is not consistent with its following verse: { إن الله على كل شيء قدير} "Indeed Allah, over all things, is competent/N". The words: "easy/ADJ" and "competent/N" are both adjectives acting as predicate (*khabar*) and should be treated similarly.

The same confusion carried over to SAWAREF participant taggers: when QAC is the gold standard, the average f-score is 0.11 (precision=0.14, recall= 0.2). With SALMA, the average f-score is 0.12 (precision=0.22, recall= 0.14). These very low scores show how hard is the problem of adjective tagging. The full precision and recall of each tagger is reported in Table 5.12. We used QAC's tag: *ADJ* and SALMA's tag= *nj*---- as the only tags of adjectives.

As a conclusion, even though adjectives play an important role in the semantic level, they need a more robust definition and warrant more investigation on how to predict them in Arabic specifically.

Table 5.12 The precision, recall and f-score of predicting adjectives in chapter twenty-nine of the holy Quran.

Tool	QA as Gold Standard			SW as Gold Standard		
	Precision	Recall	f-score	Precision	Recall	f-score
MT	0.11	0.62	0.19	0.16	0.41	0.24
KH	1	0.08	0.14	1	0.03	0.07
AR	0.07	0.15	0.1	0.07	0.07	0.07
EX	0.04	0.23	0.07	0.05	0.14	0.08
MD	0.03	0.08	0.05	0.13	0.14	0.14
MX	0.12	0.23	0.16	0.24	0.21	0.22
AL	0.07	0.15	0.1	0.07	0.07	0.07
BP	0.05	0.15	0.08	0.08	0.1	0.09
BJ	0.11	0.23	0.15	0.14	0.14	0.14
ST	0.18	0.46	0.26	0.29	0.34	0.32
WP	0	0	0	0.03	0.03	0.03
AM	0.14	0.31	0.2	0.21	0.21	0.21
QA	N/A	N/A	N/A	0.85	0.38	0.52
SW	0.38	0.85	0.52	N/A	N/A	N/A

5.8 Conclusion

This chapter defined the set of problems and subproblems in this thesis including segmentation and tagging. It listed the challenges that face the development of such an ensemble tagger. Then, it identified the critical parts of the SAWAREF system and showed the stages of the ensemble POS tagger process. It briefly showed the methodology for overcoming obstacles in the ensemble method, namely morphological alignment, diversity in tagset.

In an experiment of mapping one tagset to another, results showed a high error rate and disagreement between annotators, which suggests that tagsets should be used without mapping. Careful borrowing of diacritics in similar context shows an excellent opportunity to reduce the word ambiguity level. The open-source SAWAREF toolkit runs multiple taggers, standardises their results, and aligns the result of each analysis. An expected issue is low agreement among Arabic linguists on the definitions of grammatical categories, as exemplified by the tagging of adjectives.

6 PIPELINED ENSEMBLE TAGGER

Chapter Summary[1]:
An ensemble of black-box taggers requires that they conform to a standard segmentation schema. Because of the absence of this standard, a systematic alignment method should be applied. Our pipelined ensemble combined four heterogeneous POS-taggers and evaluated on a classical Arabic corpus. Two models of the ensemble tagger are presented: *morpheme-based* ensemble, and *form-based* ensemble.
In the **first part**, we opt to align tagger output using *tagger labels*. Four methods of alignment between segments using individual tagger's POS tags are presented and compared. The problem is not trivial as it is tackling five different tokenisation and labelling standards (the tagsets of four input taggers and the target tagset). The supervised learning using a unigram model scored the best segment alignment accuracy, correctly aligning 96.75% of morpheme segments. Using the best approach to align input POS taggers, the ensemble tagger has correctly segmented and tagged 88.09% of morphemes.
In the **second part**, we opt to align tagger output using *word forms* in a character-based setup. Unlike the first ensemble, this ensemble allows a parallel prediction of segmentation and labelling problems as it goes deeper and does not rely on the tagger's segmentation. This ensemble scores a slightly better accuracy: 88.73%.
We show that increasing the number of individual taggers raises the accuracy, suggesting that input taggers make different errors.

---

[1] Some parts of this chapter are based on:
Alosaimy, A. and Atwell, E. (2017) 'Joint Alignment of Segmentation and Labelling for Arabic Morphosyntactic Taggers', *International Journal of Computational Linguistics*. CSC Journals.
Alosaimy, A. and Atwell, E. (2017) 'Ensemble Joint Segmentation and POS Tagger for Arabic" in *The Workshop on Computational Approaches to Morphologically Rich Languages CAMRL.* Leeds, UK. (presentation).

6.1 Introduction

There is a need for a Part-of-Speech (POS) tagger for under-resourced classical Arabic, the language of the Quran and other Arabic texts from the 7th to 9th centuries CE. Using gold standard samples from the Quran and the Sunnah and several morphological taggers, the goal is to adapt these tools to analyse non-Quranic classical Arabic texts, including the Sunnah. This chapter shows different models for the **pipelined** approach of combining existing POS-taggers for Modern Standard Arabic (MSA), adapted to input classical Arabic words and texts, and to output classical Arabic POS-tags.

The adaptation used some ensemble methods, which have proven to be more effective than an individual algorithm in many cases. Because input (or *individual*) POS-taggers are heterogeneous, methods for alignment of segmentation and labelling in parallel is a necessity. POS-taggers have been developed for Modern Arabic, but they do not conform to shared standards in morphological segmentation or morphosyntactic tagsets for labelling.

The alignment between taggers serves another goal: an evaluation of taggers. When evaluating an automatic part-of-speech (POS) tagging, the segmentation scheme of words of the gold standard (the sequence of morphemes) should match the segmentation scheme of the tagger (Paroubek, 2007). For example, if the gold-standard corpus strips the suffix in *he's*, a tagger should strip it too.

Sequence alignment is a well-known problem in several computational fields. It is the process of identifying tokens that correspond in some manner in the source and the target sequences. Bitext word alignment in Machine Translation is an example that identifies translation relationships between words to limit or constrain the set of translation rules learned from a bilingual parallel corpus. The problem in this chapter is very similar. Unlike bitexts alignment, which aims for linking related words in terms of *meaning*, our aim is to find a *link* between elements (segments) of one sequence (the list of word's segments) to another in terms of *morphological analysis*. The link, however, is not clear and can be defined in several ways.

After briefly formalising the problem, this chapter presents two approaches of alignment: morpheme-based and form-based (or character-based). The

morpheme-based approach links morphemes using their *labels*, while form-based links them using their *characters.*

6.2 Problem Definition

The goal of the ensemble tagger is to use output from taggers to predict the correct class. However, since these taggers conform to different tokenisation schema, morphemes are not appropriately aligned. For example, as one tagger A split off *DEF* article and another tagger B does not, from the word */Alkitab/,* the following incorrect input will be fed to the ensemble classifier:

Word	**Features**	**Class**
Al	DEF_A N_B	Def
Kitab	N_A	N

6.2.1 Morpheme-based Alignment

The alignment problem can be formally defined as the following: having two sequences of tagged words $A = \{a_1, a_2, \dots\}$ and $B = \{b_1, b_2, \dots\}$ where a_i is a vector that represents a word in a sentence and $\forall\ a \in A;\ M_a = \{m_1, m_2, \dots\}$ is the sequence of morphemes in that word, the problem is to find $m \rightarrow n, m \in M_a, n \in M_b, a \rightarrow b$. The result of the mapping is a set of pairs: $C = \{(m_i, n_j), \dots\}, m_i \in M_a, n_j \in M_b$. Indices in pairs appear just once, limiting pairs to 1-1 mappings. In other words, the result of the alignment is a bipartite graph G=(V, E) where each edge $e = (m, n)$ and each vertex is a leaf vertex.

The Needleman–Wunsch algorithm (see Section 6.2.3) is used to compute the optimal global alignment between two sequences of tags using a variety of scoring matrices.

Two sequences of morphemes are regionally-aligned: words (delimited by a space) are aligned to their corresponding words on the other sequence, i.e. There is already an existing alignment mapping of $a \rightarrow b, a \in A, b \in B$. Therefore, a link in the alignment cannot pass word's boundaries. Such existing constraint should raise the baseline accuracy as the number of possible mappings is limited.

To illustrate the problem, the word (ولقد, */walaqado/*, and indeed) has two possible tokenisations shown in Figure 6.1. Two gold-standard corpora segmented the word into three segments, and four taggers segmented it into two segments. This tokenisation problem can vary from tagging compound names (with one tag) to

tagging a single word with multiple segments. Therefore, it is necessary to align the results of those taggers for proper evaluation and voting.

Figure 6.1 A sample of morpheme-aligned POS tags of one word that has two/three morphemes.

Segment Form	MA	ST	AM	FA	SAL	QAC
w+	conj	CC	CC	CONJ	p--c--	CONJ
la+	part_verb	RP	RP	PART	p--z--	EMPH
qado					p--b--	CERT

```
MA:  wa+/conj laqad/part_verb
ST: w+/CC lqd/RP
AM: w+/CC lqd/RP
FA: w+/CONJ lqd/PART
SAL: wa+/p--c-- la+/p--b-- qado/p--b--
QAC: wa+/CONJ la+/EMPH qado/CERT
```

In the first part of this chapter, three morpheme-based methods are compared (in addition to a simple baseline approach). Four taggers' output is aligned to two gold-standard corpora using:

1. **Rule-based:** Manually mapping tagsets from each one to the others, then aligning matched tags;
2. **Unsupervised:** Learning the alignment based on the possibility that two tags appear in the same word;
3. **Supervised:** Predicting the alignment using a parallel corpus of manually aligned tags; or
4. **Baseline:** Aligning the *core* or primary morpheme of the word, then aligning affixes starting from the closest ones to the primary morpheme one-by-one.

6.2.2 Form-based Alignment

The second part of this chapter follows a different method of alignment. It uses the morpheme's form for linking. Because segmentation form overlap between input, this instead goes to a deeper level: word’s characters (the set of its letters).

The deeper level allows the ensemble to overcome the one-to-one prediction limitation of the previous approach. The problem becomes a sequence problem

where the goal is to predict the label of each *character*. This model of the problem makes the output segmentation free from input segmentation schemas.

A character-based model jointly segments and tags the text using Inside-Outside-Beginning (IOB) format (Kudo and Matsumoto, 2001). The joint approach was successfully applied to Arabic (Diab, Hacioglu and Jurafsky, 2004; Kübler and Mohamed, 2012; Abdul-Mageed, Diab and Kübler, 2013; Algahtani and McNaught, 2015). However, previous work does not reuse other taggers for language adaptation.

In the character-based approach, the ensemble classifier is trained on the character-level instead of morpheme-level. IOB format encodes the character position in the sequence. Each character *c* is tagged with its POS tag *prefixed* by a character to indicate *character position*. Spaces between words are labeled as O. For example, a tagged sentence: "`He/PRON play/V +s/CASE`" will be encoded as:
`H/B-PRON e/I-PRON <SPACE>/O p/B-V l/I-V a/I-V y/I-V <SPACE>/O s/B-CASE`

The input to our system is a sequence of words: $W = [w_0, w_1, w_2, \dots]$ which is split into characters $C = [c_0, c_1, c_2, ..] \; where \; c_i \in \mathcal{A}$, , where $\mathcal{A}$ is the alphabet. The goal is to predict its IOB-augmented tag. Then the predicted tag is decoded into morpheme-based tagged text.

Segment Form	MA	ST	AM	FA	SAL	QAC
w	conj	CC	CC	CONJ	p--c--	CONJ
l	part_verb	RP	RP	PART	p--z--	EMPH
a	part_verb	RP	RP	PART	p--z--	EMPH
q	part_verb	RP	RP	PART	p--b--	CERT
a	part_verb	RP	RP	PART	p--b--	CERT
d	part_verb	RP	RP	PART	p--b--	CERT
o	part_verb	RP	RP	PART	p--b--	CERT

```
MA:  wa+/conj laqad/part_verb
ST: w+/CC lqd/RP
AM: w+/CC lqd/RP
FA: w+/CONJ lqd/PART
SAL: wa+/p--c-- la+/p--b-- qado/p--b--
QAC: wa+/CONJ la+/EMPH qado/CERT
```

Figure 6.2 A sample of character-aligned POS tags of one word that has two/three morphemes.

6.2.3 Needleman–Wunsch Algorithm

For the morpheme-based alignment approaches, the Needleman–Wunsch algorithm (Needleman and Wunsch, 1970) is used to compute the optimal global alignment between two sequences of tags.

The Needleman–Wunsch algorithm is a dynamic programming algorithm that maximises a score computed by summing the weights of matches and penalising for each gap inserted. The alignment depends on:

1. the penalty associated with an insertion of a gap, and
2. the weights associated with a match.

The Needleman–Wunsch alignment is projective, i.e. there are no two mappings such that:

1. $m_i \rightarrow m_j$ where $m_i \in M_a$ and $m_j \in M_b$ and $i < j$, and
2. $m_i \rightarrow m_j$ where $m_i \in M_a$ and $m_j \in M_b$ and $j < i$.

This property is helpful as taggers produce tags in the same order.

The scoring system was adapted to the problem. The score does not only count the cost of operation but also the two tokens involved in alignment. Using the similarity matrix $S^{A,B}$, the cost of one operation depends on the distance between A and B. Matching between `noun` and `N` may be given a full score, but matching between `noun` and `proper_noun` may be given a lower score. $S^{A,B} = (0 \leq m_{i,j} \leq 1)$

This algorithm is only used in two of morpheme-based methods (supervised and rule-based). The similarity matrix $S^{A,B}$ of the rule-based approach is based on the hand-crafted mapping rules. The aligned corpus (PAC) is used to infer the similarity matrix using unigrams and bigrams methods in the data-driven approach.

6.3 Data and Tools

This section briefly describes the input taggers and the reference corpora used throughout the thesis experiments of ensembles. It also contrasts the tagsets and segmentation schemas. It introduces the parallel-aligned corpus used to derive the required mappings for rule-based and supervised alignments. It also describes the orthographic adaptation of one specific corpus for character-based alignment.

6.3.1 Taggers

The ensemble used different combinations of four POS-taggers designed primarily for modern standard Arabic. Namely, they are:

1. MADAMIRA (MX) (Pasha *et al.*, 2014),
2. Stanford Tagger (ST) (Toutanova *et al.*, 2003; Monroe, Green and Manning, 2014),
3. AMIRA (AM) (Diab, 2009), and
4. Farasa (FA) (Zhang *et al.*, 2015).

They are chosen for the high reputation in the research community. They are the best four in our experiment of tagging classical Arabic (see Table 4.6). All of them are deterministic: they provide one analysis per word or at least rank its analyses. Non-deterministic taggers were excluded as it is beyond the scope of this thesis.

The taggers use statistical methods, and they relied on the Penn Arabic Treebank for training their model. MX is different in a sense it is a disambiguation tool and relies on a morphological analyser. The predicted analysis is used to rank morphological analysers outputs.

Each tagger treats the segmentation and tagging differently. MADAMIRA does not segment the raw text in advance. Instead, it formulates the problem as a word-based multioutput-multiclass classification problem, where four of the classes are for proclitics and one for enclitics. The Stanford tagger uses a pipelined structure where the segmentation results are piped to the tagger. The AMIRA tagger uses a character-level joint segmentation and tagging, where each character is labelled with the POS tag with a reference to its position of the segment. Farasa uses another pipelined structure with a different learning method and lexicons. Table 6.1 shows the supported output classes for each tagger.

Table 6.1 Features of Participating POS taggers.

Name	AM	MX	ST	FA
Base POS tag	Yes	Yes	Yes	Yes
Aspect	Yes	Yes	Yes[2]	-
Person	Yes	Yes	-	-

[2] Unless it is passive.

Gender	Yes	Yes	-	Yes[3]
Number	Yes	Yes	Yes[4]	Yes[2]
Voice	Yes	Yes	Yes	-
State	-	Yes	-	-
Mood	-	Yes	-	-
Case	-	Yes	-	-

6.3.2 Training and Testing Data

A subset of the QAC chapters is used in addition to the manually annotated part of SAC. The chapters chosen are namely: 2, 3, 4, 10, 15, 30, 45, 60, 75, 90 and 105. The total number of words is 17.8k with 5.7k and 5k diacritised and undiacritised word types. The SAC text[5] is randomly chosen prophet sayings with a total of 4.5k words with 1.5k and 1.2k diacritised and undiacritised word types. The SALMA corpus was not chosen because the annotated data is small compared to the QAC. It is, however, used in the intrinsic alignment evaluation between different tagsets.

The dataset size in all experiments are split into 80/10/10. The data is shuffled in advance, and exact splits are then determined. For experiment replicability and fair comparison between experiments, the rearranged data could be replicated as the seed of the random generator is set in advance at the start of the code (setting the seed makes the random sequences predictable).

6.3.3 Segmentation

Different segmentation schemes are introduced in the literature with no one defined as a standard because "there is no single optimal tokenisation" (Habash, 2010).

QAC and SALMA followed a fine-grained tokenisation that is influenced by traditional Arabic text. For example, QAC segment an emphatic (ن /n/ *noun letter*) suffix that attach to verbs. This segmentation is influenced by traditional Arabic

[3] Only for nominals.

[4] Number is either singular or plural.

[5] Please note that SAC was not used in this chapter as it was not fully annotated when these experimental studies were conducted. For the sake of fairer comparison between proposed approaches, which includes a reimplementation of all proposed architures in neural networks, please refer to 7.5.

grammar as this letter changes the mood of the verb (becomes [مبني] "*invariant mood*"). None of the taggers segment this emphatic letter.

MX and AM can be configured to segment the text in different "tokenisation aliases" (Pasha *et al.*, 2015). They use a very similar engine (MADAMIRA is the successor of MADA and AMIRA). D3 segmentation is used where basically all clitics are tokenised. It is the most fine-grained segmentation schema. Similarly, AM allows the user to choose which prefixes and suffixes to split off. Its "default" scheme is chosen where conjunctions, prepositions, determiners, suffixes and future markers are all individually separated. ST and FA do not allow the change in segmentation scheme. ST follows *ATB* schema: all clitics are tokenised except determiners.

In a more in-depth look, FA is a bit different: e.g. it segments off nominal suffix that marks the plurality of a noun. It segments the Alif tanween that marks the accusative case of nominals. Unlike traditional Arabic, the others do not segment the attached nominative pronouns (that acts as a subject). These differences contribute to the increase in the number of tokens in FA. ST does not segment the Al+ article from nominals. The significant portion of the rest of the differences is due to errors in the model's predictions.

The difference in the number of segments shows that both QAC and SALMA used more fine-grained segmentation schemes. It shows the challenge of adapting MSA segmentation schemas to traditionally influenced segmentation. It also indicates that segmentation varies widely between different Arabic POS-taggers.

6.3.4 Tagset

The tagsets used by the two reference corpora differ: The QAC tagset is more syntactically-driven while the SALMA tagset is more focused on the internal morphology of the words and has more morphological features (total of 22 elements). The QAC tagset is designed only for the Quran; thus it does not have tags for punctuation for example. Regarding basic tags, the possible number of part of speech categories (without any associated features such as gender or person) in the QAC is 45: 9 tags for nominals, one for verbs, and 34 tags for particles. The SALMA tagset is more fine-grained (77 tags): 34 for nouns, one for verbs, 22 for

particles, and 20 for residuals (others). Irrelevant tags (e.g. punctuation) were excluded.

Tagset of the four taggers range from 16 (FA) to 26 (AM, ST) to 59 tags (MX). Table 6.2 illustrates a mapping from each tag of each tagset to the universal dependencies tagset. This mapping is a rough mapping and is carried out by the author with no validation. It is not used in any alignment methods. The goal is to show how tagsets are widely different in some groups. For example, relative pronouns are considered a particle in FA. It does not have a proper_noun tag either. It also has tags that are a morphosyntactic feature, e.g. NSUFF. ST and AM have a very similar tagset. MX uses a separate tagset for enclitics (24) and proclitics (28) that are not included in the table. These tagsets not only consist of POS tags; they sometimes encode the form of the clitic and some morphosyntactic features. Some tags are specific for Egyptian Arabic dialect clitics.

Table 6.2 Rough Mapping of Tagsets with Universal Dependencies tagset

	UPOS	QAC	MadaAmira	Stanford	AMIRA	FA
Open	**NOUN**	N	noun	NN, NNS	NN,NNS	NOUN
	PROPN	PN	noun_prop	NNP,NNPS	NNP,NNCD	
	ADJ	ADJ	adj	JJ,VN	JJ,JJCD,JJR,VN	ADJ
	ADV	T,LOC	adv,adv_interrog,adv_rel	RB	RB	ADV
	VERB	V	verb	VB,VBD,VBG, VBN,VBP	VB,VBD,VBG,VBN,VBP	V
Closed	**ADP**	P	prep	IN	IN	PREP
	AUX *					NSUFF,CASE
	DET	DET	part_det	DT,NOUN_QUANT	DET	DET
	PRON	PRON	pron,pron_exclam	PRP,PRP$	PRP	PRON*
		REL	pron_rel	WP,WRB	WP,WRB	
	CCONJ	CONJ	conj	CC	CC	CONJ
	NUM	NUM	noun_num,noun_quant,adj_num	ADJ_NUM,CD	ADJ_NUM,CD	NUM
	SCONJ	SUB	conj_sub		CJP	PART
	PART	IMPN,DEM,EXL,FUT,INTG,NEG,RES,VOC, EMPH,IMPV,PRP,ACC,AMD,ANS,AVR,CAUS, CERT,CIRC,COM,COND,EQ,EXH,EXP,INC, INT,INTG,PREV,PRO,REM,RET,RSLT,SUP, SUR	pron_dem, pron_interrog, part_focus,adj_comp,part_dem, part, verb_pseudo, part_fut, part_interrog, part_neg, part_restrict, part_verb, part_voc	RP	DT,RP,CJP	FUTPART
	INTJ	N/A	interj	UH	UH	
Other	**PUNCT**	N/A	punc,latin	PUNC	PUNC,FP	PUNC,FOREIGN
	SYM	N/A				
	X	N/A	abbrev,digit			ABBREV

6.3.5 Parallel-Aligned Corpus (PAC)

The alignment problem needs a source of information for its decisions, i.e. our parallel-aligned corpus. Similar to the role of bilingual dictionaries in machine translations, supervised alignment methods, that try to align the output of individual taggers of the ensemble, use this corpus to infer the best alignment candidate.

Because assessing the generalization capability of the alignment methods is a key goal in our study, and since the alignment is sensitive to the similarity between the source and target aligned tagsets, we chose the 29th chapter of the Holy Quran as the gold standard corpus for *alignment*. This specific chapter is annotated by QAC (Dukes, Atwell and Habash, 2013) and SALMA (Sawalha, Atwell and Abushariah, 2013) which makes it a good candidate for a parallel-annotated corpus. The corpus is enriched with semi-automatic tagging using the four taggers. Although one reference corpus might be enough for comparing different alignment methods, it is preferable to test the alignment on multiple corpora as alignment is dependent on the similarity between the tagger and reference tagset.

The reference corpus is nearly 1000 words, morphologically segmented to produce 1709 (QAC) or 1942 (SALMA) morphemes. FA, ST, MX, and AM produce a different number of morphemes: 1615, 1448, 1426, 1409 morphemes respectively.

The goal of this PAC is to evaluate the alignment methods. In pipelined ensemble, each component is tuned independently. Assessing the alignment method is required to optimise the ensemble tagger to achieve the best accuracy. Another aim of this language resource is to construct a reference data for evaluating taggers based on one test dataset.

The corpus development process is simple: Input taggers re-tag the corpus using their own labels. These morphemes were manually aligned by the author of this thesis to SALMA and QAC. The alignment is done per word for all taggers. A single morpheme can be aligned to only one reference morpheme, which assumes that QAC and SALMA are finer in the segmentation process. If one morpheme can be aligned to multiple reference morphemes, the most similar morpheme is chosen in terms of POS tag similarity and is judged solely by the author. The other morpheme is aligned to a gap in the reference word. The whole process was done on the SAWAREF web-interface. The morphemes are shown in tabular format with

easy navigation and keyboard shortcuts. The corpus is open access at the SAWAREF data repository[1].

6.3.6 QAC Orthographic Adaptation

The script used by the QAC corpus differ significantly from the text used by input taggers (see Section 8.2.3 for a list of differences). The QAC corpus used the Othmani script, where 43.16% of the verses and 52.80% of the words are written differently from a version written using Modern Standard Arabic script. It requires special handling and manual verification to convert it to the modern orthographical standard Arabic script. As taggers assume text to be written in modern standard orthography, we used the Tanzil Project[2] to retrieve an authenticated modern script version of the Quran text.

To rewrite each segment to its MSA form, we perform the following procedure:

1. If the word is composed of one segment, we replace it with Tanzil's form.
2. Else if there is only one segment that differs in the form, we find and use the proper substring from Tanzil's form.
3. Else, we try to rewrite each segment in the QAC using attached ordered regular expressions which convert Hamza, Yaa, Special characters, madd, and missing diacritics to the required format.
4. We repeat step 2, unless no new changes are made.
5. If there are still two different segments, we raise an error, and manual handling is required.
6. Any remaining mismatching segments are treated manually.

The final result of this adaptation is a version of the Quranic Arabic Corpus that conforms with modern Arabic orthography. This adaptation, however, does not claim that the QAC is now perfect for training machine learning models for classical Arabic. In Section 8.2, a more detailed evaluation of the QAC is presented.

[1] https://github.com/aosaimy/sawaref-data

[2] http://tanzil.net/

6.4 Morpheme-based Alignments Methods

6.4.1 Baseline Alignment

This approach is a simple method to jointly align the output of two sequences. one element (called primary) is selected from each side and are linked; then, other elements are aligned with respect to their relative position to the primary element, assuming taggers will produce a sequence of morphemes with one morpheme marked as a `primary'. Formally, let the primary morpheme be: $a_i \in A, b_j \in B$. The result of the mapping is the set of pairs: $C = \{.., (a_{i-2}, b_{j-2}), (a_{i-1}, b_{j-1}), (a_i, b_j), (a_{i+1}, b_{j+1}), (a_{i+2}, b_{j+2}), ..\}$.If a_i or b_j do not exist in their respectful sets, they are substituted by a gap (or ϕ).

To illustrate this method: Assume we have two sequences: $A = \{conj, prep, verb\}$ $B = \{pc, pj, ra, vc, rb\}$ where $verb$ and vc are the primary morphemes. The alignment result will be: $\{(\phi, pc), (conj, pj), (prep, ra), (verb, vc), (\phi, rb)\}$.

This method makes three assumptions:

1. **The definition of primary morphemes is standard:** An example that illustrates the lack of this standard is the case of PREP + PRON which is common in Arabic; an example is (فيه /*fyh/ in it*).
2. **A word has only one primary stem** which is invalid when two morphemes are equal in rank. (إِنَّمَا /*<in~amaA/ but*) was segmented by QAC into two primary morphemes: *<in~a/ACC + maA/PREV*.
3. **Taggers will explicitly mark one morpheme as primary:** Some taggers do not.

To overcome these issues, tags in the tagset are ranked. The top-ranked morpheme in one word is marked as primary. This method should solve the three problems; for example, a higher priority might be given to PRON than PREP, and to ACC than PREV.

The noticeable difference in the segmentation schemes makes this baseline algorithm not efficient, so we investigated three different approaches to improve the alignment.

6.4.2 Rule-based Alignment

In this approach, rules that map one tagset to the other guide the alignment algorithm. They are used to constrain the alignment to only mapped pairs (if such exist).

Two linguists performed the task. See 5.5 for the experimental study of mapping. The scoring matrix $S^{A,B}$ is constructed as follows:

$$s_{i,j}^{A,B} = s_{j,i}^{A,B} = \begin{cases} 1 \; if \; f(i,j) = 2 \\ 0.5 \; if \; f(i,j) = 1 \\ 0 \; otherwise \end{cases}$$

where $f(i,j)$ is the number of mappings from tag i to tag j.

6.4.3 Data-driven Supervised Alignment

The second approach uses an aligned corpus to learn the probability of aligning one morpheme in one sequence to another, using its POS tag. We used our parallel annotated and aligned corpus (PAC)[3]. Incorrectly-tagged words were marked and skipped from learning.

To construct the scoring matrix, we use two basic methods: weighted count unigram and bigram. Then, these counts are normalised by dividing them on the total number of POS tag occurrences. The scoring matrix $S^{A,B}$ is constructed from the co-occurrence matrix C as follows: $s_{i,j}^{A,B} = s_{j,i}^{A,B} = \frac{c_{i,j}}{\sum_k c_{i,k}}$

6.4.4 Unsupervised Alignment

This approach uses a method adapted from the word alignment task in Statistical Machine Translation (SMT). Similarly, our corpus is multilingual (in the sense of annotation style), and is parallel at the word-level. The unsupervised alignment is done by linking POS tags of the two sides of the word-level parallel aligned corpus using the likelihood of co-occurrences.

Using our PAC corpus, we use the *fast_align* method (Dyer, Chahuneau and Smith, 2013) which uses the expectation-maximisation algorithm to maximise the likelihood of a parallel corpus. The input to the aligner looks like the following ("|||" denotes the delimiter between source and target languages):

wa_conj li_prep verb ||| p--c-- p--z-- r---a- v-c--- r---z-

[3] http://github.com/aosaimy/sawaref-data

The result of this approach is a pairs of links, that depends on the likelihood of having two tags appearing on the same word. These links may intersect, i.e. the alignment output is not necessarily projective. For example, an enclitic in one side may be linked to a proclitic on the other side. The used method, *fast_align*, is designed for word alignment of a bilingual parallel corpus, and intersection is possible in SMT. By using a priority that favours arrangements that are close to "diagonal", we could force the alignment to respect the projectivity property.

Post-processing the output was necessary to convert n-n mappings to one-to-many mapping. Among the *m* possible mappings, and rather than basically choose the first one, we pick the most confident mapping, i.e. the most-frequent pair in the whole training.

6.5 Form-based Ensemble

Previous morpheme-based approaches do not assume same tokenisation schema of the gold standard corpus. Some tokens in the gold standard corpus (ex. EMPH) will not be identified, as no input tagger assumed the same tokenisation. When tagging the word /yatyn/, if tagger A produced "yAtyn/VBP km/PRB" and tagger B produced "yAtyn/V km/PRON", then the following input and output will be expected from the morpheme-based ensemble classifier:

Input		**Output**	
Word	Features	Class	May be predicted
yAtyn	VBP_A V_B	V	Yes
		EMPH	No
km	PRP_A $PRON_B$	N	Yes

The form-based approach extends POS tagset of the gold standard with a character that indicates a character's position: *B* for the first character of morpheme, *I* for other characters. Spaces between words are tagged as *O.* The previous example will be:

Word	**Features**	**Class**
y	VBP_A V_B	B-V
A	VBP_A V_B	I-V
t	VBP_A V_B	I-V
y	VBP_A V_B	I-V
n	VBP_A V_B	B-EMPH
k	PRP_A $PRON_B$	B-N
m	PRP_A $PRON_B$	I-N

This approach can combine different taggers with no assumption of the same tagset or segmentation. Please note that this approach aligns segments based on the characters of the segment's form; i.e. it assumes that for each segment, the tagger will output the segment form and the part-of-speech tag. It does not apply to word-based taggers like MADAMIRA (by default[4]). For example, the word (فَتَنَهُ */fatanahu/ entice;torment (him)*) is tagged by MADAMIRA as follows: *pos="verb" enc0="3ms_dobj"*.

A form-based ensemble requires inputs to be aligned at the character-level. Character-based alignment uses the lexical form of the segments provided by segmenter/tagger to align the output. This approach was used in the GRACE evaluation campaign (Adda *et al.*, 1998) to align several participating taggers using a word-based "diff" tool.

6.6 Alignment Evaluation

We evaluate different approaches to alignment using an *intrinsic* metric: The accuracy of aligning morphemes. Using 80-20 split for training and testing, we report overall accuracy: the fraction of *morphemes* that have been correctly aligned to the PAC gold-standard corpus. An incorrect alignment will cause at least a doubled penalty in this metric.

Table 6.3 The morpheme-based accuracy of aligning morphemes using five approaches of alignments.

Mapping	Ru	unigram	bigram	baseline	unsup	unsup*
AM → QA	0.91	0.97	0.83	0.95	0.9	0.91
AM → SW	0.90	0.96	0.72	0.94	0.83	N/A
FA → QA	0.91	0.99	0.84	0.95	0.95	0.95
FA → SW	0.97	0.99	0.95	0.95	0.96	N/A
MX → QA	0.92	0.95	0.81	0.91	0.92	0.92
MX → SW	0.93	0.94	0.71	0.90	0.83	N/A
ST → QA	0.94	0.98	0.82	0.92	0.89	0.90
ST → SW	0.93	0.96	0.72	0.91	0.82	N/A
Average	0.93	0.97	0.80	0.93	0.89	0.92

[4] One of the outputs of MADAMIRA is the original Buckwalter complex tag. The tag shows the segments of the word, but its segmentation schema is finer than MADAMIRA.

Since there is a chance that one tool tags a word incorrectly, and this word will contribute to the error rate of the alignment, these erroneous words are excluded them from our training and evaluation. While sometimes just one morpheme is marked incorrectly, the whole word is excluded.

We performed 5-fold cross-validation for the supervised approach and the reported accuracy is the average of the five folds. The unsupervised model used the full unaligned corpus for training. However, evaluation is based on the same test portions as the supervised approach. In the *unsup** column, we report the accuracy of unsupervised learning from a larger training data (nine times original size), and accuracy has increased by around 0.5-1%.

The results in Table 6.3 show that the unigram model outperforms all other models in all our tagsets mappings. We can see that aligning taggers with SALMA is more difficult than with QAC because QAC uses segmentation and labelling schemes that are more compatible with input taggers. One exception is FA which seems to be more compatible with SALMA than QAC.

The bigram model suffered from the insufficient training corpus even though the bigram model uses more contextual information to predict alignment. One solution to this problem is back off strategies to unigram model, e.g. using Katz's backoff model (Katz, Lamel and Adda, 1987).

The unsupervised method suffered from the post-processing step which converts n-n mappings to 1-1. Both basic and most-confident strategies suffer from cases where a tag is more associated with another tag, e.g. verbs frequently collocate with pronouns. For example, using the basic method, the tag verb would have paired with `ra` (imperfect particle) instead of `vc` (imperfect verb). Since the pair (`verb`, `vc`) is more common, the most-confident method will pick this pair instead. While the most-confident method should improve the accuracy, it fails to choose the right pair when there are affixes that appear more than their stem. For example, the tag `noun` was paired with `nu` (active participial noun) and `rm` (masculine plural sound suffix), but since the SALMA tagset is finer grained, a noun can be mapped to at least 15 possible tags, which lowers the probability of `noun` → `nu`. Thus, this method chose the incorrect pair (`noun`, `rm`).

6.7 Morpheme-based Ensemble Evaluation

We evaluated different approaches to alignment using an *extrinsic* metric: The effect of alignment methods using an application of the alignment (one-to-one POS tagging). In this evaluation, we used the QAC POS-tagging of ten randomly-selected chapters. We compare alignment methods with an ensemble tagger with "no" alignment; i.e. no intervention is done to the natural order of morphemes. The Random Forest method implemented in the WEKA toolkit (Breiman, 2001) is used for the *morpheme-based* ensemble development. Random Forest has been widely used for classification problems, e.g. the gender and number tagging of Arabic words (Darwish, Abdelali and Mubarak, 2014).

We extend the alignment algorithm to work with multiple input taggers. We used a simple method: having randomly ordered two sets, aligned and non-aligned, we sequentially align one from the non-aligned set with the last-added tagger in the aligned set.

Formally, let T be the set of taggers and P be the set of aligned taggers with a size m initialised by randomly adding one tagger from T to it. Then, we select and align a randomly picked tool from $T - P$ and align it with p_{m-1} then add it to P.

While this greedy algorithm does not ensure optimal multi-sequence alignment, it performs well enough in our PAC corpus, and its decrease in accuracy seems negligible: 0.025. However, this method makes errors of prior alignment are propagated to the next pair. The reduction in accuracy was caused mainly from incorrectly labelled words; i.e. aligning two incompatible outputs.

Only aligned labels were provided to the classifier. We do not edit mislabelled segments, nor ignore them in training. Note that our individual data points were assumed to be independent, and we rely on input taggers to consider the context for classification. A sample of the input to the classifier is Table 6.4.

Table 6.4 A sample of input to the ensemble POS tagger.

MX	AM	FA	ST	QAC
verb	VBD	V	VBD	V
prep	NN	PREP	IN	P
2ms_pron	PRP	PRON	IN	PRON
det	DET	DET	DT	DET
noun	NN	NOUN	NN	N
prep	IN	PREP	IN	P
det	-----	DET	DT	DET
noun	NN	NOUN	NN	N

The results show that as we increase the number of taggers, the accuracy improves, (see). We can see that the effect of alignment decreases as we increase the number of input taggers though. Errors generated from the greedy method might have cancelled the gain of more taggers in the ensemble tagger. The ensemble tagger improved the accuracy over the best input tagger by at least 1.7%. The best ensemble tagger was an ensemble of AM, ST, and MX taggers with an accuracy 88.09%, 88.07%, 87.88%, 87.74% (using unigram, rule-based, baseline, and without any alignment respectively). However, the ensemble of all four input taggers performed a little bit worse: 87.80%, 88.06%, 87.92% and 87.90%.

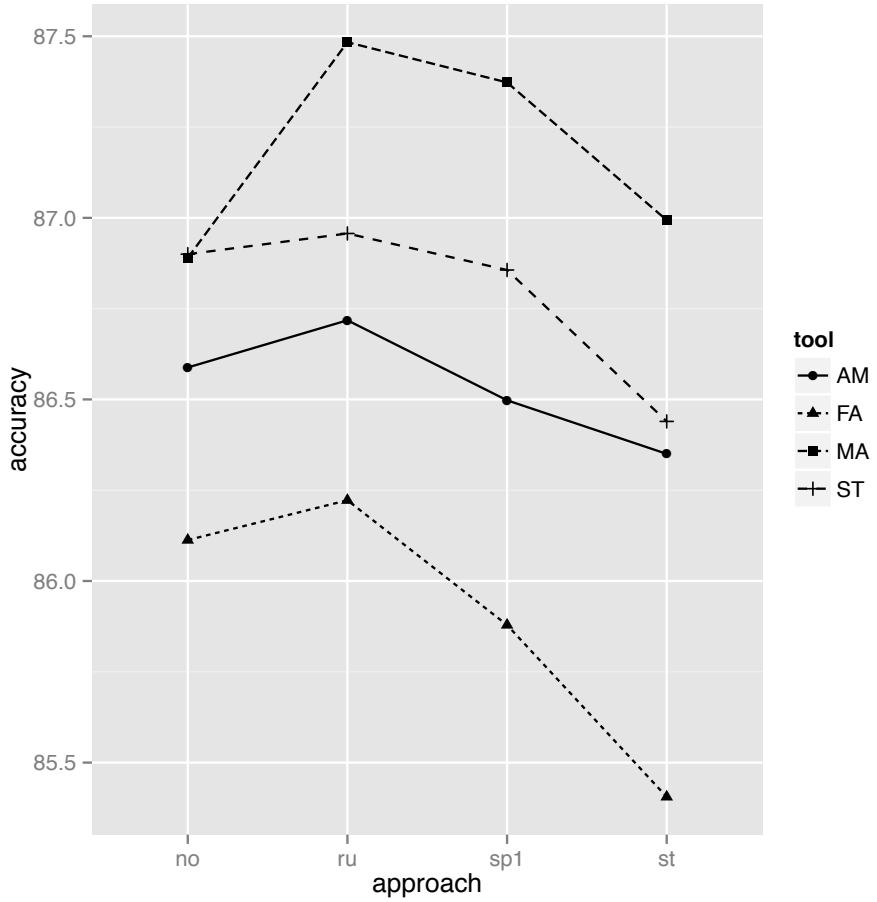


Figure 6.3 The average accuracy of each input tagger against different alignment approaches.

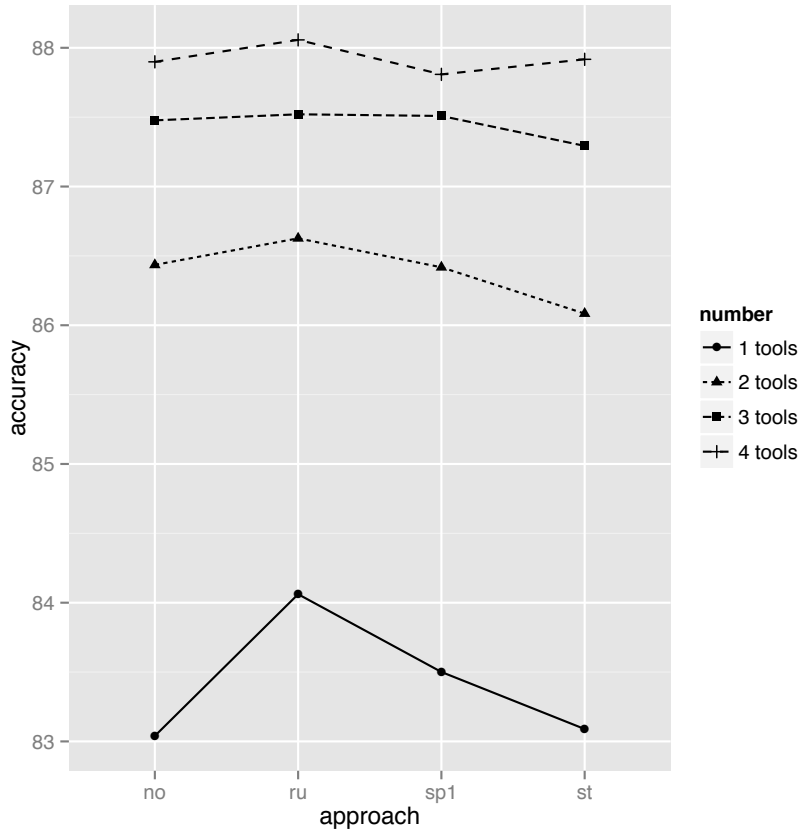


Figure 6.4 The effect of increasing the number of input taggers against different alignment approaches.

The alignment between taggers seems to increase the tagger performance slightly. Overall, the average improvement in accuracy is 0.01 and 0.036 for the unigram and rule-based approaches respectively. The fact that the rule-based approach performed better than the unigram approach does not contradict the intrinsic evaluation, as erroneous words are removed in the intrinsic assessment. The training dataset for the ensemble tagger is considerably larger than one used in the intrinsic assessment. Unseen tags might contribute to the difference as well.

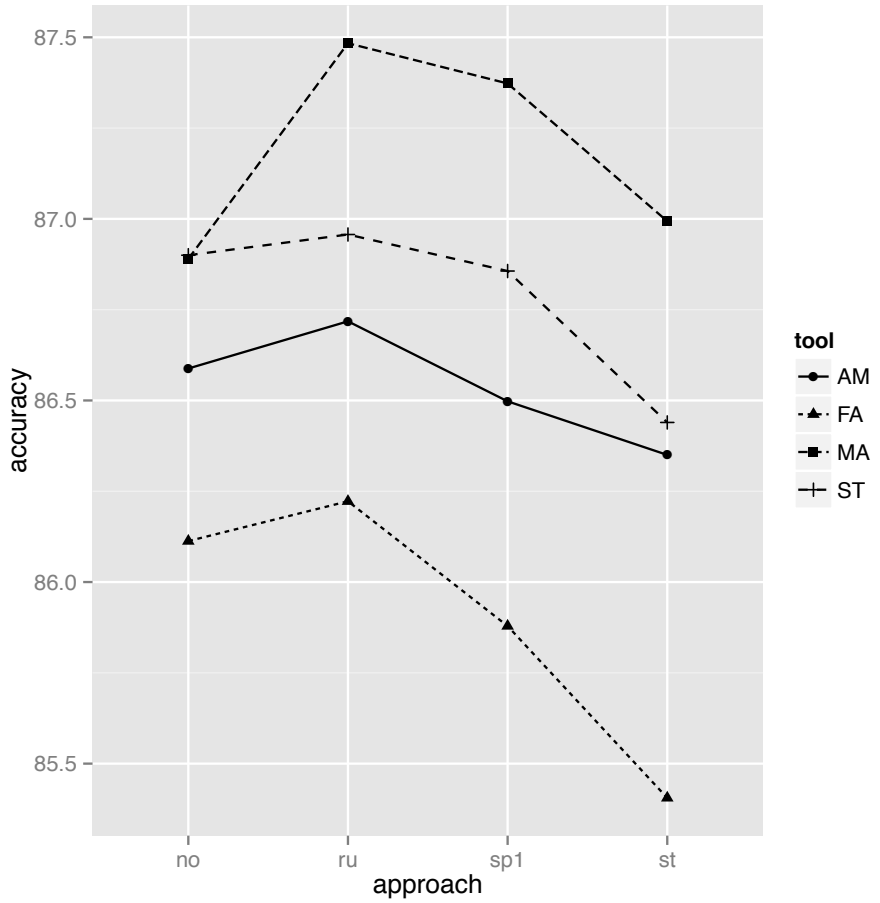


Figure 6.5 Input taggers differ in their contribution to the ensemble tagger.

The Figure 6.5 shows the average accuracy of all combinations of ensemble taggers that include the selected input tagger. It indicates that MX contributes the most to the ensemble, and alignment improved its accuracy noticeably. This contribution might be due to its fine-grained tagset. While FA used a more fine-grained segmentation scheme, its small tagset makes it less helpful to the ensemble tagger.

One significant disadvantage of this alignment is the dropping of segments that never appear in input taggers. One example is the EMPH tag, which was used in the QAC to mark the EMPH enclitic in verbs (see Table 6.7). Input taggers never segment this enclitic; instead, they tag it as a part of the verb.

6.8 Form-based Ensemble Evaluation

In this experiment, the used evaluation metric is accuracy, the fraction of correctly tagged *characters* ($Acc_{Morpheme}$) and *morphemes* ($Acc_{Character}$). Since character-based methodology preprocesses the input to convert it to a suitable

format for training, the results are post-processed such that we can compare it to other morpheme-based results.

The form-based method used a Java-based package called MALLET for sequence tagging using Conditional Random Fields. The form-based method required us to redefine the problem as a sequence problem as character positional tags plays a critical role in prediction. In the next chapter, we reimplement the two approaches and evaluate them on a common dataset and platform with the end-to-end approach proposed there.

In the character-based method, our results are comparable with the advantage that we do not require the prior assumption of similar segmentation scheme between taggers. The best combination is the ensemble that includes all four taggers and scored 88.73%.

Table 6.5 A comparative accuracy between morpheme-based and character-based approaches

Method	$Acc_{Morpheme}$	$Acc_{Character}$
Morpheme-based Ensemble	88.09	N/A
Form-based Ensemble	88.73	92.44

Note that morpheme-based accuracy is computed by recovering the morpheme from the character-level labelling. However, this does not produce necessarily the same number of morphemes in the gold-standard corpus. This results in a mismatching morpheme number between the two sequences of morphemes. The morpheme-based accuracy marks a morpheme as correctly labelled if all its characters are tagged correctly.

6.9 Morpheme-based vs Character-Based Alignment

We will start this comparison by dividing POS taggers for Arabic into two categories:

1. **Word-based Taggers** where the word as a whole is given a compound tag with no explicit mark for segmentation in the form; i.e. there is no link between segments' lexical form and their POS tags in the compound tag. Examples include the MX tagger and the Microsoft POS tagger (see 4.4.6). The tagset for enclitics might even be different than ones for the stem.
2. **Segment-based taggers:** Each segment is clearly defined by its morphological information. Some taggers mark enclitics by adding a plus

sign to indicate that it was split off from the previous/next segment.

Examples that include ST, FA, and AM taggers.

Table 6.6 Word-based vs. Morpheme-based tagging. For the word /kunna/, word-based do not specific the mark of where the word is split.

Word-based	**MADAMIRA Toolkit**	<morph_feature_set diac="كُنَّا" lemma="1_كان" bw="kun/PV+nA/PVSUFF_SUBJ:1P" gloss="was;were" pos="verb" prc3="0" prc2="0" prc1="0" prc0="0" per="1" asp="p" vox="a" mod="i" gen="m" num="p" stt="na" cas="na" enc0="0" source="lex" stem="كُن"/>
	Microsoft POS Tagger	V.Dual.Plu.Pst.Act*Subj.Plu.1
Morpheme-based	Elixir FM	VCJ---MS-- kun "k w n" "" >>| FuL |<< "" kun كن كُن SP---1MD4- nā نا نَا
	QAC	**ku V STEM|POS:V|PERF|LEM:kaAna|ROOT:kwn|1P** **n-aA PRON SUFFIX|PRON:1P**

Using the character-based approach for aligning segments between taggers is challenging because some of the input taggers to the ensemble tagger were word-based. Table 6.6 shows the two ways of tagging the word: (كُنَّا, *kuna~A*, we were). Even though the tag in the second row indicates that there are two segments in the word: `V` and `Subj` (separated by the star sign), there is no mark to indicate where the word form should be divided; i.e. the segment form is missing.

Segment-based taggers have their own issues. One issue is the effect of segment form adjustment when it is attached. When a word is split off into segments, the segments might require some modification to recover their original form. There are at least four reasons for such differences:

1. **Taa Marbouta letter**: the Ending Taa Marbouta is converted to normal Taa when concatenated to another segment, as it never appears in the starting/middle state. Splitting off segments might require recovering the Taa Marbouta letter.
2. **Maddah diacritic**: This is originally constructed from two letters: ("ا+أ", />*a*+*A*/, "Hamza with fatha and Alif letters"). For example, questioning Alif is converted into Alif with Maddah when concatenated to word with starting Alif. When splitting off segments, the Maddah diacritic should be recovered to its original two letters.

3. Concatenation of **Prepositional Lam and the determiner Al:** This concatenation drops the Alif of the determiner, and in exceptional cases drops both letters. Recovering those dropped letters depends on the context.
4. **Consonant gemination mark (i.e. Shaddah)**: This indicates consonant doubling of the letter. However, it happens that the gemination is caused by attaching a clitic to the word; thus, the letter correlates with both segments. For example, possessive Yaa is converted into a consonant gemination mark when attached to a nominal word that ends by /*y*/. Prepositional segment /*mino+*/ when concatenated with relative /*+maA*/ is shortened as /*mimaA*/.

These differences result in *different forms* of the same segment between taggers; for example, Table 6.7 illustrates how the inflected word "*wa+mi+mA*" is morpheme-based aligned and recovered by various tools.

Table 6.7 Different recovery of word's segments

	MX	ST	AM	FA
wa	wa/conj	w/conj	w/CC	w/CONJ
mi	mino/prep	m/IN	mmA/NN	mmA/part
m~aA	mA/rel	mA/WP	-	-

This illustrates incompatible segmentation schemas, and more importantly, it shows that MX recovered the */mino/* original form and therefore an extra letter */no/* is added that was not originally in the word form. QAC, the gold standard corpus, is segment-based tagged but converted letters were not recovered after splitting off segments. As a result, a few segments do not have a segment form (as the segment form was part of another segment, e.g. possessive Yaa).

Besides, not all taggers report the segment fully voweled (with diacritics). Back to the "*wa+mi+mA*" example, only MX reports segment diacritics (letters: *a, o, i, u*).

Furthermore, taggers do not always follow the same procedure of *normalisation*. The ST tagger, for example, by default normalises all Alif shapes into the normal Alif. This results in a mismatch between characters. In Appendix A, one full sentence (Hadith verse) is tagged by several taggers, and changes that are made to the segment form can be seen in context.

Morpheme-based alignment requires external resources to find the links between morphemes. These resources are usually not perfect and prone to errors. It also suffers in the case of multiple tagger alignment, as the optimal alignment is computationally expensive. However, it can work with word-based taggers and segment-based taggers. Other related work seems to prefer character-based approach, especially for Chinese.

6.10 Conclusion

This chapter presented and compared two approaches of heterogenous pipelined ensembles of part-of-speech taggers: morpheme-based and form-based. Morpheme-based ensembles using three methods of alignment were evaluated. The supervised learning method using a unigram model had the best morpheme-based alignment accuracy evaluated on the specific aligned PAC corpus. However, morpheme-based ensembles using a rule-based approach were better in terms of accuracy. This might show that individually-tuned pipeline ensembles might not be the best model. Using alignment improved the ensemble POS-tagger accuracy by 3.6%.

For future work, a more complex vector that includes morphological features might be considered in the alignment methods, especially for the unsupervised approach. Additionally, this work should be extended to include morphological analysers so that the ensemble tagger jointly disambiguates and votes for the correct analyses.

The next chapter introduces a new model for the ensemble problem: a joint ensemble with an implicit alignment using an encoder-decoder architecture. The goal is to overcome the problem of individual tuning of alignment and the requirement of explicit mapping (data-driven or rule-based).

7 END-TO-END ENSEMBLE TAGGER

Chapter Summary:
Pipeline one-to-one ensembles suffer from the requirement of explicit alignment of segmentation schemes and independent tuning of each component in the pipeline. Inspired by neural machine translation advances, this chapter introduces a joint end-to-end ensemble using an encoder-decoder approach.
A series of experiments are executed to evaluate the approach with consideration of the model of encoder-decoder models, the use of word embedding, the contribution of each input tagger, coarse vs fine-grained tagsets, and different train dataset size.
The second part involves a comparative analysis of the proposed approaches: pipelined morpheme-based, form-based and joint end-to-end ensembles. The results are compared with related work in the literature.
Before concluding, errors generated from these ensembles are examined and discussed. In light of these errors, this chapter concludes with some suggestions for future work.

7.1 Introduction:

So far, all experiments of the ensemble tagger have been made on parts of the Quranic Arabic Corpus (QAC) and were limited to the POS tag only. In this chapter, different setup configurations are evaluated for the ensemble tagger trained on parts of the Sunnah Arabic Corpus (SAC). The experiments are evaluated on the test part of the SAC. The goal is to predict the POS tag, segmentation and eight morphological features.

Pipeline one-to-one ensembles suffer from several problems:

- the requirement of explicit alignment of segmentation schemes;
- independent tuning of each component in the pipeline; and
- propagated errors in subsequent tasks.

Inspired by neural machine translation advances, this chapter implements a joint end-to-end ensemble using an encoder-decoder approach.

In this chapter, we use Deep Learning algorithms for the prediction, specifically recurrent neural networks. Deep Learning is a machine learning method that uses layers of processing units where the output of a layer cascades to be the input of the next layer. Recurrent Neural Networks (RNN), where iterative function loops are used to store information, have been successfully applied to sequence labelling problems in Arabic such as Arabic diacritisation (Abandah *et al.*, 2015; Rashwan *et al.*, 2015), Word Segmentation and Morphological Disambiguation (Darwish and Abdelali, 2017; Zalmout and Habash, 2017).

In the first part of this chapter, different parameters and machine learning features are examined. It inspects the effect of using word embedding, illustrates each input tagger contribution to the ensemble tagger, shows the effect of different coarse and fine-grained tagsets, and finally plots the effect of the train dataset size. Since the configuration space is vast in our case, a greedy approach is followed to find the best model for annotating the rest of the Sunnah Arabic corpus.

In the second part, previous approaches were reimplemented in neural network architecture with the goal to evaluate them using the same platform and datasets.

Before concluding, errors generated from the best model are examined and discussed. We compare the ensemble errors with the input taggers outputs. In light of these errors, we give our suggestions for future work.

7.2 Sequence Labelling: One-to-One vs Sequence-to-Sequence

Sequence labelling usually involves the prediction of the next label based on sequence of input and the predicted labels so far, e.g. POS tagging. Sequence labelling is distinguished from pattern labelling by the fact that individual data points (or time steps; words in POS tagging) cannot be assumed independent from other data points.

In sequence labelling, one or many input data points is often transcribed with *one* label (1-to-1, or many-to-1). However, there is a more challenging problem in sequence labelling that takes a sequence of input data points and transcribes them with a *sequence* of labels (many-to-many), e.g. machine translation. However, this problem might not be defined as sequence labelling, as it is no longer labelling "each" token in the input sequence.

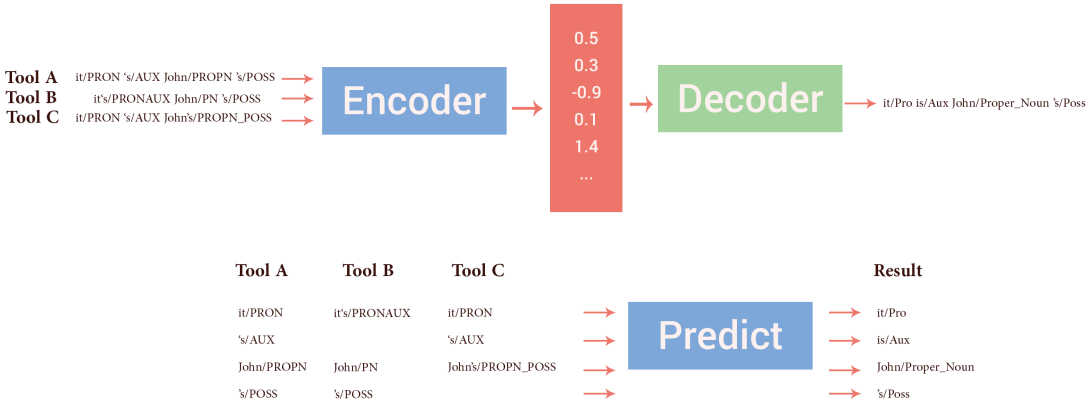


Figure 7.1 Seq2Seq model (top) vs One-to-one model (bottom)

In chapter 5, we formulated our problem as one-to-one sequence labelling. It required alignment between the predicted morphemes in the participating individual taggers. Pipeline architecture suffers from error propagation: errors generated from one alignment are propagated to subsequent tasks (e.g. another alignment in multi-tool settings, POS tagging, segmentation). It also suffers from independently tuning each task: the alignment in chapter 6 was not tuned for POS tagging. Neural networks offer an End-to-End solution and show significant advances in neural machine translation (NMT) over the traditional pipelined statistical machine translation (SMT).

RNNs are flexible in their inputs and outputs, and it is one of the reasons for choosing them. Many-to-many sequence labelling with neural networks is often

done using RNN Encoder-Decoder architecture (sometimes called seq2seq), illustrated in Figure 7.1 as introduced by Cho *et al.* (2014).

The use of Encoder-Decoder architecture removes the dependency on input shape, which is valuable in two needed outputs: recovering adjusted letters and recovering mismatches in the segmentation of the target tagset. It not only recovers dropped/converted items like the recovery of adjusted Taa Marbouta and Yaa Maqsoura letters, but it recovers mismatches in segmentation between input and output like the case of a missing EMPH tag.

In the encoder-decoder model, which has become the standard for seq2seq models, the alignment between the morphemes in advance is not required. The tagger should learn it implicitly. The input sequence (i.e. the output of the tagger) is read in entirety and encoded to a fixed-length internal representation. This representation is then used to extract the final required output tasks: POS tags and morphological features.

7.3 End-To-End Experiment Settings

The goal of these experiments is to build an ensemble of morphosyntactic taggers that predicts the POS tag, segmentation, and morphological features for automatic annotation of classical Arabic.

Both QAC and SAC follow the same POS tagset. The extended tags introduced in SAC (see 8.10.3) are for word categories that never appear in Quranic texts, e.g. digits.

7.3.1 Data, Participating Tools, Tagset and Morphological Features

The data used for training and testing and the participating tools are the same ones used in the previous experiment using pipelined alignment. Please refer to 6.3 for the details.

7.3.2 Network Architecture

The problem, as stated before, is a supervised sequence labelling. POS tagging and the prediction of every morphological feature are examples of sequence labelling problems where there are sets of labels for each problem. Segmentation can be seen as binary sequence labelling at the character level. A character is labelled as either a start mark of a new morpheme or not.

In sequence labelling, Recurrent Neural Networks (RNN), where a network uses iterative function loops to store information, has shown some advantages over standard feed forward neural networks. They are flexible in how they deal with contextual information. They can recognise sequential patterns better (Graves, 2012) as they use their internal memory to process sequences of inputs. It has been successfully applied to sequence labelling in Arabic such as Arabic diacritisation (Abandah *et al.*, 2015; Rashwan *et al.*, 2015), Word Segmentation and Morphological Disambiguation (Darwish and Abdelali, 2017; Zalmout and Habash, 2017).

From the class of RNN, the TensorFlow implementation of the Long Short-Term Memory (LSTM) architecture (Hochreiter and Urgen Schmidhuber, 1997) is used in all experiments. LSTM is a modified design of the standard RNN to overcome one serious flaw: the inability to store information for a long time. LSTM was the choice for the previously cited studies in Arabic diacritisation and morphological disambiguation, and therefore is the layer of choice to encode and decode sequences.

In all the experiments of joint end-to-end ensemble, we use a sequence-to-sequence (seq2seq) (Cho *et al.*, 2014; Sutskever, Vinyals and Le, 2014) neural network that is composed of an LSTM encoder and decoder. Although dynamic NNs perform well when sufficient training data is provided, it has required encoding the inputs and targets with vectors of fixed dimensionality. However, the dimensionality is not always known in advance in some of the tasks. This architecture allows mapping one sequence to another using two LSTMs: an encoder and a decoder. The first encodes the whole sequence in a fixed dimensional vector, and the latter decodes this vector into a newly generated sequence. This method proved to be useful in complex problems such as machine translation (Sutskever, Vinyals and Le, 2014).

The fundamental architecture is composed of an input layer, an encoder, and a decoder. An input layer is a 3-dimensional vector: (samples, time steps, features). There are two types of inputs: character-based input, which includes the lexical form, and categorical-based input, which includes POS tags and morphological features. In character-based input, the time steps are the characters

of the words, and each character is represented in a one hot encoding[61]. In categorical input, the time steps are the morphemes of one word, and the features are represented as well as one hot encoding.

The representation of input values as one-hot encoding implies that we have the same distance between different values of one feature. However, this is not always the case; for example, nouns and proper nouns might be closer to each other than a particle. One solution is to encode the sparse categorical vector in a dense vector of a fixed length. In one experiment, we evaluate the effect of using POS embeddings instead of the one hot encoding.

Note that the input sequences might not have the same length of time steps. However, in practice, it is required to pad the sequences to have the same length in Keras with TensorFlow backend. Unlike PyTorch[62], graphs in TensorFlow are defined statically. We pad string inputs with spaces, and categorical inputs with a null value to represent padding.

The next layer is a bidirectional LSTM encoder that maps the input shape (the time steps and the features) into a vector of 256 dimensions. In this layer, we use the hyperbolic tangent (tanh) as the activation function and a dropout rate of 0.01 (i.e. a fraction 0.01 of input units are set to 0 to help prevent overfitting). The output shape of this layer is a vector of 256 for each sample. The next layer repeats this vector to the number of time steps which is required for the next LSTM decoder layer, as it expects a 3-dimensional input.

The idea of bidirectional RNNs (Schuster and Paliwal, 1997) is straightforward. It duplicates the first RNN layer such that the input is fed to both layers, but with a reversed input order for the second layer. The output of the two layers can be merged via several methods, e.g. by concatenation. This approach requires that all timesteps of the input are available. Bidirectional LSTMs, in particular, were used in POS tagging (Plank, Søgaard and Goldberg, 2016; Darwish and Abdelali, 2017), and thus are used in our experiments.

---

[61] One hot encoding (a.k.a. dummy encoding) is a numerical encoding of categorical feature where a feature value is converted to a vector of 0 and 1. The vector size equals the length of labels set size. The vector is all zeros except for the label's index. For more details: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/

[62] https://pytorch.org/

The decoder layer is another LSTM that expects a sequence of fixed-length (256) vectors and will transfer the learned encoded internal representation into the output sequence. A regular feedforward dense layer is used to transform the output of each time step (morphemes) into the final label. The same weights are shared for all timesteps as the same dense layer is applied for each timestep. The network for predicting the segmentation is illustrated in Figure 7.2. The full network used in the experiments with all categorical and character information (POS tags, segmentation, and morphological features) is presented in Figure 7.3.

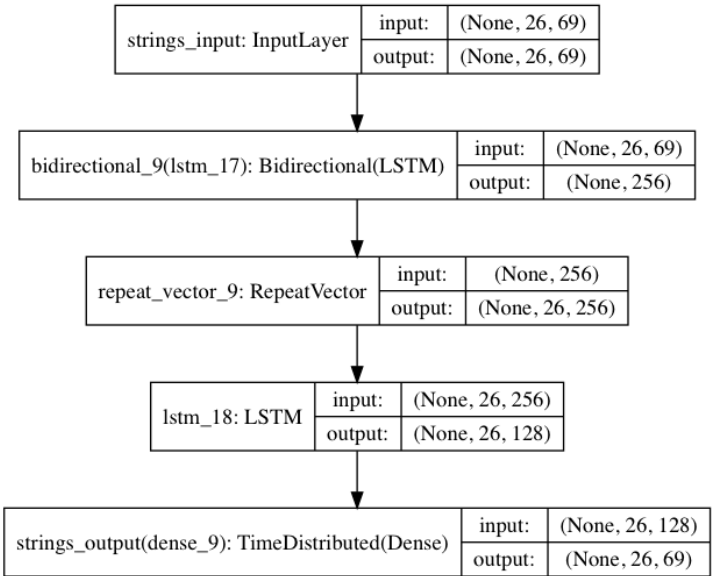


Figure 7.2 The Basic Encoder-Decoder Neural Network

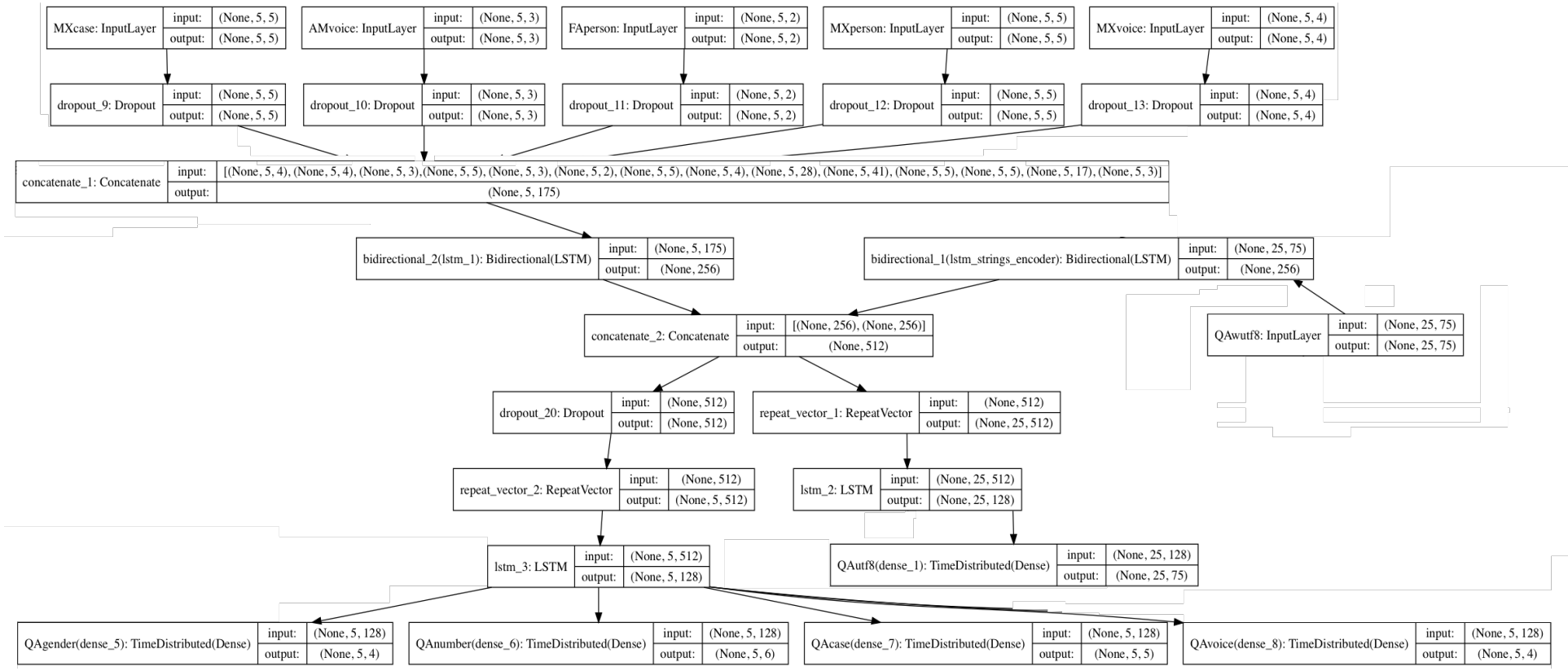


Figure 7.3 The full neural network for POS, segmentation, and morphological features

7.3.3 System Settings

All algorithms are implemented in Python using the Keras Framework (version. 2.1.4) with a TensorFlow backend (version. 1.4.1). Keras is a high-level neural networks API. TensorFlow is an open source dataflow library in Python that is used for machine learning using neural networks. It was developed by Google Brain and seems to be the most widely-used deep learning framework.

Experiments are run on a MacBook Pro laptop with a processor 2.3 GHz Intel Core i7 and 16 GB of RAM. All experiments were run on the CPU, as TensorFlow no longer supports GPU in MacOS as of version. 1.2.

7.4 End-To-End Experiments

7.4.1 Word and Morpheme Embeddings

In the following experiments, pre-trained word embeddings are used as an input to the network. Words (and subwords) are represented as continuous vectors of real numbers, a.k.a. word embeddings. This word representation is a distributional model that allows words with similar meanings to be closer to each other.

One method for this representation is the classical distributional co-occurrence sparse matrix. It counts the number of times one word co-occurred with another word in a given text. Using a decomposing technique like PCA or SGD, a word is represented in a single vector of real numbers. However, this model suffers from storing a substantial sparse matrix.

Prediction-based word embeddings (e.g. word2vec (Mikolov *et al.*, 2013)) overcomes this issue by iteratively predicting a representation of a word from its context. However, this model ignores the morphology of the words as it treats each word form independently. Another issue is that it is not generalised as it is limited to the trained vocabulary; vectors for other words do not exist and cannot be generated.

Subword Embeddings allows guessing the meaning of one word even if it is out-of-vocabulary. This models the embedding to take the morphology of a word into consideration. For example, the suffix “-borough” should indicate the meaning

of a location. Subword approaches assume that word meaning can be recreated from its components. The fastText tool (Bojanowski *et al.*, 2017), for example, does so from word character n-grams. A vector representation is generated for each character n-gram, and the word-representation is the sum of these representations.

The fastText tool is used in the following experiments. Subword embeddings are more applicable to Arabic language as it is a morphologically rich language, and the morphology plays a critical role. Besides, the number of inflected words in Arabic makes the word2vec approach insufficient without a prior segmentation. Lastly, we generate embeddings for the input word to the system in addition to the segments generated from participating taggers, so we want fastText to be consistent in each case.

The word vectors were built using fastText on a random subset of classical Arabic corpus (the texts in the corpus that were authored from eighth to eleventh centuries) that were extracted from the Shamela library (Belinkov *et al.*, 2016). The subset's total number of words is 160 million words, and the vocabulary size is 662K. The model was trained using a minimum and maximum of two and seven character n-gram lengths. The size of the word vectors is 200. The text was cleaned, and diacritics were removed. All annotation in the texts were removed, and the model was trained on the book texts only.

In Figure 7.4, we have six models: *emb* models use the trained fastText model on the input word in addition to encoding the original word as one hot vector, *only_emb* does not include the original word, *baseline* does not use embeddings at all, both for small and large training dataset. It shows that the embedding input did not contribute much in *emb* cases, even though it contributes slightly more to the model with a small training dataset. It also shows that embeddings did not represent the original word and did not encode all input word features. It was developed using the undiacritised corpus, so the lack of diacritics might cause the drop in accuracy.

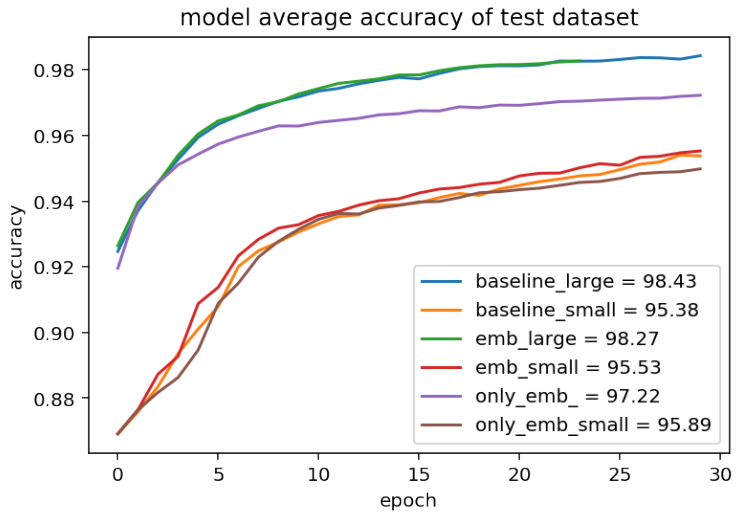


Figure 7.4 The effect of using word embeddings

7.4.2 POS Embeddings

Embeddings do not only apply word form or characters. They can be also used as an alternative way for encoding categorical Embeddings. Instead of encoding each category in one-hot encoding, these categories can be encoded jointly with the network, using embedding layers at the beginning of the network.

Feature vectors, particularly the POS tag, in one hot encoding are independent, but this is not always the case in some of the features. Some POS tags, for example, may behave similarly (e.g. nouns and proper nouns). The main benefit for this representation is in "generalisation power" (Goldberg, 2017, p. 92) which might help the network in the combination process. Also, the dense encoding reduces the computation cost of sparse vectors as shown with syntactic parsing (Chen and Manning, 2014).

We evaluate this setting in this section. In Figure 7.5, dense vectors for POS tags (POS embeddings) did not improve the overall accuracy. The effect does not show improvements in segmentation and POS tagging (the two charts on the right side). The effect is slightly noticeable in morphological features but is very limited (less than 0.015% at maximum). The embeddings might not have sufficient training data (no external resources were used), and thus the dense vector did not encode the dependency between tags to the full extent. The dense representation requires fewer parameters to be trained (1.08 millions vs 1.19 millions), and the training should be faster (in theory). However, the training only converged on the 50th epoch vs. the 39th epoch (in the baseline). The evaluation of the generated

embeddings requires an evaluation dataset of pairs of POS tagsets. So, we leave this task for the future.

The feature vectors might be helpful as well in the mapping between different tagsets. The embeddings are trained though monolingually (each feature in its space). Embeddings can be transformed linearly (or using Procrustes alignment) in a supervised approach using a set of pairs of matching tags (bilingual dictionary) or in an unsupervised approach by iteratively refining the alignment (Conneau *et al.*, 2018). This method has proven to be useful in word translation, and it might also help in finding the mapping between different tagsets. This work, however, is beyond the scope of this thesis.

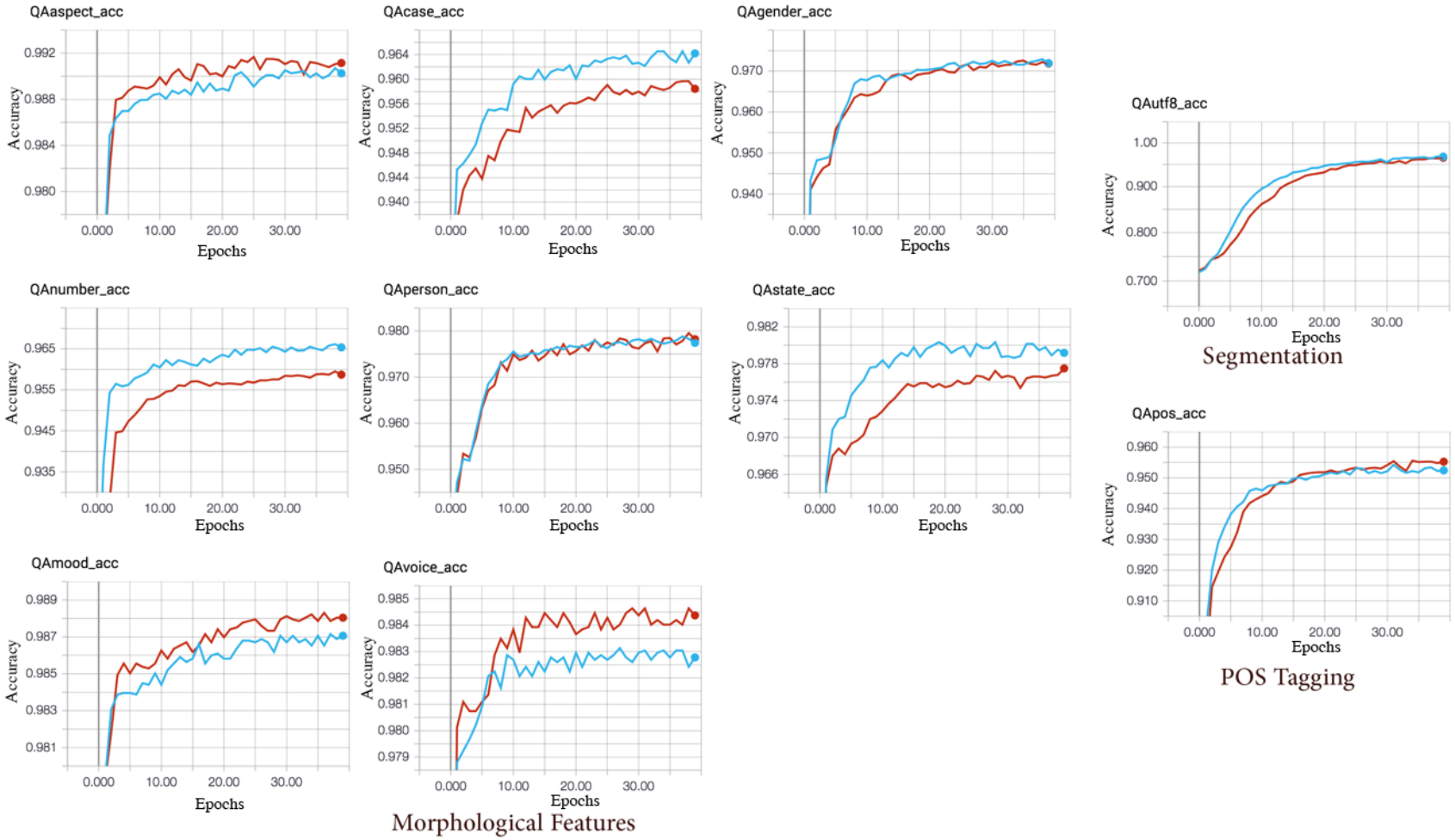


Figure 7.5 The accuracy over the training epochs using embeddings (dense vector) for POS tags (Red) vs using one-hot encoding (Blue).

7.4.3 The Effect of Training Dataset size

To remind the reader, our ensemble approach aims to help under-resourced variants by reusing existing resources. The two adaptations in this ensemble (annotation-style and domain) require a corpus of under-resourced variant annotated with the required target annotation style. However, the required size of this corpus is unknown in advance, so we evaluate our ensemble using different sizes.

The training dataset size plays a critical rule in the adaptation of the input of individual taggers. In this experiment, we show that training data size is directly proportional to the accuracy as shown in Figure 7.6. The training dataset is iteratively set to be 10, 20, .., 80% of the data, the validation split is always 10%, and the rest is for the testing dataset. The model will not be trained on the validation dataset, however, it prevents the training from overfitting the training dataset by allowing to monitor progress and providing early stopping when the validation loss is not improving. The test dataset is an entirely independent data split that we use to evaluate the model. The validation dataset in our case is not used for development or hyperparameter tunings; therefore validation and test plots should have similar patterns.

In Figure 7.6, we can see clearly that the more data is used for training, the better the accuracy. The accuracy is averaged from all outputs: POS tagging, segmentation, and morphological features. Over the training, the larger training dataset converges faster (in the number of epochs). However, the time of training for one epoch is higher with larger datasets (1 min compared to 27sec). We capped the number of epochs to 30 for time constraints.

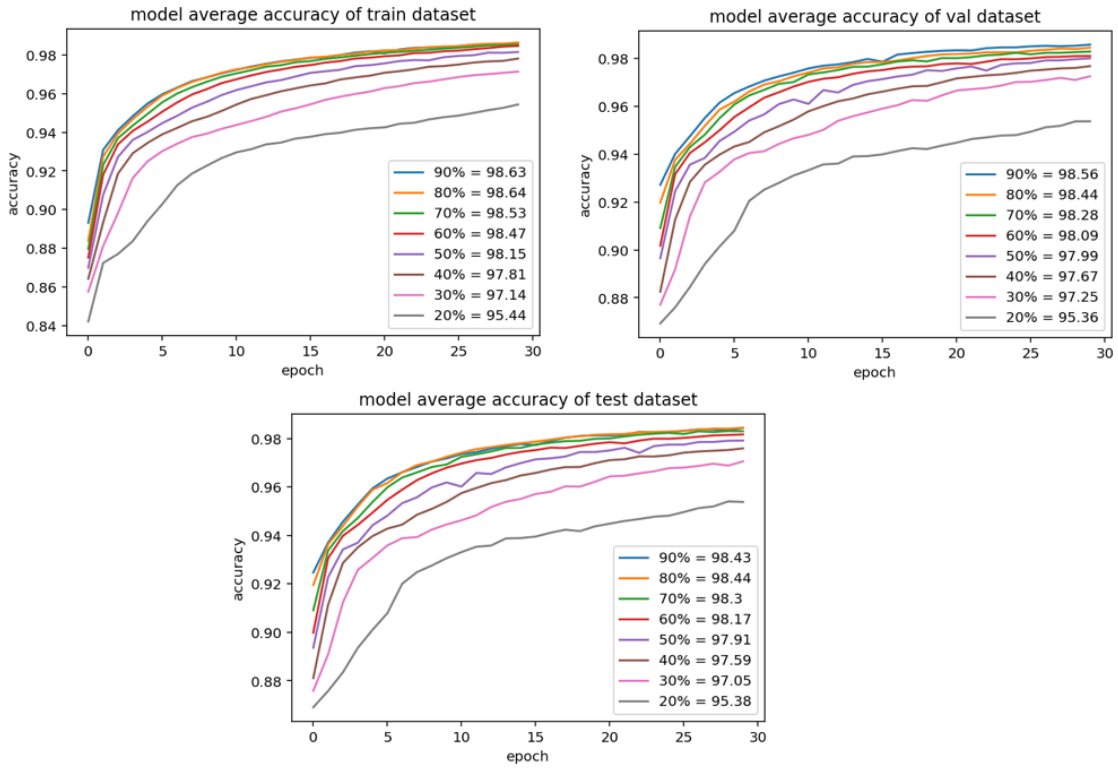


Figure 7.6 The effect of different training dataset sizes on the average accuracy.

In Figure 7.7, we can see that the segmentation accuracy primarily and POS tagging are the two outputs that suffer from small datasets. It confirms that participating taggers have different segmentation and POS tagsets, and annotated data is needed to adapt these schemas to the required schemes. As the training data get larger, the effect on the accuracy is less. There is no significant difference in many features when the training dataset size is increased.

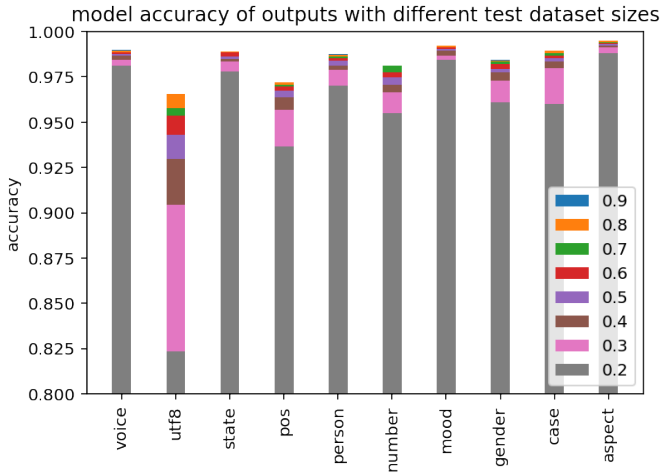


Figure 7.7 The effect of training dataset size on the accuracy of POS tagging, segmentation, and morphological features

It is notable that even with small amount of data[63], the accuracy is still high. It suggests that the annotation process for adapting low-resource languages should be iterative, and smaller dataset might be sufficient for the purpose. The amount of data required is highly dependent on the quality of input taggers and the difference between the two languages, though.

7.4.4 Different Combinations of Individual Taggers

In this section, different combinations of individual taggers are examined. This section aims to show the contribution of input taggers to the overall ensemble tagger. A baseline tagger could be created by learning from the training corpus given only the sequences of the lexical forms (no tagger's output is provided). One-tagger model can be created as well by adding the features of one tagger. The goal here is to contrast the contribution of each input tagger (or the combination of taggers) to the overall accuracy.

[63] The dataset used in all experiments is already relatively small. The whole training data is just a subset of the QAC (~30%).

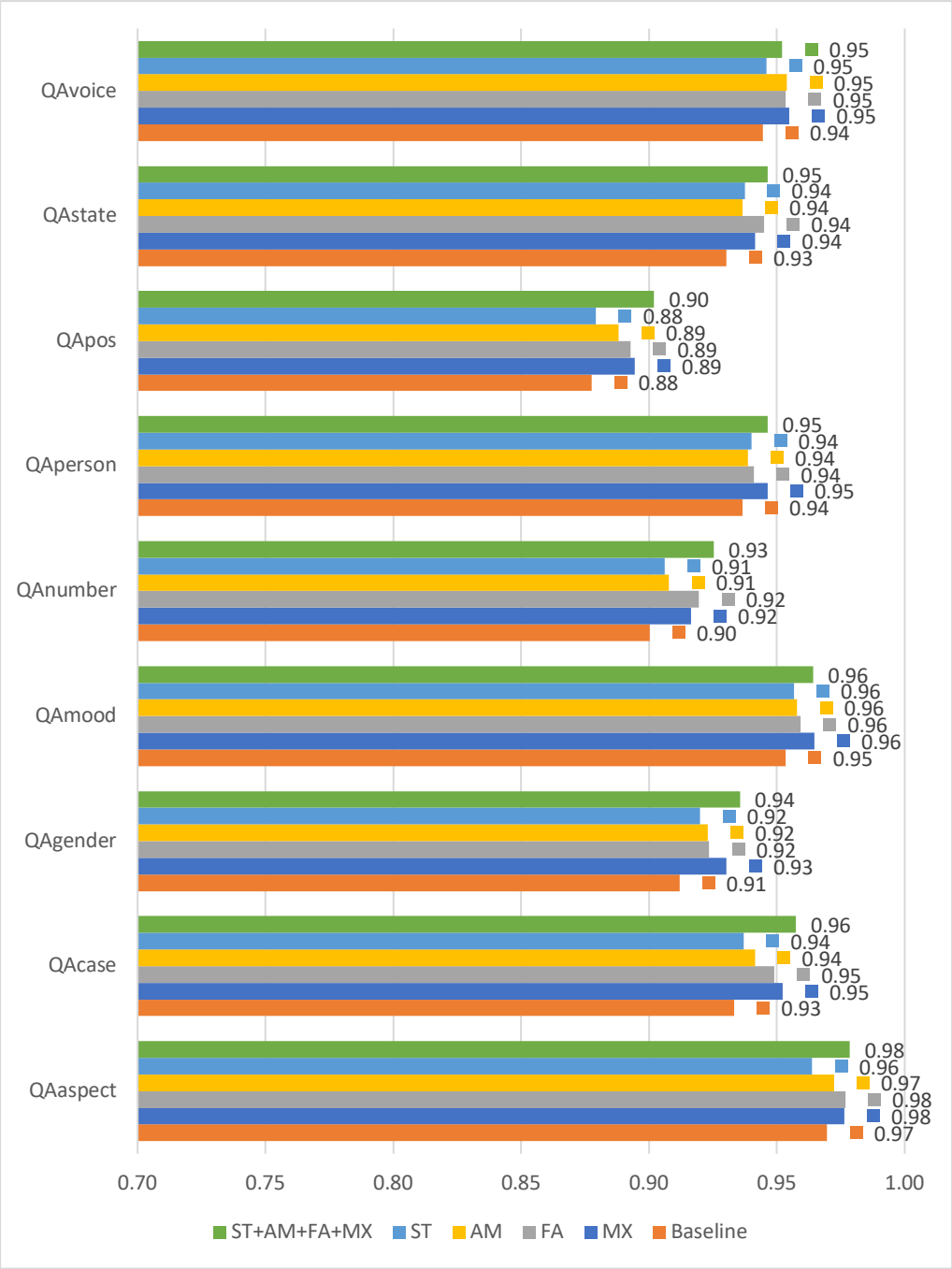


Figure 7.8 Word-based accuracy of single-tagger vs ensemble taggers

Figure 7.8 shows that the four-tagger ensemble improved the accuracy of most of the features by an average of 1%. MX tagger accuracy is the best compared to others and scored a very competitive accuracy measure.

Since accuracy is a global performance measurement, it does not give any information on the error distribution. Remember that the ensemble method exploits the differences in errors generated in each tagger, so we need to report the similarity between two taggings of the same text. Taggers have different linguistic

theories as illustrated before, but since these taggers are “adapted” to produce a similar segmentation/tagging schema, this adaption may make them more “homogeneous”. The kappa (κ) coefficient (Cohen, 1960) implemented in the Scikit-learn package (Pedregosa *et al.*, 2011) is used to compute the similarity between each pair of one-tagger models. Note that this metric (and similar metrics) do not operate on homogeneous tagging, so the effect of our adaption on error distribution may not be efficiently computed.

Since the adaptation of each tagger might make them act similarly, so we report the kappa of these taggers (for POS tagging) in Table 7.1. The table shows a high agreement between the four taggers. This high agreement might be due to the adaptation technique, the knowledge bases of this taggers, or just because MX tagger simply is superior to the other taggers (the MX tagger is more fine-grained significantly).

Table 7.1 The kappa coefficient for POS tagging between one-tagger models.

	MX	FA	AM	ST
MX	1	0.935633	0.933771	0.932335
FA	0.935633	1	0.929587	0.920686
AM	0.933771	0.929587	1	0.926287
ST	0.932335	0.920686	0.926287	1

Although the accuracy of the ensemble might not improve the overall accuracy of individual taggers significantly, the ensemble introduces a *robust* way for tagging. Researchers might not know the suitability of one tagger to their research against the others, so running an ensemble tagger does improve the accuracy over the baseline, and will adapt itself to at least the best of these taggers. The future work section suggests multiple ways to improve the performance of this ensemble such as including an Attention mechanism and stateful networks.

7.5 Segmentation Model

The accuracy of predicting the correct *segmentation* using the Encoder-Decoder model is quite low (75%). This section introduces a new modelling of the segmentation problem.

Segmentation in our case involves recovering adjusted forms, so it is not surprising that it scores the lowest. Errors in Encoder-Decoder segmentation model come from changes in letters and diacritics, with no constraints on how letters can be converted. This results in many words that are not even in the Arabic vocabulary, e.g. */fa+>aHokumu/* → */fa+>aHomu+kum/*. Some words have multiple incompatible diacritics in some letters which is not valid in Arabic, e.g. */sabiyli/* → */sabiylii/*. In addition, the task involves diacritics changes, and it makes the sequence length larger, i.e. harder to predict well.

The current model suffers from high sequence length and high possible number of characters. However, many characters (or letters) should remain unchanged. Therefore, the segmentation problem should be treated differently: it could be treated as a classification problem at the word level (like (Darwish and Mubarak, 2016)) or by predicting word clitics like (Pasha *et al.*, 2014) using a predefined set of clitics.

Pasha et al. (2014) uses a predefined set of clitics and the trained model predicts one of them. This method is not suitable for our case as it does not recover the transformations on the word segments. Darwish and Mubarak (2016) pre-processes the word form and generates a list of possible segmentations, and then the trained model will pick the correct segmentation. We follow a similar method but at the character level.

In other words, we decided to use one-to-one prediction at the character level. The problem is transformed into three problems: prediction of the character's segment position (SEG), prediction of the character's output letter(s) (LET), and prediction of the character's output diacritic(s) (DIAC). Table 7.2 illustrates the one-to-one segmentation on the way it segments two words: "it's" and إليّ /<ly~a/ *(to me)*. For each character, it should predict the target letter (LET) and diacritic (DIAC) after transformation, and the character's segment number (SEG).

Before training the model, we had to align every input word in the training corpus to its segmented version, at the character level. We used Levenshtein Distance between the two sequences (original and segmented) of each word's letters (not including diacritics). Then, diacritics are moved according to thier letter's position. The input letter and the output letter does not have to match; for example the *apostrophe* in the word "it's" is transformed into "ha", making "ha" a

new class in the classification problem. Rarely, a letter in the original form can be deleted after segmentation, i.e. transformed to an empty string.

In our training dataset, there was 52 unique letters that have been transformed into 189 different combinations of letters or an empty string and 16 unique diacritics that have been transformed into 72. The segmentation problem assigns to a letter its position which can be one of 16 possible positions. Please note that if one letter (e.g. /y/ in our example) is assigned the segment "1+2", then the segment "1+2" is one possible class of the SEG classification problem, not two segments "1" and "2".

While the accuracy of predicting the SEG is 95.73%, LET 97.99%, and DIAC 96.26%, the accuracy that one word had a complete successful segmentation SEG+LET+DIAC, i.e. letters and diacritics are transformed correctly, and each letter is assigned the correct segmentation, is 92.16% (see Table 7.3). Only the latter accuracy can be compared to our previous model of Encoder-Decoder (75%) and it shows a great improvement in the accuracy.

The result of our model may not directly be comparable to other works in the literature, due to different segmentation schemas. Mohamed's (2018) work on the development of religious corpora scored better accuracy on SEG prediction[64]: 96.32% (compared to 95.73). However, as mentioned by the authors: "The real merit is in the ability of the classifier, and its features, to go beyond what it is trained on". The accuracy of the segmentation of OOV in Emad's work is 81.56%, but it is 86.80% in our case (17.96% of words are OOV in our case and 16.4% in Mohamed's case).

[64] Mohammed assumed that each character does not undo the assimilation and instead keep the conventional written form. That is, in cases like "it's" is not recovered to "it has *or* it is". Instead, the only goal is to mark each letter with its proper segment. This is similar to QAC original settings, which lead to empty-form segments. It is similar to our SEG problem except that one character cannot be assigned to different segments; so, the segmentation classes are equal to the maximum segments number.

Table 7.2 One-to-one Segmentation.

Input word	**it's**			
Target Segmentation	it has		it is	
INP	LET	SEG	LET	SEG
i	i	1	i	1
t	t	1	t	1
‘	ha	2	i	2
s	s	2	s	2

إليَّ /<ly~a/				
إلى + ي				
INP		LET	DIAC	SEG
<	i	<	i	1
l	a	l	a	1
y	~a	Y+y	o+a	1+2

Arabic Word

In future work, we might improve the current model to include more contextual information, as it is currently work at the word-context only, although this extension seems not to improve Mohamed's work significantly (at most 0.2%). The current EN model (unlike other models) does not use information on how input taggers have segmented the text, i.e. it is not an ensemble of these taggers. So, another option is to encode how they are segmented and use an ensemble of these inputs (maybe each sequence associated with an LSTM layer).

Table 7.3 The overall, and out-of-vocabulary word-level accuracy of segmentation (SEG), letter transformation (LET), and diacritic transformation (DIAC).

	Overall	OOV
SEG+LET+DIAC	92.16	79.86
SEG+LET	94.65	84.56
SEG	95.73	86.80
DIAC	96.26	92.17
LET	97.99	95.97

7.6 Ensemble Approaches Comparison

This research has presented mainly two approaches: Pipelined and End-to-End ensembles, with two primary methods for morphological alignment: morpheme-based and form-based methods. The morpheme-based method tried several ways for alignment including rule-based and supervised ways of alignment. However, these experiments were done over two years and there are several mismatches between experiment factors. This section presents a re-implementation of these approaches on a common ground and same platform (Neural Network

implementation using Keras with TensorFlow backend, same training and test datasets). The code of all experiments is published at the author's Github page[65].This section as well summarises the differences between these approaches and analyses the error generated from each model.

7.6.1 Models

This section compares between four proposed models: pipelined morpheme-based rule-based ensemble (RU), pipelined morpheme-based supervised-alignment ensemble (SP), pipelined form-based ensemble (CH), and end-to-end ensemble (EN), as shown in Figure 7.9. These four models had the best scores in previous evaluations. Table 7.4 presents a summary of the differences between the four models.

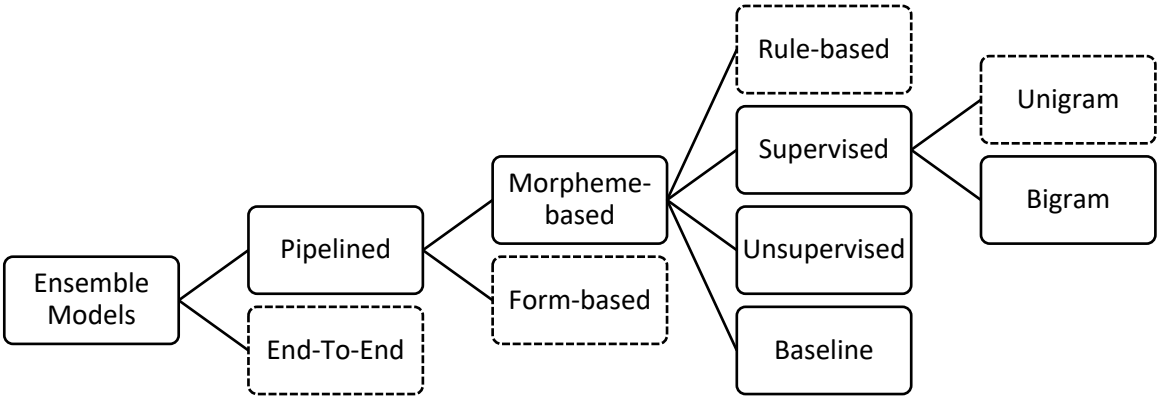


Figure 7.9 The hierarchy of presented ensemble models. Only marked models are included in this section because they scored the best accuracy in previous evaluation.

[65] https://github.com/aosaimy/sawaref-rnn

Table 7.4 Summary of differences of presented models.

	RU	SP	CH	EN
Definition	An ensemble that uses rules generated from *experts* to map morphemes of different taggers. The ensemble uses this aligned input one by one to predict the morpheme label(s).	An ensemble that uses rules generated from an aligned dataset to map morphemes of different taggers. The ensemble uses this aligned input one by one to predict the morpheme label(s).	An ensemble that uses the word form to map the form characters (with their labelling information) of each input tagger. The ensemble uses aligned character-based information one-by-one to predict the character label(s). Character labels encode morpheme boundaries.	An ensemble that utilises an Encoder-Decoder network to encode the sequences of each input tagger, concatenate these encoding, and decode the results into a new sequence of morphemes.
Sequence labelling type	One-to-one: Each morpheme (or character) is labelled individually with respect to the context. Unlike seq-to-seq models, the final segmentation is restricted to input form length (in the CH model) or input segmentation models.			Seq-to-seq: A sequence of word morphemes is encoded, then decoded to predict another sequence of labels.
Pipelined vs Joint	Fully Pipelined: Alignment precedes tagging, segmentation is the result of voting between aligned morphemes.		Partially Pipelined: Alignment precedes tagging, segmentation is jointly predicted with tagging.	Fully Joint: Alignment and segmentation is done jointly in the embedding model.
Error propagation	Pipelined models suffer from errors generated from previous steps. Errors generated from prior alignment methods result in lower consistency of input data.			Fewer changes to input data are required.
Alignment Tuning	Pipelined models tune previous steps on an evaluation dataset of aligned inputs. This tuning is abstracted from the final goal: the tagging results of the ensemble.			Tuning of network weights is done at the same time as training the ensemble.
Alignment Evaluation Dataset	Morpheme-based models require a dataset that is aligned on the morpheme level. Morpheme boundaries are not explicit, and this alignment is prone to errors. The evaluation is not necessary for the ensemble, but it can help to spot errors.		No prior dataset is required. However, some rules for adapting mismatches might be required.	No alignment is required.
Alignment method	Rule-based requires a mapping between tagsets, which in turn	The supervised method requires an aligned dataset to	Alignment is form-based using string similarities algorithms. However, it	The alignment does not need any human intervention.

	requires a thorough understanding of both tagsets.	the morpheme-level from which the alignment rules are generated.	assumes that the form for each output is the same, or some adaptation is required.	
Dropping some labels	Morpheme-based models suffer from the limitation on the output segmentation. They are limited to the intersection between input and target segmentation. Finer segmentation in the output will not be reproduced.		Mapping to character-level solves the problem of morpheme-based models.	End-to-end model mimics the output sequence of the training dataset and does not have a similar limitation.
Same form output	Morpheme-based models vote for one input's segmentation, which does not guarantee adapting output segmentation to the target segmentation.		Form-based assumes no additional character is inserted in the form; i.e. form adjustment is not supported.	The end-to-End model mimics the output sequence of the training dataset.

7.6.2 Implementation

The (re)-implementation of these models is done using the latest version of the Keras package (v2.1.4) with a backend of the latest version of TensorFlow (v.1.4.1). The end-to-end neural network is the same as reported earlier in this chapter.

Pipelined models were implemented with a single LSTM layer that returns states of each timestep, which ensures having a one-to-one model.

All networks are multi-output networks. The set of outputs are 9 outputs: POS tag, and eight morphological features[1]. Instead of training each output individually, the network benefits from sharing layers by utilizing information from other morphosyntactic features. We adopt a multi-task approach similar to approaches done by Søgaard and Goldberg (2016) and Inoue, Shindo and Matsumoto (2017).

The network (illustrated in Figure 7.10) starts with a set of input layers that represent eight features in the four input taggers (resulting in 36 features). In addition, one input layer that represents the lexical form is defined. It is either an inline one-hot encoding of the character (in form-based models) or a predefined embedding of the morpheme (in morpheme-based models), using the FastText model (see Section 7.4.1).

These inputs are concatenated into a single layer that is fed into a bidirectional LSTM layer with 256 hidden units. Unlike seq-to-seq models, the full sequence is returned (vs. only the last output), which ensures having the same timesteps in following layers.

The output of the LSTM layer is supplied to each required target feature. In each feature, a dense layer for every temporal slice of the input (using TimeDistributed wrapper) is applied to predict the final values of outputs.

[1] The set of features used at this comparison is the nine features (i.e. tagging). Three more segmentation related outputs (SEG, LET, and DIAC) are only used and evaluated for EN model. Segmentation is not included in this comparison because the segmentation problem does not utilize the segmentation results from input taggers, because they adhere to different segmentation schemas.

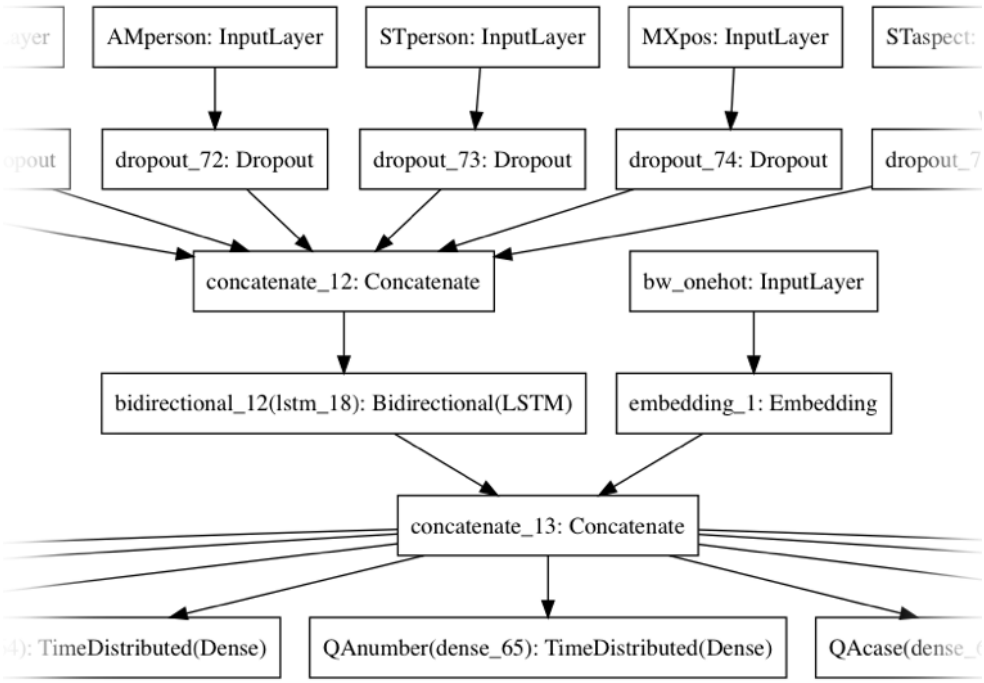


Figure 7.10 The network used for pipelined models. The input consists of a long list of features (8 features x 4 taggers), and output includes all target features (the complete lists are not shown). Character-based ensemble uses one-hot encoding of the character letter (bw_onehot), while morpheme-based ensemble uses an embedded vector of the morpheme form.

The model used adaptive moment estimation (ADAM) (Kingma and Ba, 2014) as the optimizer, instead of the classical stochastic gradient descent procedure. Unlike stochastic gradient descent, ADAM does not maintain a single learning rate; instead, it adaptively update the learning rate associated for each weight in the network (between batches). It is widely-used in the recent research for its efficiency in achieving optimized network weights in a shorter time. The details are not relevant to the research.

The set of outputs are evaluated in each epoch, and weights are updated accordingly. Because each one of the outputs is a categorical feature, we used categorical cross-entropy as the cost (or loss) function to measure the performance of the classification. The output of each final node (after activation[2]) is a probability of each class. For example, the node "QAgender" outputs a probability for each

[2] The actual output of the last dense layer is actually a set of numbers which are "softmaxed", i.e. each output is assigned a decimal probability (between 0 and 1), where all probabilities add up to exactly 1).

class: “male”, ”female”, “irrelevant”. Categorical cross-entropy is a generalization of log loss to multi-class classification problems, and it quantifies the difference between actual and prediction distribution. The loss increases as the predicted probability diverges from the actual label. It is defined for multi-class classification problems as follows:

$$\text{CrossEntrpy} = -\sum_{l=1}^{L} y_l \log(p_l)$$

Where L the set of labels, y_i is the actual label (either 0 or 1), and p_i is the predicted probability.

7.6.3 Padding Sequences

TensorFlow operates on tensors where the network is a directed acyclic graph (DAG). However, TensorFlow requires the definition of the graph before a model can run. The sequence length must be fixed (usually the maximum length of training sequences). Shorter sequences are usually padded with zeros to fill the fixed-length requirement.

Without a careful treatment of output, the padding requirement may lead to a biased cost function (the predicted outputs of padded elements may bias the loss). This is especially relevant in this comparative evaluation as the padding is different between the four approaches.

Two approaches are used to solve the problem: masking and custom sample weights. The masking layer masks timesteps that are equal to a certain value from all downstream layers. However, when a sequence is encoded using LSTM, the masked timesteps will no longer be effective to downstream layers. The sample weights technique allows the definition of custom weights for each sample, including its timesteps. For training purposes, the weight of padded timesteps is 0.05, but in validation is 0. This configuration allows the training process to not completely ignore padded timesteps (as they should be marked as padded), but give more priority to other timesteps. Padded timesteps in the validation dataset are ignored when computing the accuracy of the prediction on the validation dataset (by zero-weight configuration) which isolates the evaluation metrics from any padding side effects. This configuration explains the high difference in training accuracy compared with validation and test datasets (see Figure 7.11).

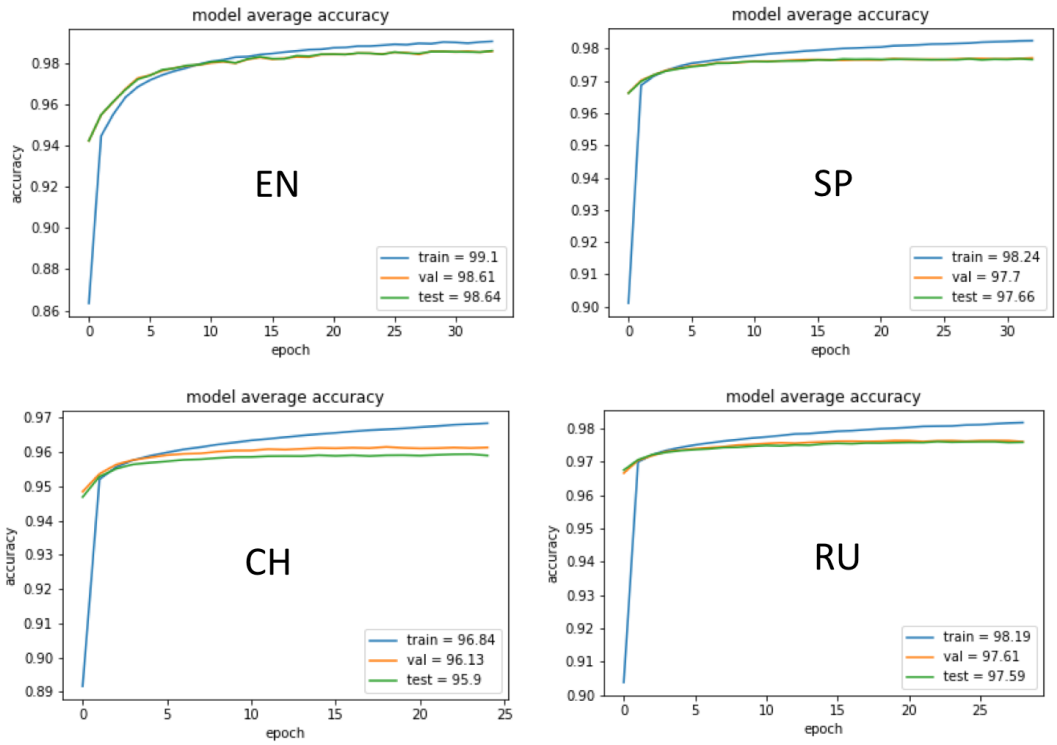


Figure 7.11 The training, validation, and testing sample-level accuracy of each approach over the training epochs.

7.6.4 Evaluation

Accuracy is the most used metric in the literature (Paroubek, 2007), so it is used to report the ratio of the number of words/segments that are correctly tagged over the total number of word/segment forms tagged.

Overall *precision and recall* are meaningless since every morpheme can be tagged with exactly one tag (i.e. ambiguity =1). They will both just equal the accuracy measure, as the tagger and the reference datasets are one-tag based.

Accuracy of the models is reported in two ways: sample-level and word level. Sample level is the one used internally for defining the loss (cost function), but the sample is different between approaches (e.g. morpheme-level vs character-level). Therefore, the word-level is the metric used to compare different approaches, i.e. the portions of words that are predicted correctly.

Word-based accuracy is reported for each output (see Figure 7.12 and Table 7.5). The best scoring model is the end-to-end model for all outputs features. A very similar pattern between test and validation can be seen.

Table 7.5 The accuracy of each output for all four proposed ensemble models

	EN	SP	RU	CH	(Marton, Habash and Rambow, 2013)[3]
Aspect	97.85%	95.12%	94.41%	92.53%	99.1%
Case	95.75%	82.68%	82.82%	79.60%	86.3%
Gender	93.56%	88.46%	88.95%	83.09%	98.6%
Mood	96.42%	94.09%	94.23%	90.65%	98.6%
Number	92.53%	83.49%	83.62%	77.49%	99.2%
Person	94.63%	91.50%	91.72%	87.20%	99.1%
POS tag	90.20%	85.32%	85.64%	81.79%	N/A
State	94.63%	88.86%	89.26%	84.25%	95.6%
Voice	95.21%	92.04%	91.72%	89.35%	98.9%
Aggregate	74.87%	55.41%	55.01%	45.97%	N/A

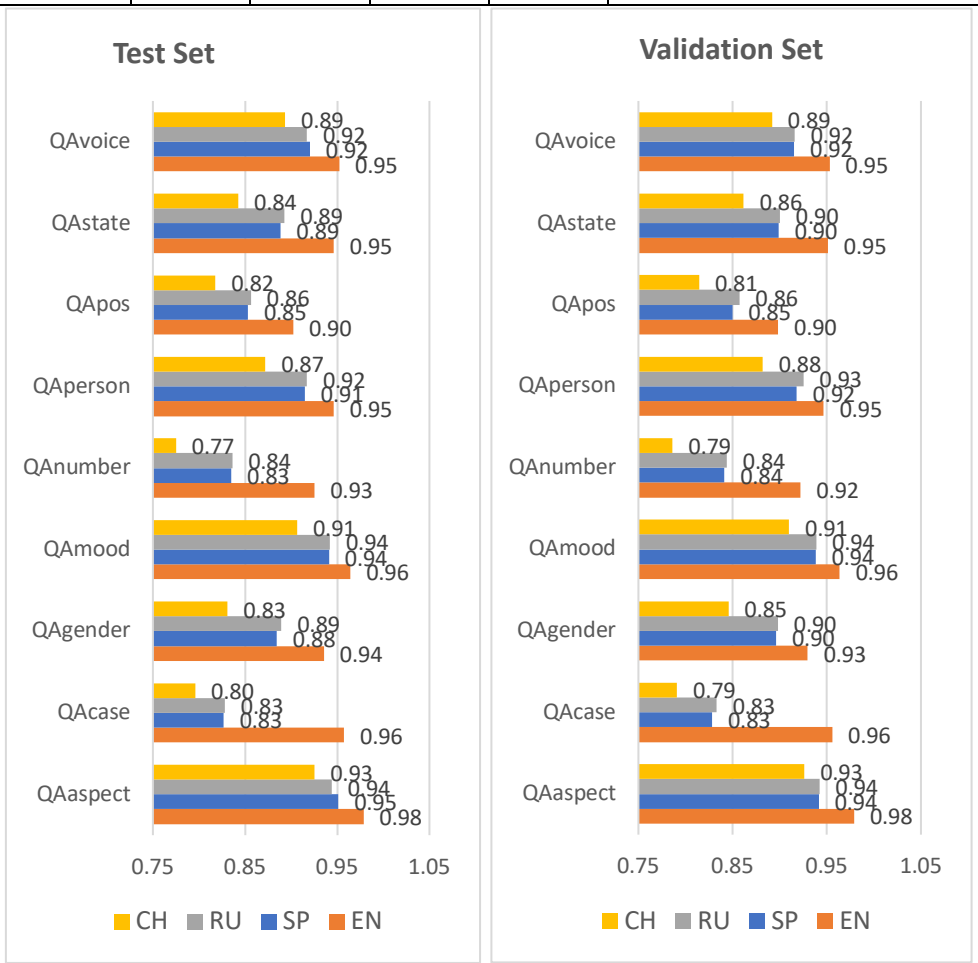


Figure 7.12 The word-based accuracy of four ensemble approaches that predict validation dataset outputs.

[3] Their experiments used Penn Arabic Treebank, i.e. MSA Arabic, which is 19.3 times larger training dataset. The accuracies of functional gender and number are used instead of form-based ones.

Sample-based accuracies are higher than word-based accuracies in all approaches, see Figure 7.13. Word-based accuracy marks a word as correctly predicted if *all* of its samples are correct.

Word-based and sample-based accuracies show that number, gender and case features in addition to POS tag scores the lowest. Number and gender are two functional features which makes their prediction more complex. These two features are under-specified in the annotation of the QAC (Please see discussion in next chapter and Table 8.1 on page 186). They are underspecified for nouns, proper nouns and adjectives with different rates; for example, the number is only specified for 36% of nouns.

The case feature is a known problematic feature. Some approaches in the literature ignore it in parsing (Marton, Habash and Rambow, 2013), although it was the most helpful feature in the gold standard in their experiments.

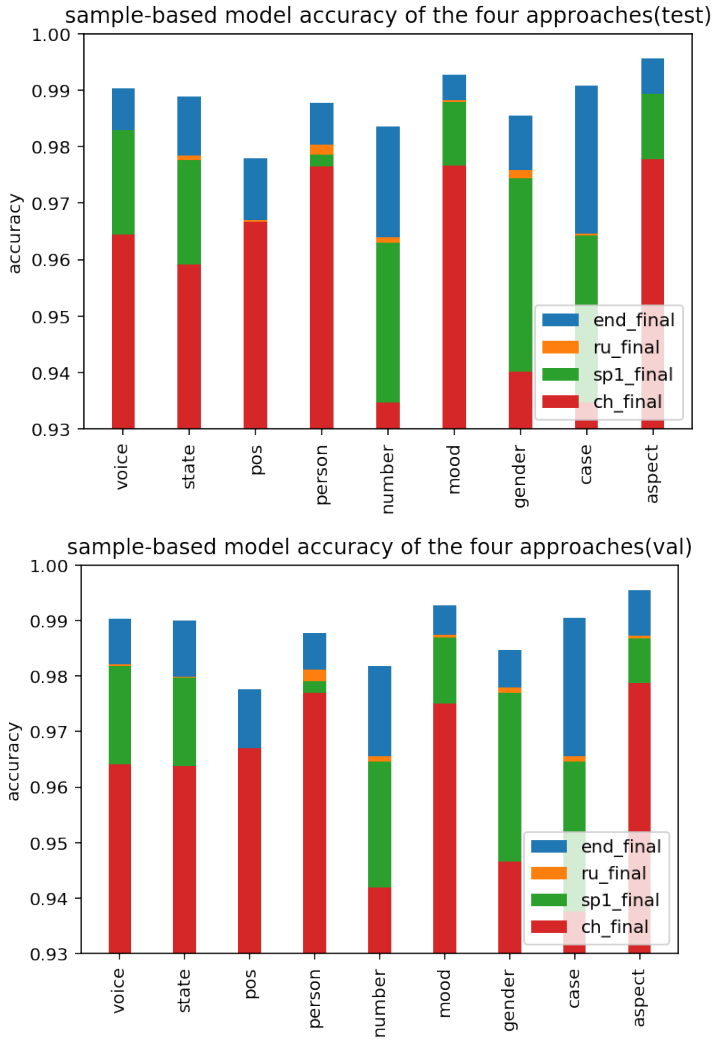


Figure 7.13 Sample-based model accuracy of the four approaches

The *real* OOV (out-of-vocabulary) metric is not computable because it is not possible to determine what are the OOV words in the input taggers (as we have no access to their training dataset). However, *ensemble* OOV words that have not appeared in *our* training dataset constitute about 17.96% of words. The OOV accuracy for each output is provided in Table 7.6.

Table 7.6 The overall accuracy and Out-of-Vocabulary accuracy.

	EN	EN-OOV	Drop Difference
number	92.53%	72.93%	19.60%
person	94.63%	78.08%	16.55%
voice	95.21%	80.98%	14.23%
gender	93.56%	79.42%	14.14%
case	95.75%	83.89%	11.86%
aspect	97.85%	86.58%	11.28%
mood	96.42%	85.23%	11.19%
PoS Tag	90.20%	79.64%	10.56%
SEG	95.73%	86.80%	8.93%
state	94.63%	88.14%	6.49%
DIAC	96.26%	92.17%	4.09%
LET	97.99%	95.97%	2.02%

The *aggregated* accuracy is the percentage of words that had a completely correct tagging in all output classes. It is 45.97%, 55.41%, 55.01%, and 74.87% for CH, SP, RU, and EN approaches, respectively.

It is clear that the EN model surpasses other models in all of the tests. Several improvements can be made to this model specifically and for other models. The next section discusses and analyses the ensemble errors in prediction and suggests actions for future improvements.

7.7 Error Analysis

This section discusses the errors produced by the ensemble analysers. Most discussion will be on the EN model as it scored the best in all features.

The aggregated accuracy of the EN model is 74%. The remaining 26% of words are incorrectly tagged (i.e. has at least one error in their tagging, e.g. *male* is incorrect). The total number of outputs in the EN model is 12 which includes POS

tag, eight morphological features, and three segmentation-related features (SEG, LET, DIAC). Figure 7.14 shows the percentage frequency distribution of the incorrect words. We can see that words with a single error makes the majority of the incorrectly labelled words. This might suggest for future work that using a lexicon (or a morphological analyser) filtered or ranked based on the prediction might fix some of the erroneous outputs. The source of the error in single-error words is illustrated in Figure 7.15.

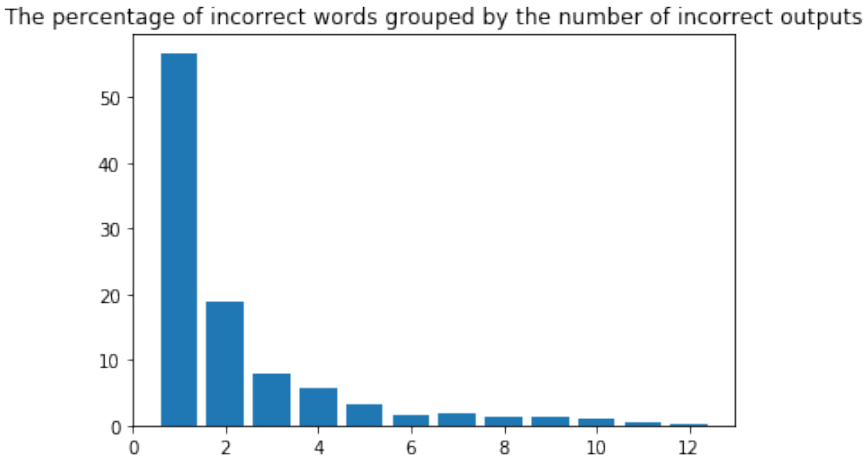


Figure 7.14 The percentage word frequency that has *n* prediction errors.

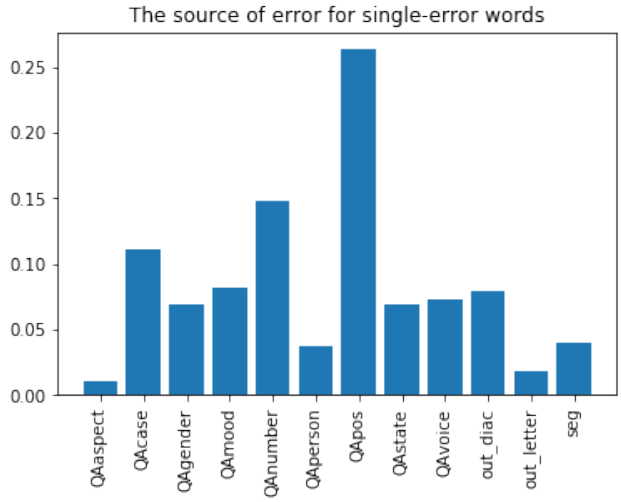


Figure 7.15 The type of error for words that have a single error

7.7.1 POS Tagging

POS tagging word accuracy (90%) is significantly better than other approaches.

Most of the errors come from predicting ADJ (adjectives) as N (nouns), RES (restriction) as EXP (exception), PRO (prohibition) as NEG (negation), REM (resumption) and CIRC (circular) as CONJ (conjunction), or vice versa. Please see Figure 7.18 for full confusion matrix. Please see Section 8.2.1 for more details about the similarity between these tags.

The overall POS accuracy does not show the performance of tagging a specific tag. Many tags are naturally under-sampled in the QAC tagset. Figure 7.16 and Figure 7.17 show the F_1 score for each tag and the frequency of each tag. The lowest scored tags suffer from ambiguity at the word level, inconsistent/incorrect tagging in the reference corpora (see 8.2.1), and under-representation in the training/validation corpus. The figure as well shows the imbalance problem where some classes are under-sampled. Two ways are usually used to handle the imbalance: oversampling and custom loss function. Oversampling usually is hard as the samples in text classification are related. The other option is to give higher weight to samples from a certain class, which results in paying more attention to these classes. However, tuning these hyperparameters (weights) requires a development set and we will leave it for future work.

A reduced tagset obviously is one option to improve the accuracy, but this should be done based on the needs of the target downstream application. Using the universal dependencies tagset (UD), a coarser tagset mapped to the QAC tagset in Table 8.8, the accuracy improved to 92.96%. Another option is to jointly learn the prediction of segmentation, POS tagging, and dependency parsing (like (Zhang *et al.*, 2015)), which shows a significant improvement on OOV words. Options regarding our model include using stateful NN and custom class weight.

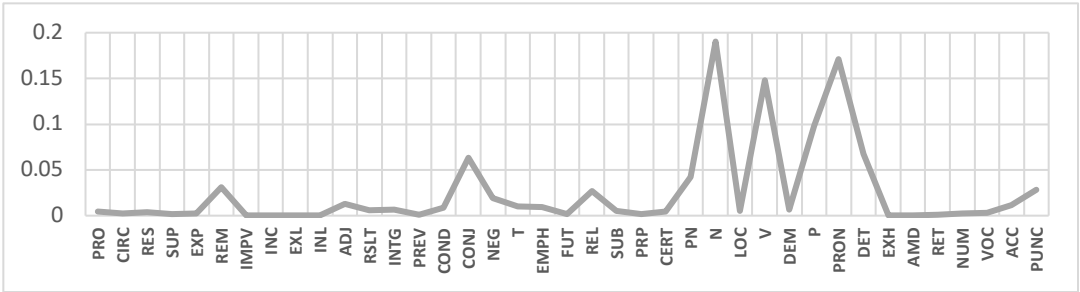


Figure 7.16 The frequency of each POS tag.

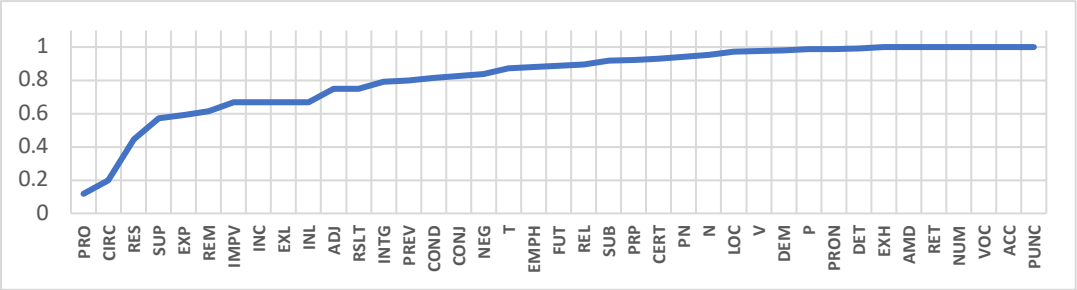


Figure 7.17 The tagging F1-score of each POS tag.

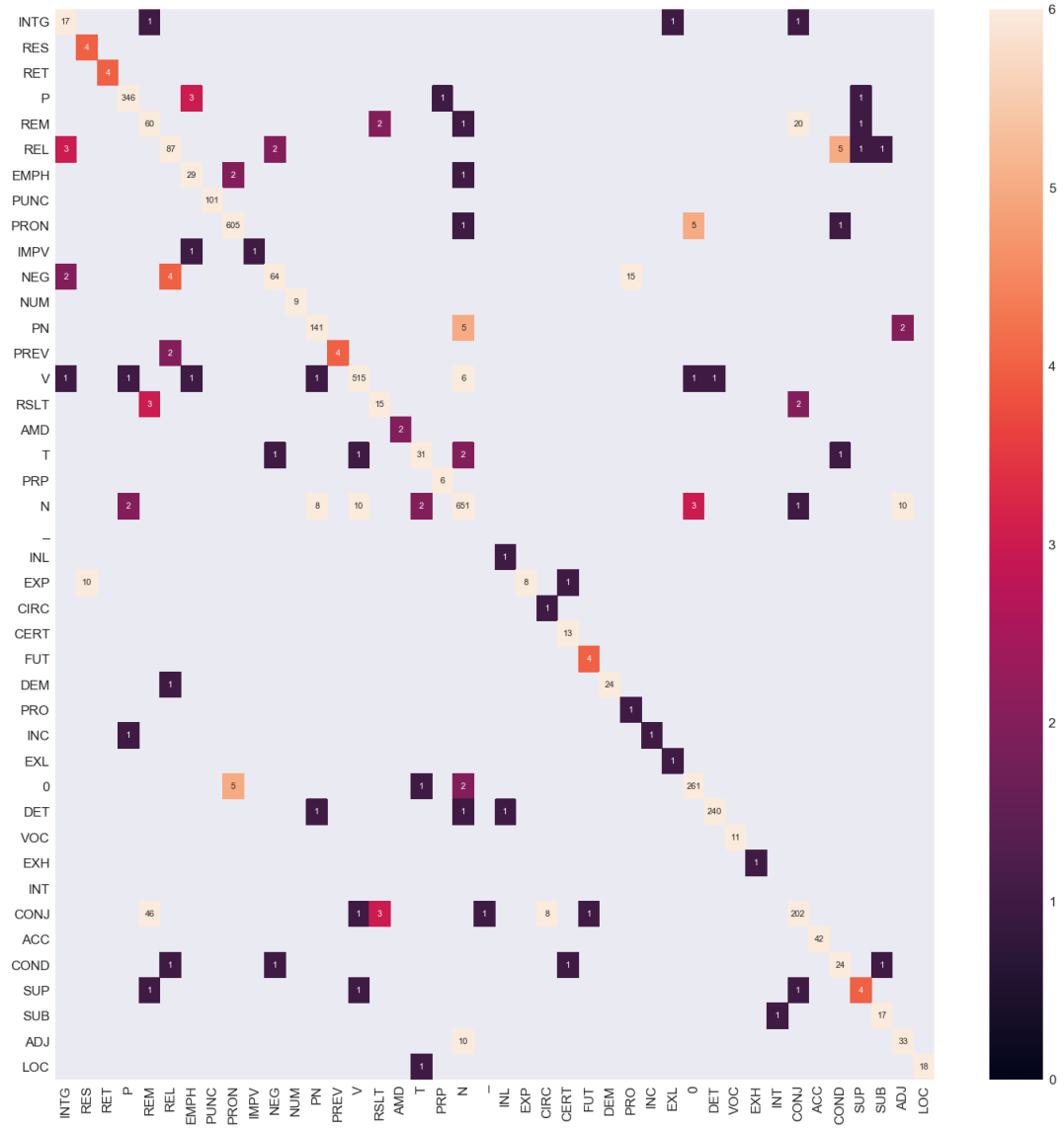


Figure 7.18 The confusion matrix of POS tagging (EN model).

7.7.2 Morphological Features

Morphological features can be divided into two categories: functional and form-based. Form-based features scored better results as all input taggers are form-based. The choice to functionally tag the number and gender features makes its

prediction more complex. Morphological features infrequently predict a value that is incompatible with the POS tag. Instead of predicting each morphological feature independently, it might be beneficial if the predicted POS tag is used as input for predicting the feature's value. Some errors in morphological features come from missing values in the reference corpora or inconsistency between the two datasets (see Section 8.2.1). The presence of diacritised input text seems to improve mood and case accuracies compared to previous works in MSA. Some features are only related for specific POS tags, and errors come from incorrectly tagging their POS tags.

7.8 Comparative Evaluation

The accuracy of the ensemble taggers can be compared with other related POS taggers. This comparison is, however, challenging because of different standards in annotation and training and testing datasets.

In regards to tagging classical Arabic text, Alashqar (Alashqar, 2012) used six different taggers (Unigram, Bigram, Trigram, Brill, HMM, and TnT) trained on the Quranic Arabic Corpus. The best word-based accuracy achieved was 80.4% using the full QAC tagset. His result might not be directly comparable since it uses an older version of QAC which is word-based. In addition, the training/testing splits are not specified, except the ratio of training to testing. The reported word-based accuracy might be comparable cautiously as there might be some edits to the newer versions of QAC.

Mohamed (Mohamed, 2018) recently published a new classical Arabic fine-grained corpus of 60k words annotated with PATB-like tagset. The tagset used is complex and has 133 segment-level tags and 949 word-level *compound* tags. Using TiMBL toolkit, a memory-based learning toolkit, the accuracies of full automatic segmentation and POS tagging on development and test datasets are 89.8% and 87.8%, respectively. These accuracies are not directly comparable as settings such as test and training datasets, tagset, segmentation are different.

In regards to an Arabic heterogeneous ensemble, Alabbas (Alabbas, 2013) reported a high accuracy (99%) of an ensemble tagger that combines AMIRA, MADA, and maximum-likelihood taggers to predict the tagging of MSA text. The work is not directly comparable to our results as his training/test datasets and tagset

are different; the author used an ensemble tagger on Modern Standard Arabic, which means that only annotation-adaptation is required.

Comparing with input POS taggers might be appealing. POS taggers (MX, ST, AM, FA) used different tagsets when reporting their accuracy (see Chapter 4), which make a direct comparison with our ensemble system unfair. They used various data splits which makes them even incomparable among themselves. Most of them used data splits from the PATB, but the data split is different: the test dataset of MX is 10% of Part 3, and of AM and ST are 10% of each part; and Farasa used a WikiNews corpus for testing as "Testing done on a subset of the ATB is problematic due to its limited lexical diversity, leading to artificially high results" (Darwish and Mubarak, 2016, p. 1). We do not have access to the PATB as it requires an expensive Linguistic Data Consortium (LDC) membership.

However, adapting their results to match our test dataset (with QAC annotation style) is an option. This is done by training morpheme-based or end-to-end models that have only one input tagger. These models that has only the tagger are no longer ensembles, but they are adapted to label using the QAC annotation style. The *morpheme-based* model of adapted MX, ST, AM, and FA taggers correctly predicted 83.86%, 84.78%, 83.28%, and 80.22% respectively (see Section 6.7). The *end-to-end* model of adapted MX, ST, AM, and FA taggers correctly predicted 89.44%, 87.92%, 88.81%, and 89.26% respectively (see Section 7.4.4). This should as well give a rough estimation of how likely an input tagger can help our ensemble tagger.

Another option is to compare the ensemble results to off-the-shelf taggers. This is only possible to taggers that are not customized to their own tagset or segmentation schema, e.g. our input taggers. In addition, we want to compare with taggers that support lexical form adjustment at the morpheme level (word form assimilation). We used a state-of-the-art tagger that is built for usage with Universal Dependencies schema: namely UDPipe (Straka and Straková, 2017). Default settings for training are used: 100 epochs for segmentation, 20 for tagging, 0.1 dropout, 0.005 learning rate, 50 batch size. We do not use the lemma as a feature or ask the model to predict it for fair comparison. Table 7.7 shows the F_1 measure of UDPipe and the accuracy of EN-ensemble. Please note that UDPipe does not use accuracy like our work, instead they used F_1 measure as described in CoNLL 17 shared task (Zeman *et al.*, 2017). In UDPipe and the shared task, the number of

nodes (word segments) differ between the gold standard and system output. So, the precision is the correct tagging nodes percentage of system nodes, while the recall is the correct tagging node percentage of gold nodes. In our case, the nodes are words whose alignment are maintained at evaluation. In almost all accuracies produced by the UDPipe toolkit, the ensemble scores better except in morphological tagging.

Obviously, input taggers are trained on a larger training dataset than our training dataset which is exploited by the ensemble tagger.

Table 7.7 The accuracy of UDPipe vs. EN ensemble

	UDPipe F_1	**EN-Ensemble Accuracy[4]**
Segmentation	88.50%	92.16%
UD POS Tagging	83.55%	93.44%
X POS Tagging	82.38%	90.20%
Morph. Features	76.67%	76.16%
Overall Tagging	73.16%	77.16%

7.9 Conclusion

This chapter introduced a novel method for ensemble tagging by using an encoder-decoder architecture to perform a sequence-to-sequence learning. In the first part of this chapter, different configurations of this model are introduced. It shows that the sequence-to-sequence method clearly surpasses the one-to-one modelling of the problem. POS embedding did not improve or worsen the accuracy significantly although it reduced the parameters of the network, thus the training speed. Word embedding in Arabic needs to care more about optional characters (diacritics). Embeddings trained using undiacritised texts do not improve the overall accuracy as well. Intuitively, the larger training dataset, the better adaptation and tagging. However, an acceptable accuracy can be achieved with comparably small datasets, the usual case of under-resourced languages. The ensemble tagger introduces a robust method for tagging and it can either match or improve the accuracies of one-tagger models. Re-using other tools improves the accuracy over the baseline, which makes these tools a valuable linguistic resource regardless of heterogeneity. These results suggest that researchers should consider re-using existing methods although their required tagset/segmentation schema is different.

[4] The accuracy in our case equals the F1 measure because recall and precision are equal since the system tags and gold standards tags are equal by definition.

In the second part, a re-implementation of the previous methods with a neural network is done to ensure similar and fair comparison. An end-to-end model of the problem surpasses previously proposed pipelined methods in all accuracy measures. Not only does it have the freedom from manual feature engineering, the end-to-end model is superior to other models almost in all classes. It suggests that manual alignment or mapping is not needed with the existence of a large enough training dataset.

The error analysis section discussed the errors generated from proposed models in detail. It suggested several guidelines for future work for improving the overall accuracy and robustness. For future work, the attention mechanism may be used to pay more attention to some features or timesteps. In addition, encoding one tagger's outputs should be contrasted with the current encoding of all input taggers. A stateful network should be used in production stages as the current network discards the states between batches, thus losing the context information. One direction for improvement is the integration of lexicons, and it could easily be achieved in several ways, e.g. by concatenating a vector that represents the sum of one-hot encoding of the lexicon results, as shown beneficial by Inoue, Shindo and Matsumoto (2017). Another direction for improvement is combining presented ensembles, i.e. ensemble of ensembles, although this approach should be evaluated first to ensure that they produce different errors.

The next part discusses the annotated data used in this thesis and introduces a new linguistic resource for the Hadith genre of classical Arabic. It also presents a novel tool for annotation that aims to speed up the tedious annotation process while maintaining the consistency and accuracy.

PART III

Sunnah Corpus Annotation

8 SUNNAH ARABIC CORPUS ANNOTATION: DESIGN AND METHODOLOGY

Chapter Summary[5]:

Sunnah Arabic Corpus is an annotated linguistic resource that consists of 144K words of the Hadith narratives (an utterance attributed to prophet Mohammed), extracted from the Riyāḍu Aṣṣāliḥīn book (a.k.a. The Meadows of the Righteous), a compilation of 1896 hadith narratives written by Al-Nawawi and compiled on 1334. The book is widely known to Muslims and has been studied and translated into several languages.

The first section of this chapter examines whether the Quranic corpus is a good representative sample of the classical Arabic texts in general. It illustrates the need for an additional manually annotated corpus for classical Arabic.

This chapter presents the design of the corpus collection and the methodology of its annotation. The annotation has been done through several layers: orthography, segmentation, and morphology. The diacritisation level is increased to the level that all words in the corpus are diacritised. Clitics from each word's free morpheme are detached. All tokens are assigned a part-of-speech tag in addition to eight morphological features.

[5] Some parts of this chapter are based on:

Alosaimy, A. and Atwell, E. (2017) 'Sunnah Arabic Corpus: Design and Methodology', in *Proceedings of the 5th International Conference on Islamic Applications in Computer Science and Technologies*. Semarang, Indonesia. (in press)

8.1 Introduction and Motivation

Language resources (LRs) are recognised as critical components in the development of Natural Language Processing. Annotated corpora, as one example of LR, are used to perform statistical analysis, hypothesis testing, accent verification, verifying grammar within a language domain and for building statistical computational models. Several scholars show the need for freely available Arabic resources, e.g. (Yaseen *et al.*, 2006; Albared, Omar and Ab Aziz, 2009), especially gold standard annotated corpora. In the case of classical Arabic, there are very little available annotated corpora, but they are limited to Quranic texts. Mohamed (2012) built a small corpus of religious texts (all texts are considered classical) and confirmed the need for a larger classical corpus.

The Sunnah Arabic Corpus (SAC) is a corpus of Arabic Hadith (prophet sayings) that is a freely available morpho-syntactically annotated corpus using a fine-grained tagset that conforms with traditional Arabic grammar.

The SAC is tagged with a fine-grained tagset as it aims to take advantage of showing very subtle grammatical differences, the reflects the interest of the experts in syntax and morphology, rather than some unknown specific needs of the end users (e.g. information retrieval). Fine-grained tagsets can always be reduced so that they can support a broader range of downstream applications. Studies show that tagging using fine-grained tagsets and then converting them to reduced tagset is more effective (Kübler and Mohamed, 2012; Zeroual, Lakhouaja and Belahbib, 2017).

After arguing that the Quranic Arabic Corpus is not sufficient, the rest of the chapter is divided into two parts: the first part is an overview of the corpus content, where the second part is more about its annotation. In the first part, we will list the main features and potential uses of SAC, its design and structure, and its availability and accessibility. In the second part, we explain in more details our annotation guidelines on three levels: orthographical, lexical, and morphological. Besides, we talk about the alignment of translations. We conclude by evaluating our collection and annotation process.

8.2 Quranic Arabic Corpus As a Training Corpus

The Quranic Arabic Corpus is a semi-automatic morphologically annotated corpus of the text of the Holy Quran. The annotation of the Quran Arabic Corpus

was done using an automatic tagger then each predicted analysis is examined by two linguists. The second linguist reviewed the annotation after changes made by the first. After that, the public was asked to check the correctness of the annotation.

In this section, we discuss several issues of using the plain Quranic Arabic Corpus as a training dataset, which includes text and style variance, different orthography standard, annotation inconsistencies, and annotation representation problems.

8.2.1 Annotation Consistency

The Quranic Arabic Corpus annotation is very accurate. However, we were able to spot some inconsistency in tagging some *POS tags*, namely ADJ vs N (e.g. word 2:35:9, 50:24:5 vs 2:276:10, see Section 5.7), REM vs CONJ (e.g. look at 2:74), and RES vs EXP (e.g. same sentence "لا إله إلا هو": 9:31:19 vs 59:22:6). These cases including others make their prediction erroneous.

The QAC annotation used an Arabic book of grammatical analysis of the Quran (Salih, 2007) for reference for borderline cases, i.e. the annotator is asked to follow the book for all the annotation. However, the book and the corpus lack the guidelines for handling borderline cases. This makes reusing the same tagset harder for the case of SAC.

ADJ (adjectives) and NOUN (nouns) are very similar in Arabic. Adjectives function as a noun just like in English, and both categories inflect for four categories: gender, number, definiteness, and case. Participles functioning as adjectives inflect as well for voice. Tagsets differ in their definition of adjectives: QAC marks a word as an adjective when it qualifies its preceding corresponding noun, i.e. attributive adjective. QAC usually does not tag predicative adjectives; however, it is not consistent in this matter; for example, verse 29:19 is not consistent with its following verse. The words: "easy/ADJ" and "competent/N" are both adjectives acting as predicate (*khabar*) and should be treated similarly.

RES (restriction) and EXP (exception) are two particles for expressing exception. In traditional Arabic, they differ in their effect on the case mark of the postposition phrase. Usually, if the sentence is complete and sound after removing the exception particle and negative particle, the exception particle is tagged as RES. Because the variance between REM and COND is minimal and only affects the case mark, we suggest reducing the tagset to include one of them.

In a similar matter, REM (resumption) and CONJ (conjunction) are two particles to connect clauses or sentences or to coordinate words in the same clause. In Arabic grammar, conjunction particles cause the word to grammatically follow the previous status of a word (i.e. they come after a word and follow it in status). REM does not; however, this distinction is minimal and does not apply to the majority cases where REM and CONJ are connecting sentences.

Another issue in this matter is the inconsistencies in the *features list*. Morphological features for one POS tag should always be tagged, e.g. case feature for nouns. However, in some words, some values of features are missing. For example, the *number* feature is tagged for 85.13%, 35.81%, 92.74%, 1.62% and 100% of ADJ, N, PRON, PN, and V, respectively. For the full coverage of feature annotation vs. the respective UPOS tags, see Table 8.1. Some features have a "default" value, e.g. the *mood* and *aspect* features of verbs. Some features have a neutral value, but this value is not explicitly specified and is not documented.

Because we would like to train a model to predict the value of these features, we do not want to have unknown values. After a close look into the corpus, we decided to fill the missing values with the most common value (singular for *number* and masculine for *gender*) for the gender and number of nominals. In Table 8.1, the starred percentages can be filled using Arabic knowledge: the *case*, *number* and *person* of pronouns can be inferred from the form of the pronoun, and the *gender* is neutral in first-person or dual pronouns. The *gender* of verbs and *voice* for adjectives are neutral when it is unspecified. However, the rest of underspecified features (underlined percentages) are not easily recoverable. These percentages contributed to the errors of our ensemble analysis.

Table 8.1 Missing Features in Specific UPOS tags

	UPOS	NOUN	PROPN	ADJ	VERB	PRON
1	**Gender**	4%	81%	0.8%	13%*	22%*
2	**Number**	64%	98%	15%	0%	7%*
3	**Person**	N/A	N/A	N/A	0%	15%*
4	**Voice**	N/A	N/A	83%*	0%	N/A
5	**Case**	0%	0%	0%	N/A	100%*
6	**Definite**	0%	0%	0%	N/A	N/A
7	**Mood**	N/A	N/A	N/A	0%	N/A

8.2.2 Text and Style Differences

The Quranic text is a unique classical text. It was preserved through many years away from any changes. It is segmented in verses (6236 numbered verses). There is no mark for sentence boundaries except verses. However, verses can have multiple sentences, and one sentence can span into two verses (e.g. 2:119 and 2:220). It has no punctuation marks; however, its text is annotated with pause marks which were removed from the text in the QAC. It is worth mentioning that Brierley, Sawalha and Atwell (2012) developed an open-source boundary-annotated corpus that the Quran text is segmented into 8230 sentences using pause marks. However, this work does not concatenate sentences that span multiple verses and is not combined in the QAC.

8.2.3 The Quranic Orthography

Quranic Arabic Corpus is an annotated resource where each word in the Quran is morphologically segmented and annotated. While there are authenticated scripts of the Quran that follow current orthographical writing rules in MSA (e.g. Tanzil Project and (Elhadj *et al.*, 2010)), the script used in the QAC is the original Othmani script. The Tanzil format of each word/segment is not given, and there is no direct mapping between these formats.

Some words in Othmani script are a compound of two or three words in the Tanzil format. The number of words in Othmani is slightly lower than the Tanzil version (77430 vs 77797 respectively) due to the concatenation of vocative particles Ya (361 cases) and Ha (4 cases) with their nominals, and two rare cases. For example, the latter word is written as three words in Tanzil project: "*ya+bona+&um~a*" but in QAC, is written as "*yaA Abona >um~a*". We can notice that there are even orthographical changes when splitting. Since POS taggers were trained on MSA format (which Tanzil project follows); these instances in QAC are split off and realigned to Quranic script in MSA format.

Additionally, QAC follows Quranic diacritics which is an extended set of diacritics. There is a relationship between these diacritics and the Tajweed (the rules which govern and help the pronunciation during the recitation of the Quran). For example, the constant Noon letter is not given a *sokun* diacritic when it is in "Ikhfaa" mode. Additionally, some letters are written indifferently (e.g. Yaa->Alif Maqsoura, Alif Madd-> Hamza+Alif, Yaa and Waw as special small diacritics). In

the Tanzil project, diacritics are written according to MSA standards. These differences in word orthography resulted with segments in the QAC needing to be rewritten according to MSA orthographical rules. However, the Tanzil project does not provide segmentation as QAC does.

Finally, some words even in the Tanzil project do not conform to modern writing standards. This applies to variability situations such as when Taa and Taa-Marbouta are written interchangeably (امرأة/امرأت), and for dropped Alif (بسم، باسم), dropped Waw (يدع، يدعو).

To recover from this mismatches, we aligned the text in the QAC text with the text in the Tanzil project, as shown in the algorithm in Section 6.9.

8.2.4 Annotation Representation Scheme

In QAC, the tagset was chosen to comply with the Arabic traditional tagset. It is represented as a CoNLL-like lemma-and-feature format: it consists of four columns–id, form, POS tag, and features separated by vertical bars. The annotation was word-based; then was converted in later versions to morpheme-based. It has a main tagset size of 23 tags appended with a comprehensive feature list. Converting its annotation to morpheme-based required introducing some tags that only appear as a suffix (which was tagged previously as features), e.g. DET. A morpheme does not span over words except for a single case, where a proper noun that spans two words but treated as one morpheme: 37:130:3, إِلْ يَاسِينَ/PN.

While there is some redundancy in the feature list with the POS tags, some feature values are underspecified (maybe to obtain small size file). Since the features are written together, it is not easy to spot missing features (as previously described in 8.2.1). It is not easy to know that there are default values for some features.

Values for each morphological feature is written in the documentation. However, features sometimes have a “neutral” value. For example, the gender of dual verbs is neutral. The value in these cases is left unspecified, and it is hard to know that missing features are neutral without having a background in the Arabic morphology system.

To handle these problems, the annotation scheme required us to build a custom file reader. It handles missing values, and more importantly aligns and

provides Tanzil's text in addition to Othmani script. The QAC (v.2) can be obtained in CoNLL-U format from Sawaref project in Github[6].

8.2.5 Morpheme Form Adjustment

The annotation scheme chose a segmentation that does not recover fused morphemes. For example, attaching a pronoun "هم/hm/*them"* to the "على /ElY/ *to*" preposition makes it "علي/Ely/to+هم /hm/*them*" (which makes the first segment a homograph for a famous proper noun). The /y/ in the first segment is not recovered to /Y/ in the QAC. When segmenting one word, the annotation guidelines should deal with two problems: shared letter between two morphemes, and reshaping of letters.

As described in 6.9, morphemes can be fused in Arabic into one word with a geminate (double) or maddah diacritic. The geminate and maddah diacritic belongs to two morphemes. We found that this decision is not consistent. In most cases, the doubled letter belongs to the first morpheme (e.g. 41:15:11 مِنَّ/P+ا/PRON) but sometimes it belongs (e.g. 41:12:11 زَيَّ/V+نَّا/PRON).

Besides, this decision produces some zero-length (or null) morphemes such as Yaa-ending words followed by Yaa Alnesbah (first-person possessive pronoun), e.g. 44:18:3 إِلَىَّ/P+(null)/PRON. There are 208 zero-length forms in the corpus, all of which are either PRONs or INTG.

Some specific letters are changed when attached to another morpheme. It is usually the case of ending letters (letters that only appear at the end of a word): the Taa Marbouta /p/ and the Yaa Maqasorah /Y/. They are replaced by their sisters': Taa Maftouha /t/ and Yaa letters /y/.

Although annotating the word with its lemma might be efficient in some annotation guidelines, as the lemma recovers the original letters, we hypothesis that machine learning, especially character-based methods can benefit significantly from recovering original form. They can model the effect of attaching morphemes on the level of inflected form, not the lemma. However, recovering original form is not supported by QAC.

[6] https://github.com/aosaimy/sawaref-data

8.2.6 Form vs Function Features

Two morphosyntactic features (the gender and the number of nominals) can be in disagreement between their form-based and functional morphology. Broken plurals, particularly, are singular morphemically (regarding its form) but they function as plural. Broken Feminine nouns do not use the usual feminine suffix (Taa Marbouta), but they use pattern-based. Some nouns are morphemically feminine but they function as masculine, and many feminine nouns do not have their gender morpheme. For more information about gender and noun features in Arabic, please refer to (Habash, 2010, p. 53).

QAC says that the annotation of gender feature of nouns is functionally annotated: "nouns are tagged for gender according to grammatical gender, since this determines how the noun will function syntactically"[7]. However, there is no similar note about the number feature. After some inspection of the corpus, there are number of form-based annotation.

Functional features seem more useful for parsers in agreement and assignment interactions. Even though the accuracy of predicting functional features is less than form-based, the contribution of functional features are more (Marton, Habash and Rambow, 2013).

8.3 SAC Design

The Sunnah Arabic Corpus currently has only one book: Riyāḍu Aṣṣāliḥīn, a compilation of 1896 hadith narratives written by Al-Nawawi and published in 1334. The book will henceforth be referred to as *Riyad*. Riyad was chosen for several reasons:

1. It has been widely accepted as a valid source of prophet sayings.
2. Its codex was validated and investigated by several scholars by a scientific paleographical process.
3. A small subset (42 narratives, 4479 words, ~5% of the book) has been studied linguistically in traditional Ia'rab Arabic grammar (by two books).
4. It has been translated into at least 18 languages.
5. Its narratives have been explained by six written books, at least by 11 scholars (spoken explanation).

[7] http://corpus.quran.com/documentation/gender.jsp

6. It is a good representative sample of the Hadith texts in general as it is quoting narratives from other significant books (see Section 8.9)

While it is available through many websites (e.g. IslamHouse.com[8], Sunnah.com[9]), we chose to download an e-book version of the book from the Shamela[10] library, a downloadable repository that contains at least 5300 Arabic books in Islamic studies, as this library has become the standard library of classical Arabic books. It has already been used to obtain classical Arabic text in building several corpora (Alrabiah *et al.*, 2014; Belinkov *et al.*, 2016; Zerrouki and Balla, 2017).

Two versions of Riyad were available in the Shamela library, and we chose the version with ID# 2348 (Alfahal, 2007). This version is the one investigated by Maher Alfahal who made his investigation and commentaries open freely. Both versions have the same numbering, hadith text (with some slight differences), but they both differ significantly in the commentaries.

Diacritisation of both versions is not full (not every letter has its short vowel). Maher's version is more thorough and accurate using a sample of five narratives randomly chosen. Quranic verses are fully diacritised in both versions (a standard in book editing). Diacritisation has been merged as described in section 7.6.

Shamela books are available in three formats: PDF, EPUB, and BOK (used for their downloadable desktop software). The PDF format is used for the scanned images of the book, and the text is not easily extractable. The BOK version is not suitable as it requires their software to open. Therefore, we chose to proceed with the EPUB format, an e-book file cross-platform widely-used format to view and read the book. Since EPUB format is XML-based, the extraction of the XML version of the book is easy. However, we found that the XML version of Riyad does not tell the difference between different components of the text (like footnotes' co-reference, and page numbers). It does not separate the chain of narrators, prophet sayings, and citation. Neither does it provide a table of contents (see for an example). Therefore, we developed custom software[11] to extract the narratives and verses in a structured format which identifies footnotes, chapters, and sections,

[8] https://islamhouse.com/ar/books/111275/

[9] https://sunnah.com/riyadussaliheen

[10] library http://shamela.ws/index.php/book/2348

[11] http://github.com/aosaimy/riyadh-corpus-collection/

remove inline annotations (for example page break of the original book: "*[p.34]*") and separate footnotes from the original text and links it with co-reference. It also merges narratives that span into multiple pages. It also imports Quranic units annotations from the QAC corpus.

8.4 Corpus Content

Riyad is a collection of 2330 units (precisely 435 Quranic verses and 1896 hadith narratives). It is classified into 20 chapters, and each chapter contains several sections, with a total of 372 sections the covers Islamic morals, acts of worship, and manners. Each section covers a specific topic, and verses and narratives that support the topic.

Figure 8.1 XML version of one page of Riyad book extracted from its EPUB version.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xml:lang="ar" lang="ar" dir="rtl"
xmlns="http://www.w3.org/1999/xhtml"
xmlns:epub="http://www.idpf.org/2007/ops">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-
8"/>
<link href="../style.css" rel="stylesheet" type="text/css" />
<title>رياض الصالحين ت الفحل</title></head>
<body class="rtl">   <div dir="rtl" id="book-container"><hr/>
<a id='C159'></a><a id='C160'></a>
<span class="title">(6) -  كتاب عيَادة المريض وَتشييع المَيّت والصّلاة
عليه وَحضور دَفنهِ وَالمكث عِنْدَ قبرهِ بَعدَ دَفنه</span><br /><span
class="red">144 - </span><span class="title"> باب عيادة
المريض</span><br /><span class="red">894 - </span> عن البَرَاءِ بن
عازِبٍ رضي الله عنهما، قَالَ: أمَرَنَا رسولُ الله - صلى الله عليه وسلم - بعِيَادَةِ
الْمَرِيضِ، وَاتِّبَاعِ الجَنَازَةِ، وَتَشْمِيتِ العَاطِسِ، وَإِبْرَارِ الْمُقْسِمِ، وَنَصْرِ
المَظْلُومِ، وَإِجَابَةِ الدَّاعِي، وَإِفْشَاءِ السَّلَامِ. متفقٌ عَلَيْهِ. (1)<span
class="footnote-hr"> </span><span class="footnote">(1) انظر
الحديث (239).</span>
</div><hr/>
<div class="center">الحديث: 894 ¦ الجزء: 1 ¦ الصفحة:
273</div></body></html>
```

The corpus consists of ~144K *Arabic* words of which about 110K words compose Hadith narratives. The rest compose either the author's commentaries, his introduction, or Quranic verses. More statistics about the corpus are in Table 8.2.

After removing short vowels and punctuations, the number of word types of hadith narratives is ~17K. The word frequency list is presented in Table 8.3.

Table 8.2 Some statistics about the Sunnah Arabic Corpus.

Counts		**Counts**	
Tokens	170453	**Word Types**	17786
Words	144106	**Fully diacritised Words**	102746
Sentences	7670	**Fully diacritised Words**	86.08%
Paragraphs	2075	**Distinct 5-grams**	90347
Documents	372	**Hadith Narratives**	1896
Ann Tokens	7602	**Ann Words**	4528
Ann Sentences	406	**Ann Docs (Narratives)**	60

Table 8.3 The frequency list of Sunnah Arabic Corpus.

Word	**Count**	**Word**	**Count**	**Word**	**Count**	**Word**	**Count**
الله	7883	إلى	528	هذا	251	قلت	160
عليه	3476	يا	528	تعالى	243	أنس	157
قال	3182	كان	523	يوم	243	بها	157
صلى	2528	له	516	أي	237	و	145
وسلم	2470	ولا	450	عمر	218	فإذا	142
من	2068	حتى	447	فيه	213	منه	141
رسول	1990	إن	412	الذي	212	بين	139
رضي	1856	أو	405	صحيح	210	وما	138
عنه	1419	إذا	373	لي	206	النار	137
في	1417	أبو	347	لم	206	عائشة	134
وعن	1406	وقال	339	الترمذي	206	أنه	132
أن	1173	ابن	337	قالت	205	فلما	130
رواه	1151	هريرة	336	رجل	202	شيء	130
ما	876	عنهما	331	فإن	200	والله	128
لا	867	حديث	319	وهو	195	إني	127
فقال	833	يقول	303	هو	194	مع	124
بن	829	حسن	297	عنها	192	فقلت	124
عن	825	عبد	282	الجنة	192	الرجل	123
على	808	رواية	275	قد	188	الصلاة	122
أبي	804	وفي	274	وعنه	185	أهل	122
متفق	747	البخاري	269	كل	177	أحد	121

النبي	650	به	266	اللهم	174	أحدكم	120
مسلم	615	ذلك	266	وإن	174	علي	118
ثم	564	الناس	258	ومن	169	عند	118
إلا	530	داود	255	سمعت	165	شيئا	117

Figure 8.2 A sample of one annotated narrative in CoNLL-U format.

```
# newdoc chapter_id=4 hadith=189
# newpar
# sent_id=1
# text = عن ابن عمر رَضِيَ اللهُ عَنهُما , قَالَ : صَلَّيْتُ مَعَ رَسُولِ اللهِ - صلى الله عليه وسلم
1       عَنْ        عَن±from;about      ADP     P       _
2       ابْنِ       ابْن±son            NOUN    N       Case=Gen|Definite=Ind|Gender=Masc
3       عُمَرَ      عُمَر±Omar;Umar     PROPN   PN      Case=Nom|Gender=Masc|Number=Sing
4       رَضِيَ      رَضِيَ      VERB    V       Aspect=Perf|Number=Sing|Person=3|Voice=Act
5       اللهُ       اللَّه              PROPN   PN      Case=Nom|Definite=Def |Number=Sing
6-7     عَنهُما     _       _           _       _       _
6       عَنْ        عَن±from;about      ADP     P       _
7       هُما        _                   PRON    PRON    Case=Gen |Number=Dual|Person=2
8       ,           ,                   PUNCT   PUNC    _
9       قَالَ       قَال±say            VERB    V       Aspect=Perf|Gender=Masc|Mood=Imp
10      :           :                   PUNCT   PUNC    _
```

For each unit, we keep a record of its numbering, chapter and page in the original printed book, and automatically split its text into sentences with tags to describe the purpose of that sentence using a simple rule-based segmenter.

Hadith units were POS tagged and annotated semi-automatically using the Wasim annotation tool, and Quranic units are matched with their QAC annotation. The format for storing annotation is CoNLL-U format v2.0, which is used by the Universal Dependencies project (Nivre *et al.*, 2017). See Figure 8.2 for an example of an annotated sentence.

8.5 Potential Uses

Potential uses for the corpus are as follows:

- It will help Arabic learners by understanding the interaction of sentence components since it follows the traditional Arabic grammar, ʾiʿrāb (إعراب).
- It will help linguistic researchers interested in Hadith to study the stylistic and vocabulary and other linguistic studies.
- It will help researchers in translation studies to compare different translations of the same Hadith.

- For researchers of Arabic Language Processing, it will help in improving machine translation and classical Arabic understanding using morphological and syntactical annotations.

Moreover, Mohamed (2012) confirms "the need for building religious Arabic linguistic resources" by showing that a POS tagger trained using a small corpus of classical Arabic outperforms another one trained on the Penn Arabic treebank which is 21 times larger.

8.6 Corpus Website

The corpus website aims to offer:

- Part-of-speech concordance search results organised by lemma or surface form.
- A morpheme-based part-of-speech tagged corpus with its morphological features.
- I'rāb of a sample of hadiths in a novel visualised way[12].
- Morphological and lemma-based search for the corpus.
- A parallel text of English-Arabic aligned on the Hadith level.
- A parallel text of Arabic-Arabic commentaries aligned on the hadith level.

8.7 Accessibility and Availability

The Sunnah Arabic Corpus is freely available under the Creative Commons Attribution-ShareAlike 4.0 International License. This permissive licensee allows commercial uses and allows adaptations of the work to be shared as long as others share alike. The corpus will be also available online[13] which allows easy to use corpus functionalities.

8.8 Annotation Setup

The annotation of the SAC has been done using the Wasim toolkit (see Chapter 9). The toolkit was configured to use the MADAMIRA toolkit in the backend as a lexicon resource. To recover from the mismatch between MADAMIRA and the QAC tagsets, we mapped each tag in the MADAMIRA tagset

[12] Left for future work.

[13] http://corpus.al-osaimy.com

into a set of tags in the QAC. These mappings were initially from work carried in 6.4.2, 6.4.3, and the parallel annotated corpus. Solutions with tags in the MADAMIRA tagset that map to *n* tags in the QAC are copied *n* times and displayed in the selection panel (morphological analyser solution picker pop up).

Besides, if a word was annotated previously (either in the Quranic Arabic Corpus or annotated parts of narratives), its previous possible annotations will be displayed at the top. We build an offline lexicon of annotated parts of narratives in addition to the QAC corpus. This practice increased the consistency and reliability of the annotation. The context of this annotation is shown next to the analysis to help the annotator understand why it was annotated in such a way. Sometimes, errors in previous documents are spotted using this helper tool.

Documents were chosen randomly from a particular set of the Riyadh Asslaheen book: the intersection set of documents that are also in the Nawawiah Forty Hadiths book (Nawawiah) (An-Nawawi, 1976). Nawawiah has been wholly annotated in the traditional Arabic grammar (Ia`rab) by two works (Yosef, 2003; AlOmari, 2005). These set of documents are reserved for validation of the morphological annotation.

The annotation was done semi-automatically. Models were built and used for initial annotation, and the annotator performed corrections. Models used are built on a cumulative basis. The first model was trained on the only QAC corpus. Next, models were trained with each 1000 additional annotated words in the SAC in addition to the QAC. Models show good improvements as the training dataset increases.

Models are trained using the UDPipe toolkit. The tokeniser component is trained with 100 epochs with a batch size of 50 per iteration (with a dropout of 0.1) and a learning rate of 0.005. Character embeddings are 24 dimensions. For the tagger component, we train two models separately: one for lemma and one for POS tag and morphological features.

To help the annotator find the proper POS tag of closed set words (words that have not been tagged as nominals/verbs), Wasim shows possible annotation of these words derived from the QAC corpus. It is mostly useful for homographs, where some can have up to seven possible tags. For example, the clitic 'w' in the QAC tagset can be CONJ, REM, CIRC, SUP, PRON, COM, or P. In each case, the

annotator is given a set of examples and the possibility of that tag (without taking the context in consideration).

8.9 Orthographical Annotation: Diacritisation

"Borrowing" diacritisations from similar contexts have raised the percentage of diacritised characters of the corpus, which in return reduced the word ambiguity. This section shows the diacritisation ambiguity level on the case of the Sunnah Arabic Corpus, brief results of the experimental study of automatic diacritisation previously proposed in 5.6, and the guidelines for standard diacritisation of Arabic.

8.9.1 Ambiguity in Sunnah Arabic Corpus

In this section, we demonstrate the ambiguity level by expressing the number of possible diacritisation within the language (expressed by morphological analysers). Using the SAWAREF toolkit, we ran four morphological analysers, namely Elixir Functional Morphology (EX) (Smrz, 2007), ALMORGEANA (included in MADA toolkit) (AL) (Habash, Rambow and Roth, 2009), AraMorph (BP) (Buckwalter, 2002a), and AlKhalil (KH) (Boudchiche *et al.*, 2016), on the lexicon of SAC (17.7K distinct words). The average number of possible diacritised forms is shown in Table 8.4. It shows the maximum, mean, and median of the number of possible diacritisations per morphological analyser. The coverage column refers to the average percentage of diacritised letters.

Table 8.4 Possible Diacritisation Statistics Per Morphological Analyser.

MA	Max	Mean	Median	Coverage
EX	124	8.46	6	67.46%
KH	96	10.38	7	80.64%
BP	20	2.38	2	47.67%
AL	23	3.69	3	42.65%

Differences in statistics do not necessarily imply better coverage. Diacritisations of one example word, as analysed by each tool, are shown in Table 8.5. We can see that BP and AL do not recover the last diacritic (/u/, /a/, /i/), and therefore different moods and cases of verbs and nouns are not iterated. Some tools, like KH, produce diacritisations with *Tanween* if it is suitable (like in indefinite nouns). However, KH sticks to the Hamza form and does not produce other possible

Hamza locations. EX produces all shapes of the Hamza letter which led to the largest possible number of diacritisations. AL makes a similar pattern of iterating possible Hamza, Madda, or plain Alif shapes.

Table 8.5 Possible diacritisations of the word (آثم, /|vm/, "a sin") from four MAs.

Tool	Possible Diacritisation	
EX	|vam |vama |vamu |vimN |vimK |vima |vimu |vimi >avamN >avamK >avama >avamu >avami >avima >av~ama >av~im >uvima >uv~ima <ivmN <ivmK <ivma <ivmu <ivmi	آثَم آثَمَ آثَمُ آثَمٌ آثَمٍ آثِمَ آثِمُ آثِمٌ أَثَمَ أَثَمٍ أَثِمَ أَثِمٌ أَثَّمَ أَثِّمَ أُثِّمَ أُثِّمٌ إِثَّمَ إِثِّمَ إِثِّمٌ إِثْمَ إِثْمٍ إِثِمَ إِثِمٌ
KH	|vama |vamu |vamo |vuma |vumu |vumo |vimN |vimK |vima |vimu |vimi |vimo	آثَمَ آثَمُ آثَمْ آثَمٌ آثَمٍ آثِمَ آثِمُ آثِمْ آثِمٌ
BP	|vim	آثِم
AL	|vim >avam >avima >av~ama <ivom	آثِم أَثَم أَثِمَ أَثَّمَ إِثْم
ALL	<ivmK <ivmN <ivma <ivmi <ivmu <ivom >avam >avamK >avamN >avama >avami >avamu >avim >avima >uvima |vam |vama |vamo |vamu |vim |vimK |vimN |vima |vimi |vimo |vimu |vuma |vumo |vumu	إِثْمٌ إِثْمٍ إِثْمَ إِثْمُ إِثْمْ أَثَم أَثَمٍ أَثَمَ أَثَمٌ أَثَمَ أَثَمِ أَثِمَ أُثِمَ آثَمَ آثَمَ آثَمُ آثَمْ آثَمٌ آثَمِ آثِمَ آثِمٌ آثِمَ آثِمُ آثِمْ آثُمْ

Even though all these morphological analysers produce full diacritisation, we notice that diacritisation is not standard in multiple notions: for example, the place of the tanween, diacritizing the letter that precedes a long vowel or not, and diacritizing the Alif and Lam letter of AL article for definition (the difference between sunny and moony AL).

We merged all diacritised forms from these after standardizing their diacritisation (see 5.6.3 for rules). One example is shown in Table 8.5 (last row). After removing duplicates, the average ambiguity per word is 17.42 possible diacritisations/word.

8.9.2 Automatic Diacritizing

Diacritisation reduces the degree of ambiguity of morphological annotation, and thus increases the accuracy of part-of-speech tagging (Dukes and Habash, 2010; Habash, Shahrour and Al-Khalil, 2016). Since we have multiple

versions/investigations of the same book, and the same narrative might be recited or quoted in other books as well, we merge the diacritisation by combining words that have similar 5-gram context.

The SAC satisfies the five assumptions in our methodology and is a good candidate for our diacritisation process due to several reasons:

1. It compiles narrations reported in other Hadith books (e.g. Albukhari) which make these books a good source for diacritisation.
2. Its codex was validated and investigated by several scholars by a scientific palaeographical process; at least there are two digitally available validated versions of the same text.
3. Its narratives have been explained in 6 written books.

The details of this methodology and the evaluation on the case of Sunnah Arabic Corpus can be found in section 5.6. In short, the source text is initially about 48.66% diacritised, and after borrowing diacritisation, the percentage jumps to 76.41% with low diacritic error rate (DER=0.004), compared to 61.73% (DER=0.214) using the MADAMIRA toolkit, and 67.68% (DER=0.006) using the Farasa toolkit. More importantly, this method has reduced the word ambiguity from 4.83 diacritised forms/word to 1.91, which suggest that it is useful for the morphological annotation task.

8.9.3 Manual Diacritisation

In this section, we introduce the guideline section for diacritizing the Sunnah Arabic Corpus. As illustrated before, several diacritisation standards exist, and for consistency and stability, we write a short list of guidelines for annotators to follow. The guidelines cover diacritizing multi-word tokens (before segmentation) and morphemes (after segmentation).

Rules:

General

- Each letter should have two diacritics at most.
- In case a letter has two diacritics, one of them should be the Shaddah diacritic.
- A Sokun diacritic cannot be accompanied with a Shaddah diacritic.
- There should be no duplicate diacritics.

- Diacritics cannot standalone. Any diacritic must accompany a letter.

Long Vowels

1. Diacritizing a constant letter that precedes long vowels is necessary, including the Shaddah diacritic if the constant is a long consonant (geminate).
   Goal: This is to facilitate a way to find long vowels in the future.

2. Diacritizing a long vowel (includes Alif, Alif Maqsorah, non-consonant Waw and Yaa letters) is unnecessary. Never diacritise Alif and Alif Maqsorah letters.
   Goal: This is to save annotators' time.

3. DiacritiseWaw and Yaa letters if they are not long vowels.
   Goal: This is to draw a distinction between long vowels.

4. The Sokun diacritic should be placed on the Waw letter when it marks a group of people (Waw Aljamaha)
   Goal: This is to differentiate it from A-Muthanna.

Definite Article AL

5. Diacritizing the Lam letter in the article AL is unnecessary.
   Goal: All Lam letters that are undiacritised are part of articles, and this saves time.

6. Diacritisethe long consonant letter after AL with Shaddah (only if multi-token).
   Goal: This is to distinguish between Soony and Moony articles.

7. Shaddah diacritic should be removed when segmenting it from the solar AL article.
   Goal: This is to reduce the sparseness of the words.

Tanween

8. The tanween diacritics should be placed on the letter it modifies (tanweened letter), not on the Alif or the Alif Maqsoura letters.
   Goal: This is the correct place of tanween. Incorrect placement contradicts with Rule 2.

Shaddah

9. The Shaddah diacritic should always be written before other diacritics.
   Goal: This is for consistency reasons.

10. The Shaddah diacritic should always be companioned with other diacritics except with (Rule 1, Rule 11).
    Goal: This is to ensure no missing diacritisation.

11. The Shaddah diacritic should be placed after the solar AL article (Rule 5).
    Goal: This is because it is long consonant (geminate).

12. The Shaddah diacritic should be removed when segmenting it from the solar AL article.
    Goal: This is to reduce the sparseness of the words.

13. The Shaddah should be segmented into its two origin diacritics (a Sokun and a diacritic) if it is formed because of inflexion.
    Goal: This is to reduce the sparseness of the words.

Maddah

14. Maddah can only be with Alif.
    Goal: Alif with Maddah is considered a different letter (in the Unicode representation). However, it is actually the result of two letters merged with a diacritic. It should not be misspelt as a normal Alif.

15. Maddah should be segmented into its two origin diacritics (a Hamazah with Fatha and a long vowel Alif) if it is formed because of inflexion.
    Goal: This is to reduce the sparseness of the words.

Diacritic of Declined/Conjugated nouns/verbs

16. Diacritizing of the "last" diacritic (the case and mood marks) of a declined noun or a conjugated verb is optional but is strongly encouraged.
    Goal: This is to save the annotator time and to reduce the sparseness of the words. The correct case/mood mark is not easily recovered though. We plan later to extend the tagging by adding case/mood mark to morphological features.

17. Diacritise invariable nouns or verbs.
    Goal: This is to reduce the sparseness of the words.

Hamzah Wasel

18. Do not diacritise the Alif if it is a Hamza Wasel.
    Goal: This is to mark Hamzah Wasel as it affects pronunciation. It is a long vowel alif but sometimes is dropped to avoid the double unvoweled letters.

19. The diacritisation of the last letter of a word should not differ according to the subsequent word (the meeting of two vowels).
    Goal: This is to reduce the sparseness of the words.

Hamzah

20. The lower Hamzah on the Alif can only occur at the beginning of a word.
    Goal: This is for consistency reasons.

21. Do not diacritise lower Hamzah.
    Goal: This is to save the annotator's time as it is obvious.

8.10 Morphological Annotation

In this section, we provide the official guidelines used for the annotation of the Sunnah Arabic Corpus. These guidelines are meant for the consistency and stability of the annotation of the corpus and are meant to be used for human annotators. Since the annotation is done in a semi-automatic process, we tried to adapt the automatic part of the process to follow these guidelines. However,

automatic processing is prone to errors and do not necessarily comply with our guidelines. Annotators should always correct these errors.

The morphosyntactic annotation aims to build a dependency treebank for classical Arabic eventually. In a dependency treebank, each node/token is tagged with one tag that represents a dependency relationship between a governor and a dependent. The syntax annotation is beyond the scope of this thesis; however, the annotation guidelines are designed to allow continuing the annotation process with syntactic annotation in future.

In segmentation, tokens are systematically segmented so that the annotation is done to syntactic words (not orthographic words), which means that we want to split off *clitics* (not affix) such as the w+/CONJ from nouns. In contrast, we generally do not split off Y+ from imperfect verbs, as this affix does not contribute to the traditional Arabic grammar (لا محل لها من الإعراب), i.e. is not dependent (not a complement or modifier to the head).

In other morphological features, we select a subset of features from the recommended morphological and lexical feature set from (Marton, Habash and Rambow, 2013), which explored the contribution of possible morphological sets to parser's accuracy in the context of Modern Standard Arabic.

The guidelines here is a collection of the best practices of the annotation of the Quranic Arabic Corpus and the Universal Dependency version. 2. POS tagset is an extended version of the QAC, and the morphological and lexical features comply with the Universal Dependency guidelines. In all of these guidelines, we do not include the guidelines of the syntax annotation due to irrelevance.

8.10.1 Segmentation

In addition to dividing the text into a group of words separated by spaces and punctuation (i.e. tokenisation), SAC divides the words into their morphological segments (i.e. segmentation), if applicable.

Although we rely on morphological analysers for the segmentation of the corpus text, they do sometimes fail to identify the correct segmentation, because the word is homogeneous. Non-diacritised (i.e. underspecified) texts have more orthographical homogeneity than the diacritised texts.

We can segment a word into prefixes, the stem of the word and suffixes. Prefixes and suffixes are bound morphemes, while the stem is a free morpheme.

Usually, an inflected word consists of one free morpheme and one or more bound morphemes. However, some combinations of a prefix and suffixes can form a valid word like (bi+hi, PREP+PRON, with+him).

Unlike the QAC, we decided *not* to mark segments as either a *bound* or *free* morpheme nor as a *stem* or *affix*. We followed the CoNLL-U guidelines published by the Universal Dependency project. In the CoNLL-U format, a list of morphemes is listed with no requirement of labelling a morpheme as either an affix, stem, bound or free.

The primary reason is to save annotators time. Even though this information might be helpful (e.g. in morphological alignment, see Section 5.4.5), it can be recovered for most of the morphemes with a little manual work for some ambiguous morphemes. The second reason is the lack of a standard definition of stems which adds more confusion, as illustrated in our comparative evaluation of taggers in Chapter 4. For example, stems of words that consist of two bound morphemes are arbitrarily chosen. We do not have the resources to check its contribution to the parser's quality, and we leave it for future work.

In general, there are five proclitics and one enclitic that should be detached (see Table 8.6). The content of this table is originally automatically generated by analysing the Quranic Arabic Corpus. From the list of all inflected tokens, we extract the possibility of attaching these clitics to other POS tags. We utilise the PREFIX/STEM/SUFFIX mark used in the QAC to determine the possibility of attaching one affix to a free morpheme. We expect that the tokeniser to split off non-Arabic characters of the form of the bottom four tags (13-16).

In Table 8.6, prefixes and suffixes are necessarily bound morphemes, while the 16 categories are the free morphemes. Pronouns, for example, can be a free or bound morpheme; however, bound pronouns only inflect nouns, adjectives, verbs and adverbs. Determiners are always bound morphemes, and therefore the determiner's row is all empty. Table 8.6 does not list all possible inflexions. Rarely, two particles can be inflected such as (إِنَّمَا, <in~amaA, "no more than; only"), and both particles are free morphemes. The table shows all possible inflexions regardless of the state of the morpheme (free or bound).

Table 8.6 The compatibility table of affixes and UPOS tags.

				Bound Clitics

	Bound morpheme	UPOS	XPOS (Examples)	Question Particle[14]	Conjunctions[15]	Prepositions [15]	Complements[16]	Definite article [15]	Pronouns [15]
1	**Nouns**	NOUN	N	✓	✓	✓		✓	✓
2	**Proper Noun**	PROPN	PN	✓	✓	✓		✓	
3	**Adjectives**	ADJ	ADJ/IMPN	✓	✓			✓	✓
4	**Verbs**	VERB	V	✓	✓		✓		✓
5	**Adverbs**	ADV	T/LOC	✓	✓				✓
6	**Pronouns**	PRON	PRP/DEM/REL	✓	✓	✓			[17]
7	**Particles**	PART	ACC,AMD,ANS ,…	✓	✓	✓			
8	**Prepositions**	ADP	P	✓	✓				
9	**Conjunctions**	CCONJ	CONJ	✓		✓			
10	**Subconjunctions**	SCONJ	SUB	✓	✓	✓			
11	**Determiner**	DET	N/A[18]						
12	**Interjection**	INTJ	INTJ						
13	**Symbols**	SYM	SYM						
14	**Punctuations**	PUNCT	PUNCT						
15	**Numbers**	NUM	NUM						
16	**Other**	X	X						

Any combination of two morphemes that do not follow this table will show a warning in the Wasim annotation tool (see Chapter 9). Wasim will also display an online subset of the guidelines that are related to the highlighted word.

Please note that the 12th to 16th categories should always be tokenised and separated from other words by the tokeniser. Punctuation should be separated from

[14] The Question Particle is the token that is tagged XPOS:INTJ and UPOS:PART. It attaches to any free morphemes.

[15] Conjunctions and prepositions, the definite article, and pronouns are tagged as UPOS:CONJ, UPOS:ADP, UPOS:DET, UPOS:PRON.

[16] Complements are a subset of particles (XPOS:CAUS, XPOS:CIRC, XPOS:COM, XPOS:EMPH, XPOS:EQ, XPOS:FUT, XPOS:IMPV, XPOS:INTG, XPOS:REM, XPOS:RSLT) that can attach to verbs.

[17] Pronouns do not attach to other pronouns unless both are bound morphemes.

[18] Determiner are used in the QAC only for the definite article, which is not a free morpheme.

each other, for example, a double quotation and a colon. Numbers, however, should not be separated into its digits. The date should be marked as one token.

Table 8.7 The possibility of attaching one UPOS tag to another.

	UPOS	NOUN	PROPN	ADJ	VERB	ADV	PRON	PART	ADP	CCONJ	SCONJ	DET
1	**NOUN**						29.45	3.05	12.63	11.86		43.01
2	**PROPN**							15.51	32.59	25.39		26.5
3	**ADJ**						0.17			0.17		99.65
4	**VERB**						68.61	14.42		16.97		
5	**ADV**						42.87	26.49		27.77	0.21	2.66
6	**PRON**	15.36			37.47	1.22	7.45	10.97	16.86	10.67		
7	**PART**	3.53	1.13		17.48	1.67	24.35	33.77	2.73	14.57	0.37	0.4
8	**ADP**	22.07	3.57				56.47	4.12		4.72	0.83	8.22
9	**CCONJ**	16.14	2.17	0.01	24.17	2.06	27.83	17.11	3.68		0.17	6.66
10	**SCONJ**					1.24		34.16	50.93	13.66		
11	**DET**	74.05	2.86	5.72		0.25		0.59	8.1	8.43		

8.10.2 Lemmatisation

In addition to tagging inflectional features, we annotate the Sunnah Arabic Corpus with one lexical feature. Arabic is a Semitic language that inherits the templatic characteristic, so a word can be described by its root and pattern. Lexical features usually include the word lexeme (or its representative: the lemma), pattern (either the pattern of the word or the lemma) and the root of the word. Some research in the literature (Smrz, 2007; Sawalha and Atwell, 2013) include the number of root letters, verb root (or form), and noun finals.

In Arabic, there is no infinitive form of verbs. We chose the represented word form that is most commonly used in the traditional dictionaries. While words are traditionally grouped by their roots, the different senses are iterated using one representative word (the lemma). For verbs, it is usually the *perfective third-person masculine singular* form of the verb. For nouns, it is *nominative singular masculine* (if possible) form.

Similar to the QAC, we have tagged words with their roots[19]. The root of a word is the original consonants letters of the word before injecting it into a vocalised

[19] The root feature is set on the MISC column. It is automatically generated and not yet validated.

pattern. Arabic derivational morphology is mostly templatic (see Section 2.4). The root provides a more profound abstraction than lemma as it abstract over the derivational and inflectional morphology while lemma abstracts over inflectional morphology.

This abstraction seems handy: Marton *et al.* (2013) found that the combination of lemma and root increased the parsing accuracy by 0.03. The gain is attributed to the reduction in data sparseness, and the grouping of semantically related words together.

In addition to parsing, lemma and root annotation is useful in the context of information retrieval. It is useful for finding all occurrences of a particular word, especially for highly inflectional languages like Arabic, where one word can have hundreds of possible inflected forms. For example, the lemma ***kataba*** can be found in "over 400 different forms" (Kübler and Zinsmeister, 2015, p. 43).

8.10.3 POS Tagging

Instead of using the well-known coarse traditional three-way tagset (nominals, verbs, particles), we implemented a two-level tagset: coarse (UPOS) and fine-grained (XPOS) tagsets. Each segment is tagged with two tags from each tagset. We followed and extended the fine-grained tagset of the Quranic Arabic Corpus, coupled with the Universal Dependency Tagset. The original tagset of the Quranic Arabic corpus has about 45 tags: nine tags for nominals, one for verbs, 34 tags for particles and one for Quranic initials. The universal tagset is 17 tags (with one tag (AUX) never used in Arabic text). Each XPOS tag is mapped to one UPOS tag as shown in Table 8.8.

Because the original tagset used in the QAC is dedicated to the Quranic text, which has no punctuation or numbers, we have extended the tagset by adding some tags encountered in the SAC. This extension has been designed to suit classical Arabic texts in general. Added tags are marked with a star in Table 8.8.

We decided to support universal tagset annotation by adapting the Universal Dependency tagset. The project develops cross-linguistically consistent treebank annotation, and its tagset is used to annotate treebanks in many languages. At least there are over 60 supported languages with more than 100 treebanks. This annotation facilitates the use of other taggers and parsers.

Arabic grammarians have studied classical Arabic since the centuries of the prophet and his companions. They developed a grammar known as “I3rab”. Since we plan to annotate the corpus using such grammar in the future, we found that two tagsets follow its terminology: The SALMA tagset (Sawalha and Atwell, 2013) and the QAC tagset (Dukes and Habash, 2010).

The QAC tagset has been used over the SALMA tagset for several reasons:

- The QAC tagset is designed for the syntax annotation.
- The QAC tagset is used and tested in the QAC.
- The QAC corpus is larger in terms of the number of words.

The detailed classification scheme requires that each tag be clearly defined, giving examples in the annotated document. This guideline should include how to identify difficult borderline situations so that all the examples in the group can be consistently marked. Tagset schemes must specify how to select a label if a word may have different labels in a different context (Atwell 2008). The SALMA tagset is described in more detail compared to the QAC. However, both tagsets suffer from missing guidelines for difficult borderline situations.

The QAC annotation used an Arabic book of grammatical analysis of the Quran (Salih, 2007) for reference for borderline cases, i.e. the annotator is asked to follow the book for all the annotation. However, the book and the corpus lack guidelines handling for borderline cases and is only limited to the Quran. This makes reusing the same tagset harder for the case of SAC.

For this purpose, we implemented a consistency checker and helper component in the Wasim annotation tool. The QAC tagset should be easily grasped for annotators with strong traditional Arabic grammar background. For the borderline situations, we ask the annotator to mark these situations for later judgment, and plan to write detailed guidelines for these situations.

Table 8.8 Two-level part of speech tagset used in SAC.

UD Tag (UPOS)	**QAC Tag (XPOS)**	**قسم الكلام**	**Description**
NOUN	N	اسم	Noun
PROPN	PN	اسم علم	Proper Noun

UD Tag (UPOS)	QAC Tag (XPOS)	قسم الكلام	Description
ADJ	ADJ	وصف (وليس صفة)	Adjective
	IMPN	اسم فعل أمر	Imperative verbal noun
PRON	PRP	ضمير	Personal Pronoun
	DEM	اسم إشارة	Demonstrative Pronoun
	REL	اسم موصول	Relative Pronoun
ADV	ADV	ظرف زمان*	Time Adverb*
	ADV	ظرف مكان*	Location Adverb *
VERB	V	فعل	Verb
ADP	P	حرف جر	Preposition
CCONJ	CONJ	حرف عطف	Coordinating Conjunction
SCONJ	SUB	حرف مصدري	Subordinating Conjunction
PART	ACC	حرف نصب	Accusative particle
	AMD	حرف استدراك	Amendment Particle
	ANS	حرف جواب	Answer Particle
	AVR	حرف ردع*	Aversion Particle
	CAUS	حرف سببية	Causal Particle
	CERT	حرف تحقيق	Certainty Particle
	CIRC	حرف حال	Circumstantial particle
	COM	واو المعية	Comitative particle
	COND	حرف شرط	Conditional particle

UD Tag (UPOS)	QAC Tag (XPOS)	قسم الكلام	Description
	EQ	حرف تسوية	Equalisation particle
	EXH	حرف تحضيض	Exhortation particle
	EXL	حرف تفصيل	Explanation particle
	EXP	أداة استثناء	Exceptive particle
	FUT	حرف استقبال	Future particle
	INC	حرف ابتداء	Inceptive particle
	INT	حرف تفسير	Particle of interpretation
	INTG	حرف استفهام	Interrogative particle
	NEG	حرف نفي	Negative particle
	PREV	حرف كاف	Preventive particle
	PRO	حرف نهي	Prohibition particle
	REM	حرف استئنافية	Resumption particle
	RES	أداة حصر	Restriction particle
	RET	حرف اضراب	Retraction particle
	RSLT	حرف واقع في جواب الشرط	Result particle
	SUP	حرف زائد	Supplemental particle
	SUR	حرف فجاءة	Surprise particle
	VOC	حرف نداء	Vocative particle
	INL	حروف مقطعة	Quranic initials
	EMPH	لام التوكيد	Lam emphasis

UD Tag (UPOS)	QAC Tag (XPOS)	قسم الكلام	Description
	IMPV	لام الامر	Lam Imperative
	PRP	لام التعليل	Lam explanation
DET	DET	ال التعريف	The definite article
INTJ	INTJ*	خالفة	Violation
X	X*	أخرى	Other
SYM	SYM*	رمز	Code
PUNCT	PUNCT*	علامة ترقيم	Punctuation mark
NUM	NUM	أرقام	Digits

8.10.4 Morphological Features

Morphological features play a critical role in helping parsers to determine the correct parsing tree. These features distinguish lexical and grammatical properties that are not covered by a POS tag. The line, however, between a POS tag and a morphological feature is cloudy. Sometimes, morphological features are explicitly or implicitly encoded in a POS tag, e.g. NNS.

Parsers use word order, POS tag and morphological features to find the syntactic role of one word. However, in morphologically rich languages, which are known to have free word language, the role of the word order is limited, and the role of morphological features is prominent. Morphological features help parsers in two ways: agreement (like noun-adjective and verb-noun agreements) and assignment (assign the subject label to nominative noun) (Marton, Habash and Rambow, 2010).

The morphological features space is vast in morphologically rich languages. Universally, there are 48 morphological features, but most of them do not apply to Arabic, with an average of 5.2 possible values for each feature. In the SALMA tagset, a tagset designed to be a standard tagset that "adds more fine-grained details to the existing tagsets" (Sawalha and Atwell, 2013, p. 63), the author presented at least 16 morphological features with an upper limit of about 2 million possible values for one word. The average number of morphological values is 4.2.

Consequently, designers of the parsing model should aim to carefully select morphological features for annotation that contribute to the accuracy of the parsing. Optimally, a feature should be considered when it is accurately predictable and useful for making an attachment or labelling decision; however, the feature should be omitted if its added information is redundant to another feature (Marton, Habash and Rambow, 2013). For example, the GENDER feature might help determine the attachment of an ADJ in the following example: (ربكم ذو رحمة واسعة /rbkm *w rHmp wAsEp/ *Your Lord is full of mercy all-embracing*). While the CASE is very relevant, it might not be accurately predicted. CASE and CASE_MARKS in the SALMA tagset are mostly redundant.

The selection of supported morphological features in our corpus is initially based on Marton, Habash and Rambow (2013). In one part, the authors explored the contribution of ten distinct inflectional features, namely DET, PERSON, ASPECT, VOICE, MOOD, GENDER, NUMBER, STATE, CASE, and RAT. They contrast the contribution of functional vs form-based features of GENDER and NUMBER. The best model contains five features: DET, PERSON, FN-NUMBER, FN-GENDER, FN-RAT.

Their results were presented in the context of undiacritised Modern Standard Arabic text. The included tagsets are not based on traditional Arabic grammar. In traditional Arabic grammar, four more morphological features play critical roles: CASE, MOOD, ASPECT, and VOICE. For example, the labelling of the subject in a passive sentence is different than an active sentence (نائب الفاعل vs فاعل). Additionally, the CASE feature specifies the role of the noun phrase in the sentence. We expect these four features to be more accurately predictable with an input of fully diacritised classical text, so we add them to the list. Therefore the final set of features is DET, PERSON, FN-NUMBER, FN-GENDER, VOICE, ASPECT and CASE (see Table 8.9).

Table 8.9 The included morphological features and their values.

#	**Arabic Name**	**English Name**	**Possible Values**			
1	النحوي الجنس	Gender	Masc	Fem	-	
2	الحالة الإعرابية	Case	Nom	Acc	Gen	-

3	العدد النحوي	Number	Sing	Dual	Plur	-
4	التعريف	Definiteness	Def	Ind	Cons	-
5	الإسناد	Person	1	2	3	-
6	البناء	Voice	Act	Pass	-	
7	نوع الفعل	Aspect	Perf	Impf	Imp	-

We decided to use functional features instead of form-based features for gender and number. The broken plural form of nouns is tagged as plural even though it does not have one of the sound number suffixes. Similarly, the gender of nouns is tagged such that it satisfies grammatical agreements: e.g. adjective-noun agreements. Functional features seem more useful for parsers in agreement and assignment interactions. Even though the accuracy of predicting functional features is less than form-based, the contribution of functional features is more (Marton, Habash and Rambow, 2013).

8.11 Meta-Annotation

Classical corpora represent an interesting and challenging use case because they are the basis of empirical studies in many disciplines. They shows a wide variety of reuse possibilities. General annotation of classical corpora requires different meta-data that reflects the relationship between the original historical text and their interpretation (Odebrecht, 2018). Luckily, all annotations of the original text are in footnotes which we kept at the document level.

Hadith corpora should as well reflect the science of Hadith principals (Najeeb *et al.*, 2015). We follow an abstract level of classification at the sentence level. Each narration (Hadith) is composed of three components: Isnad (the chain of narrators), Matn (the text of the narration), and the Takhreej (the list of reporters nad their comments). Each hadith in the Sunnah Arabic corpus is segmented at the sentence level, and sentences are meta-annotated with the class of which Hadith component they belong to.

The morphological annotation is not always perfect. In many cases, annotators can be confused by different possible annotations, and a further examination by another expert should be done in the future. Annotators can mark

these using the NOTSURE tag. Additionally, some annotations need further justification for other people. For example, the choice of a lemma, a case value or a POS tag might need a justification to remove the misinterpretation.

This is more important when we deal with highly respected (or religious) texts. People tend to have different interpretations of the holy text, and these interpretations originates from ambiguities in orthography, morphology, syntax, or semantics. For example, the prophet saying: (إنما يرحم الله من عباده الرحماء /<nmA yrHm Allh mn EbAdh AlrHmA'/ *Allah is Compassionate only to those among His slaves who are compassionate [to others]*) has three valid morphological analyses (Table 8.10) (Akbari, 1986, p. 75). With the limit of one possible annotation, some meta information about the annotation is required, e.g. justification, other possible annotations.

Table 8.10 Different valid annotations of one prophet saying.

	Form	Form	One	Two	Three
1-2	إنما	<nmA			
1	إنَّ	<n	ACC نصب	ACC نصب	ACC نصب
2	ما	Ma	PREVكاف	REL موصولة	SUB مصدرية
3	يرحم	yrHm	V	V	V
…					
8	الرحماء	AlrHmA'	NOUN CASE=Acc	NOUN CASE=Ind	NOUN CASE=Ind

Meta information is written in the same file. The CoNLL-U format allows miscellaneous information to be written on different levels: segment, multi-token, sentence or document. Segment and multi-token miscellaneous information are stored in the eighth column: MISC. Document- and sentence-based meta information are stored as prior lines with a leading hash symbol. In the MISC column, we store whether the analysis (via mark FROM_MA=1) is initially from a morphological analyser, though some edits might have been made later. We mark NOTSURE=TRUE segments to allow further investigation later. Annotators can put some notes (e.g. for justification) on different levels: document, sentence or elements. Wasim, by default, stores an annotator identifier and session annotation information such as the start, finish, breaks date and time. Also, before the start of the sentence, its text and its id are written as a comment to ease reading of the full sentence. In the last resort, the saved text also helps in finding changes of words or segments (e.g. missing words).

8.12 Conclusion

This chapter introduced the corpus and described its collection process, content, and its distribution and availability. It argued that the Quranic Arabic corpus needs some adaptation before it can be used for training machine learning models. It described briefly the project's potential uses in different fields including linguistics studies, natural languages processing, and translation studies. In the second part, it introduced the process of orthographical and morphological annotation with in-depth description of the process and tagsets.

For future work, we aim to continue the process of semi-automatically annotating the corpus and include other books as well. It will be very helpful to manually align the corpus to different languages/commentaries at the sentence level. In addition, word-to-word translation proved to be helpful in Quran understanding, and we might consider automatic word alignment with other languages.

In the following chapter, we present a new linguistic resource: Wasim, an annotation toolkit that was used in annotating the Sunnah Arabic corpus.

9 WEB-BASED ANNOTATION TOOL FOR INFLECTIONAL LANGUAGE RESOURCES

Chapter Summary[1]:

We present Wasim, a web-based tool for semi-automatic morphosyntactic annotation of inflectional languages resources. The tool features high flexibility in segmenting tokens, editing, diacritizing, and labelling tokens and segments. Text annotation of highly inflectional languages (including Arabic) requires key functionality which we could not see in a survey of existing tools. Wasim integrates with morphological analysers to speed up the annotation process by selecting one from their proposed analyses. It integrates as well with external POS taggers for kick-start annotation and adaptive predicting based on annotations made so far. It aims to speed up the annotation by completely relying on a keyboard, with no mouse interaction required. Wasim has been tested on four case studies and these features proved to be useful. The source-code is released under the MIT license[2].

[1] Some parts of this chapter are based on:

Alosaimy, A. and Atwell, E. (2018) 'Web-based Annotation Tool for Inflectional Language Resources Major features', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Miyazaki, Japan: European Language Resources Association (ELRA), pp. 3933–3939.

[2] The source code and a demo are available at http://wasim.al-osaimy.com

9.1 Introduction

Inflectional languages or fusional languages are a group of languages where they tend to inflect words to express grammatical features such as the person, gender, and number features. Inflexions can be constructed with an affix (prefix, suffix, or even infix) or as a vowel change. For example, the word *cats* is inflected with a suffix for the number feature to indicate the plural form of a noun. Words are often inflected by at least one free morpheme and at least one bound morpheme. Free morphemes can stand by itself (e.g. “cat”) while bound cannot (e.g. “-s”).

Because of their tendency to inflect words, POS tagging text in inflectional languages is usually hard. A typical problem is substantial lexical data sparseness due to the high number of possible inflexions of a single word. To reduce sparseness and number of Out-of-Vocabulary (OOV) words, inflected words are often segmented before or in parallel with POS tagging. However, the segmentation process is prone to errors. Inflexion boundaries are often not marked which increases the number of homographs (two or more words spelt in the same form but with different POS tag or pronunciation (e.g. due to differences in diacritisation). Some orthographical changes are caused by inflexions, making it hard to recover the original word form. As a result, a segmentation process sometimes fails to detect morphemes.

Wasim is a web-based tool for semi-automatic annotation of text for gold standard corpus production. It was developed for the annotation of our Sunnah Arabic Corpus (SAC) (see Chapter 7), a collection of classical Arabic sayings ascribed to the prophet Mohammad. It has also been tested in four case studies.

The tool features high flexibility in segmenting tokens, editing, diacritizing, and labelling tokens and segments. It can be integrated with morphological analysers to ease the annotation by selecting from its proposed analyses. It aims to speed up the annotation by entirely relying on a keyboard, with no mouse interaction required. The source-code is released under the MIT license, which means it is free to use, copy, or modify for any purpose, including commercially.

9.2 Motivation

Recent research developments in, and uses of, Arabic annotated corpora were the main inspiration behind this tool. These uses have allowed these corpora to play a growing role in some linguistic and computational research areas such as part-of-speech tagging, segmentation, and diacritisation. Additionally, the lack of freely available annotated corpus of classical Arabic increases the importance of creating such a resource, which may encourage researchers to conduct more studies in the aforementioned research areas.

The chapter aims to develop an open-source language-agnostic annotation tool for textual corpora that is *efficient* in terms of time and accuracy. The annotation of Arabic text is more tedious and time-consuming than its equivalent in poor morphological languages. The development of Wasim increased the efficiency of the annotation project of the Hadith Arabic corpus, which aims to annotate about 80k words of classical Arabic text with morpheme-based POS tag, lemma, morphological features in addition to adding missing orthographic vowels of the text (diacritisation).

For the project, we analysed the required set of features needed for annotating SAC and used these as criteria in a survey of existing tools. Morphosyntactic annotation of SAC (and other highly inflectional language corpora) requires additional specialised functionality:

1. Segmentation of one word into a set of segments
2. Addition of orthographical accents or diacritics
3. Listing a set of solutions from a lexicon dictionary (internally or externally using a morphological analyser)
4. Consistency validation and integrating annotation guidelines (e.g. homographs).
5. Adaptive prediction based on historical tagging
6. Efficient keyboard-based navigation and labelling

9.3 Major features

The annotation of text in a highly inflectional language is usually harder because:

1. Words are highly ambiguous, which results in many homographs (i.e. more need of a lexicon),
2. Words need to be segmented into a set of morphemes, and

3. As a result, taggers performance is usually poorer and mostly rely on a lexicon or a morphological analyser to improve the accuracy.

Semi-automatic annotation should help to remove the ambiguity of words as it should be able to correct tagger errors. Many times, these errors are in the ranking of the solution set provided by the morphological analyser. Therefore, the most needed feature is the integration of a morphological analyser, which allows the annotator to re-select the proper analysis in case of incorrect automatic tagging.

In addition, an efficient way to segment words into a set of morphemes is a necessity. For example in Arabic, one word in six words is inflected, and an inflected word (multi-word token) consists of an average of 2.06 syntactic words (or morphemes)[3].

9.3.1 Morphological Analyser Integration

Wasim integrates with morphological analysers to speed up the process of annotation. Morphological analysers take a word as input and produce a list of possible analyses (which include word's segmentation and lemma and segment's POS tag and features). By providing a set of possible analyses, Wasim allows annotators to select one analysis. Once a solution is chosen, all its values of POS tag, lemma, segmentation, and morphological features will be reflected in the word analysis. The chosen solution can be edited though.

In the SAC project, the number of morphological features are ten features, in addition to segmenting the word into its set of morphemes and marking its POS tag. We hypothesise that it will be more efficient to select a solution instead of doing them all from scratch. However, this hypothesis depends on the quality of the morphological analyser. Annotators have to mark all features though if the analyser returns no results. Once a newly-created analysis is detected, it will be saved in the server for possible later requests.

Wasim provides two ways of morphological analyser integration: first, using an embedded supplementary tool that acts as a pure lexicon memory, it reads the annotated part of the corpus and index words with their annotations. Then, it allows HTTP requests to be made from Wasim, and it will return all possible solutions of the token in hand.

---

[3]http://universaldependencies.org/treebanks/ar-comparison.html

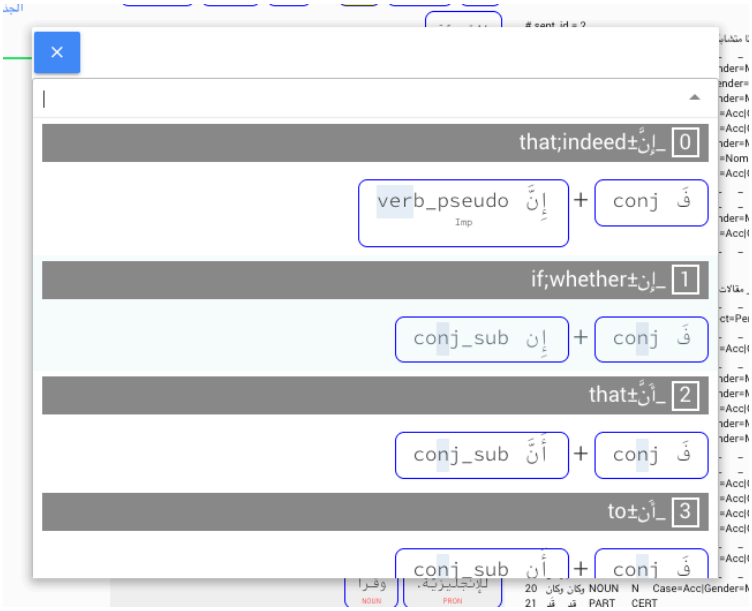


Figure 9.1 The list of possible solutions from a morphological analyser. A solution is usually a bundle of POS tag, segmentation, lemma and morphological features. Selecting one solution will replace all its content to each proper annotation field.

Second is using an external morphological analyser. Analyser outputs must be in CoNLL-U format with word id in the MISC column that maps to the original word index of the submitted sentence (e.g. WID=2). The reason is to allow Wasim to group the MA's analyses of one word.

A mapping between the MA's tagset and the project tagset may be required, and this can be easily defined in the configuration. If the mapping results in an ambiguous tag in the project's tagset, Wasim will duplicate the analysis for each possible tag. For example, if "NOUN" is mapped to PN and N, two analyses will be presented to the annotator.

9.3.2 Consistency Reinforcement

Consistency (a.k.a. "stability" when measuring the consistency of one annotator alone over time) of corpus annotation process is critical to ensure that all annotators in all texts follow the same procedure of annotation over time. High consistency means very little disagreement in the annotation, and this helps to train machine learning algorithms successfully.

To increase the consistency of the segmentation and tagging of a corpus, Wasim followed three approaches: First, it allows the use of automatic POS tagger.

Second, it integrates with morphological analysers. Third, it generates a list of common homographs periodically. Homographs are associated with their possible POS tags and segmentation. Possible segmentations are only shown when the token in hand is a homograph.

Usually, in annotation guidelines, there are some guides of specific words, usually homographs. However, in highly inflectional languages, those homographs are overwhelming, and such offline guideline may miss some homographs, or guidelines document will be lengthy. This feature serves as an online guideline for annotators, which is automatically built up.

In the segmentation layer, Wasim warns the annotator when a segmentation of a word differs from previous segmentation of the same word. If the annotator insists, its new segmentation will be added. A similar process is happening for morphological tagging.

The list is generated periodically from the annotated part of the corpus, and the possible segmentations/POS tags of homographs are kept. Each homograph will have a set of examples in context for each sense. Moderators can edit the list, and add guideline notes of tagging such cases. The list will appear in Wasim with its notes when selecting a word in the list. See Table 9.1 for example.

Table 9.1 Example of ambiguous part-of-speech helper.

Ambiguous Word	**POS tag**	**Frequency**	**Example**	**The reason for choosing POS tag.**
mina	P	78%	wamaA < unozila mino qabolika wabiAlo|xirapi	When it is a preposition followed by a genitive noun.
			EalaY hudFY mino rabi~himo wa<uwla}ika	
	REL	16%	wamina Alna~Asi mano yaquwlu |mana~A	When it means a relative pronoun “Al*y”.
			<atajoEalu fiyhaA mano yufosidu fiyhaA	
	…			

9.3.3 POS Tagging Integration

Instead of starting the annotation process of a corpus from scratch, Wasim integrates with UDPipe to provide a kick start in the annotation process. UDPipe delivers trained models for more than 60 languages that tokenise, tag, lemmatise and dependency parse raw text and save results in CoNLL-U formatted files. UDPipe can be trained on the part of the corpus that has been annotated as well. Other tools can be used as long as they generate CoNLL-U formatted files. For Arabic for example, Sawaref, Madamira, Stanford, Farasa and AMIRA tools can all be used (translation into CoNLL-U format can be done using Sawaref tools).

9.4 Data Representation

Wasim follows the Universal Dependencies v 2.0 (UD)[4] (Nivre *et al.*, 2017) in the same way it represents sentence segmentation, POS tagging, morphological features, segmentation, and lemmatisation. In the UD project, segmentation of text is based on a "lexicalist view of syntax". Texts are segmented into *syntactic* words; this should not be confused with phonological or orthographical words. That means clitics like CONJ should be separated from VERBs, even though they appear in the same orthographic word (with a space boundary). However, in this chapter, syntactic words are called *tokens* and orthographic words are called *words*.

UD does not have a standard for diacritisation. Wasim follows its own representation of diacritisation of Arabic (see Section 8.9). We enforce such representation by performing a series of transformation using "regex" expressions[5]. Moderators can implement a similar approach for other languages.

9.5 Tool Description

The Wasim tool has mainly two components: a *front-end* interface which allows interacting with annotator and provide warnings and feedbacks, and a *back-end* server that manages sessions and storage of CoNLL-U files.

The front-end web-based tool is built using the Ionic framework using the Typescript/Javascript programming language. The main screen for document annotation () contains four sections: 1) A toolbar at the top is used for warnings and

[4] http://universaldependencies.org/

[5] A regular expression, or regex is a favourite way to define a search and replace pattern.

helpful shortcuts. 2) The middle column shows the words in small boxes (with its XPOS tag and lemma beneath it) with the current word in process highlighted in a different colour. Multi-word tokens show their morphemes linked by a “+” symbol. Instead of displaying words in a tabular format (like in CorA, SAWT), we display words in natural paragraph flow, allowing the annotator to examine each word's context easily. 3) The left column shows key-value pairs of the lemma, and morphological features. 4) The tab-based right column shows the synchronised CoNLL-U format of the current document, and some useful statistics about the document. Closed features are a dropdown list with an auto-complete feature. shows a screenshot that shows the main components of Wasim.

The screenshot shows the annotation page for one document. The middle part represents one sentence where each box is a token (with its XPOS tag). The left side shows feature annotation. The top bar represents file-level actions including advanced search engine, previous annotations, save to the cloud, download to the drive, undo and redo actions. On the right side, CoNLL-U synchronised representation of the sentences is presented for the current sentence.

CoNLL-U representation on the right side is editable at any time, as Wasim synchronise changes. Changes will be validated, and errors are reported in an error log box below it. In case of valid changes, such changes are reflected in the Wasim widgets. Wasim give an option to the annotator to make changes in bulk like copying previous annotations, though it should be rarely used.

Three useful subviews are displayed on demand: **A.** a list of other alternative solutions retrieved from a morphological analyser. **B.** a tabular format of morphological features and possible values. **C.** a segmentation view that allows segmenting words easily. The front-end of Wasim can be seen as a CoNLL-U file editor: it parses the file, validates the syntax and visualises the sentences with a synced side by side view of CoNLL-U file.

The back-end is a server operated using Node.js Express server, and is responsible for authentication and managing annotated and raw files. A connection with the server using WebSocket is established for the several reasons: such as morphological analyser requests, sessions, diacritisation requests, and temporary session backup.

Figure 9.2 The main screen for document annotation.

Each project is a folder in the system that contains document files, configuration files, a database of homographs and a file of the corpus lexicon. It manages the versioning of files using the popular Git version control system. Git system tracks all the changes that are made to files, and allow multiple operations, e.g. *diff* to show changes to a file in the colourful interface. Annotated documents are moved to a subfolder.

All annotations are stored in CoNLL-U format as plain text files. Accessing one file from an annotator will grab a copy of that file; however, this might allow other annotators to work on the same file. To prevent this, Wasim implements a simple lock system where a file is locked while a connection is maintained with the server (using WebSocket). We only release the lock if the annotator accessed another file or the connection is closed.

Wasim is designed to be configurable to support preferences and project related setup. Project setup includes its name, language, remote Git repository, UDPipe model, morphological analyser path and several other preferences. Projects must define their own fine-grained tagset (unless UD tagset is used), with their morphological features. Wasim allows custom key-binding for actions. The configuration files are saved in the project level as JSON files.

The annotation process can be entirely manual or semi-manual. In the case of semi-manual, the corpus is first tagged using UDPipe models. Automatically generated tags can be then checked and manually edited using Wasim. In the next section, we will describe the supported morphosyntactic layer in more details.

9.6 Morphosyntactic tasks

Wasim provides an easy interface for the annotation of six tasks. While these tasks can be processed sequentially, we allow annotators to work on any of the tasks at the same time. Tasks sometimes are interrelated, e.g. if the automatic tagger produced the wrong POS tag, it might also produce the wrong morphological segmentation or lemma. Since Wasim uses morphological analysers, if the annotator chose one solution, it will affect multiple tasks at the same time. Therefore, we allow the annotator to edit previous tasks without leaving the screen. However, we expect the annotator to use the MA feature at the beginning of the word segmentation, diacritise then segment the word, mark POS tag, and finally mark morphological features.

Since Wasim allows annotation of text on many levels at the same time, the annotator might skip a task accidentally. Wasim provides a guide to go through tasks in keyboard mode. It highlights tasks sequentially to grab the annotator's focus on the current task.

However, depending on the corpus annotation goals and preferences, the annotator can customise the view; e.g. deactivates one/multiple tasks, or disables CoNLL-U view. The annotator can write post-process rules to check the validity and consistency of different tasks as well as constraints on different tasks.

Wasim is designed to increase productivity for these particular annotation tasks while sacrificing some amount of simplicity (many shortcuts/buttons on the screen). While the learning curve (the rate of a person's progress in gaining experience) is steep, we hypothesised that Wasim features would improve the time required for annotating one word.

9.6.1 Morphological segmentation

Inflectional languages tend to inflect morphemes to express different grammatical features. Unlike many other annotation tools, we do not assume the text to be tokenised/segmented. Annotators can easily tokenise words by editing their forms. Word can be segmented as well by placing a pointer in the proper position and inserting a particular character ("+" sign by default). The two generated morphemes will clone the data from the original word except for its form which will be divided. The multi-token form will remain the same though. The original word in the main screen will be replaced by two morphemes linked by "+" symbol. The annotator can remove segmentation by simply hitting the "backspace" button in one morpheme, and it will merge to the previous morpheme.

With the integration of morphological analysers, annotators should mostly select the proper segmentation/tagging from its provided list. Manually segmenting one word should be resorted to as a last choice, the case when there is no proper segmentation.

Since we follow CoNLL-U representation, UD representation keeps the form of both the word and the token in its two-level indexing scheme. The form of one token can be rewritten as if it was not inflected. Free morpheme form can be altered because of the inflexion, and annotators can recover its original form, e.g. "John's"

can be recovered to either “John+has” or “John+is”. The original form (John+'s) will be written in the MISC column.

```
1-2   John’s     _     _     _     _     _     _     _     _
1     John       _     NOUN  N     _     0     _     _     _
2     has        has   AUX   BE    _     0     _     _     ORG=’
```

Unlike other formats, the format as illustrated above keeps two forms of one morpheme: inflected form (e.g. John’s) and free form (e.g. has).

9.6.2 Diacritisation

A diacritic (sometimes called accents or short vowels) is an optional small glyph added to letters to change the sound of the letter. Diacritisation is the process of adding those glyphs. In our Sunnah project, we asked for this addition as diacritics reduces the ambiguity of words.

This process is tedious as it requires to add diacritics for each letter. Since the number of the possible diacritisation patterns is low, we enable the use of morphological analysers to generate the possible diacritisation of a word. The annotation process is then eased by only selecting the correctly-diacritised word. The annotator has the ability, though, to edit the form if no solution is provided.

Additionally, Wasim uses a diacritisation tool (Alosaimy and Atwell, 2018) that borrows more thorough diacritisation from similar context (see section 5.6 for details). This method is different from significant diacritiser as it does not “guess” diacritisation, but rather “borrows” it if it exists from a similar context. Context can be defined in different ways: e.g. n-word gram.

Wasim allows moderators to enforce some standard on the diacritisation. For example, in Arabic, it can be configured to ignore diacritisation of letters proceeded by a long vowel. These transformation rules can be enforced using a set of regular expressions (regex). These rules will only be applied to a subset of morpheme/words that conform to certain conditions. For example, in the guidelines of SAC, we require no diacritisation on the Lam letter of the definite article “Al-”. We had a rule that removes such diacritisation of the subset morphemes that has a POS tag: DET.

9.6.3 POS tagging

POS tagging in Wasim is morpheme-based. We assume that the tagset is assignable to any morpheme regardless of its location (e.g. prefix or base). Tags can be easily chosen from a list of POS tags ordered by their frequency or

alphabetically. The most common POS tags are shown at the top, and pressing its associated number will assign it to the current in hand morpheme.

9.6.4 Morpheme-based morphological features

Morphological features can be easily marked through a popup that offers a single input line for all morphological features together. This popup offers keyboard navigation to select the features. It also acts as a search input, so that only features that match the input text is visible.

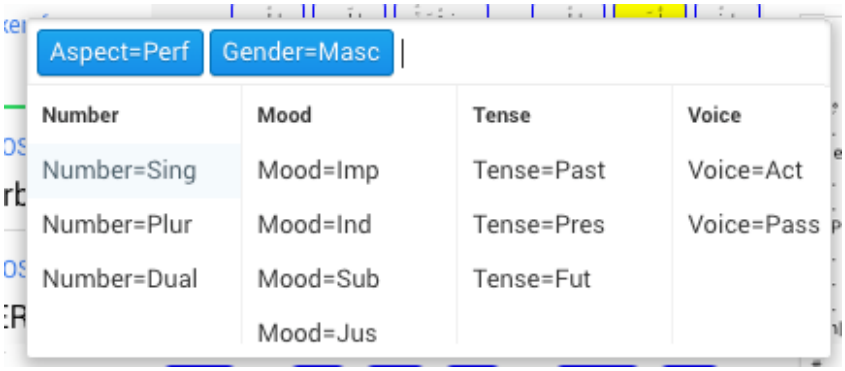


Figure 9.3 Features annotation popup one-line input with an auto-complete feature of a VERB token.

Only the subset of morphological features that is compatible with the segment's POS tag is shown (see Figure 9.3). For example, “Mood” is only shown with VERBs. The compatibility table is configurable, and by default, we used the compatibility of the UPOS tag and UD morphological features.

Once the input gets the focus of the user, it shows a drop-down list of all possible values. Once a value is selected (e.g. “MASC” for gender), other incompatible values hide accordingly. The goal is to speed up the annotation by selecting values in one place and asking for relevant morphological features only.

9.6.5 Lemmatisation

Wasim offers a simple interface for lemmatisation. If it is integrated with a morphological analyser, the lemma of the chosen solution will be assigned. The lemma, however, can be edited manually.

9.6.6 Sentence Segmentation layer

Wasim provides the ability to alter the text and separate one sentence into two. By convention, the CoNLL-U format leaves an empty line as an indicator of

sentence start/end. Also, words of each sentence are numbered orderly starting from 1. CoNLL-U also allows the tagging on the sentence level by allowing comments at the beginning of the sentence. These are reflected in Wasim, and the ID numbers of words will be reordered. Sentences can have multiple tags, and a tag can be assigned to sentences.

9.7 Case Studies

We provide four case studies to show the use of four languages. In each case, we evaluate one major feature and the effect of that feature on the speed and accuracy.

In each case, we annotate a couple of sentences (an average of 70 words) depending on the target language of the case. While the text size is small and might not clearly show the improvement, these experiments are for illustration purposes rather than to actually measure the difference. The annotator who has done these four experiments is the author of the tool; therefore, most of the effect of the learning curve is excluded.

For each case, the text is divided into two halves, H1 and H2, and both halves are tagged twice (two rounds). In all cases and for both rounds, the annotator is the same person. Both halves are tagged with the feature enabled (F=True) and then disabled (F=False) but in a **different** order for each half. The steps are $\{H1_{F=True}, H2_{F=False}, H1_{F=False}, H2_{F=True}\}$, and the first two steps are in the first round. In the last two steps, the annotator already knows the texts and should annotate it faster. However, the results between step 3 and 4 are comparable as the word counts are similar.

In Arabic cases, we used the QAC tagset and asked the annotation to follow its annotation guidelines. UDPipe is trained as well on the Quranic Arabic Corpus (Dukes and Habash, 2010) (converted to CoNLL-U by the author and available here[2]). The morphological analysers used here is MADAMIRA, and its results are parsed and converted to CoNLL-U format using Sawaref toolkit. A manual mapping from MADAMIRA tagset to QAC is defined and used.

Time is used as a metric for efficiency. The Intra-rater reliability is high in all cases which shows that using features does not affect the accuracy. Mismatches

[2] https://github.com/aosaimy/sawaref-data

between the two rounds are reviewed and corrected in a third round. The accuracy concerning the fraction of correctly annotated words is then used to evaluate the correctness of the two rounds compared with the gold standard (third round). More metrics are reported per case requirement. In all cases, we only evaluate the accuracy of segmentation and POS tagging, although all tasks are done. Diacritisation, lemmatisation, and other features accuracy are not included. In the end, we show brief statistics on our Sunnah Arabic Corpus Annotation.

9.7.1 Modern Standard Arabic and Morphological Analyser

In this case, the annotator used the morphological analyser to select one candidate analysis from a list of proposed analyses. “Uses of MA” report the case of annotators selecting an analysis even though such analysis was corrected later. We report the number of times that the annotator used the MA and the number the proposed analysis is edited. Clearly, the results show that using MA is helpful in speed and accuracy, but in most cases, it is prone to errors. Using MA improved the annotation accuracy and speed significantly. All the texts used in these experiments are appended to the thesis.

Table 9.2 Comparison between using and not using MA in accuracy and speed.

	Using MA		**Without**	
	Step 1	Step 4	Step 2	Step 3
Word count	50	51	51	50
Morphs count	72	70	70	72
Accuracy	96%	100%	84%	84%
Time (secs)	1358	635	1819	1729
Time (s/m)	18.86	9.07	25.99	24.01
Uses of MA	39	43	-	-
Number of edits	30	31	-	-

9.7.2 Quranic Arabic and Consistency Reinforcement

In this case, we show how the warning and helper guidelines help to improve the accuracy. The consistency Reinforcement feature used the whole QAC corpus to build the list of homographs and their segmentation and tagging. We report the number of homographs that has been displayed on the screen, Table 9.3. 5-8 out of 25-24 morphemes shows that homographs in the Quranic Arabic (a case of highly inflectional language) is relatively frequent. Using this feature increased the

accuracy in step 3 compared to step 4. We expect the effect of this feature to be more apparent in large corpora.

Table 9.3 The accuracy and speed when using CR feature.

	Using Consistency Helper		Without	
	Step 1	Step 4	Step 2	Step 3
Word count	15	16	16	15
Morphs count	25	24	24	25
Accuracy	100%	100%	100%	93%
Time (secs)	269	278	331	284
Time (s/m)	10.76	11.58	13.79	11.36
homographs	5	8	-	-

9.7.3 Sunnah Arabic and Keyboard Navigation

In this case, the annotator does not use the keyboard for navigation. He can use it for typing in the correct form or segmentation. We also report the number of mouse clicks vs the number of uses of keyboard shortcuts. Table 9.4 shows that using keyboard shortcuts reduced the annotation time by about 30% (9.34 vs. 6.89 and 8.3 vs. 18.3), even though the number of presses are higher than the number of clicks.

Table 9.4 The accuracy, speed, keyboard presses and mouse clicks comparison with two modes.

	Using Keyboard		Using Mouse	
	Step 1	Step 4	Step 2	Step 3
Word count	31	30	30	31
Morphs count	38	37	37	38
Accuracy	100%	100%	100%	100%
Time (secs)	355	307	677	262
Time (s/m)	9.34	8.3	18.3	6.89
Presses/clicks	131	166	147	87

9.7.4 English and UDPipe

In this case, we used a trained model of Linguistic Data Consortium English Web Treebank LDC2012T13 to kick-start the annotation process. We compare the process of assigning (only) POS tags and show that Wasim is language agnostic and can work for left-to-right languages as well. Since the text excerpt is too small, we do not show the effect of using an adaptive training UDPipe model. Table 9.5

shows that semi-automatic tagging advances the speed of the general annotation task. This is, however, highly dependent on the quality of the automatic tagger.

Table 9.5 The effect of using a tagger (semi-automatic vs manual annotation)

	Using Tagger		Without	
	Step 1	Step 4	Step 2	Step 3
Word count	31	30	30	31
Accuracy	96%	100%	96%	90%
Time (secs)	67	47	170	203
Time (s/w)	2.16	1.57	5.67	6.55
No. of Edits	0	0	1	3

9.7.5 General Case: Sunnah Arabic Corpus

Wasim is used as well for the project of morphological annotation of SAC. So far, words have an average of 1.3 morphemes, and we spend 10.9 secs/morpheme on average to annotate a morpheme with all features enabled[3], i.e. 9.17 morphemes per minutes.

In SAC, the speed of the annotation is rising over time due to two reasons: the automatic tagger become more accurate over time, the annotators are gaining experience. Apparently, the speed of annotation depends on several factors such as text, language, course vs fine-grained tagging, and annotator experience. Therefore, reported speed measures should be taken with caution.

9.8 Wasim vs other annotation tools

The comparison with other tools needs similar experimental settings in all of the annotation tools. However, morphological annotation is known to be time-consuming and costly, so repeating the same experiment was not an option.

The authors of MADARi, however, reported a similar (but not identical) experiment. They used their tool on annotating one dialectical corpus. The task is divided into two tasks: spelling corrections and morphological tagging. Morphological tagging involve tokenisation, POS tagging, lemmatisation and English glossing and the annotation rate is 277 words/hour or 4.61 words/min. This is not directly comparable to Wasim experiment with annotating SAC (average rate of 7.05 words/min) as SAC used classical Arabic (vs. dialectical), did not lemmatise

[3] Features include POS tagging, segmentation, and six morphological features.

nor provide English gloss of the words, and did not have a look into the corpus before (vs prior spelling correction step).

9.9 Wasim Front-End Modular Design

Wasim is implemented using version 3 of Ionic Framework[4]. Ionic is a modular design for mobile and web application. In our case, Wasim has mainly three types of modules: providers, components, pages. Wasim is self-packaged which make it easier for others to install. shows the overview design of Wasim.

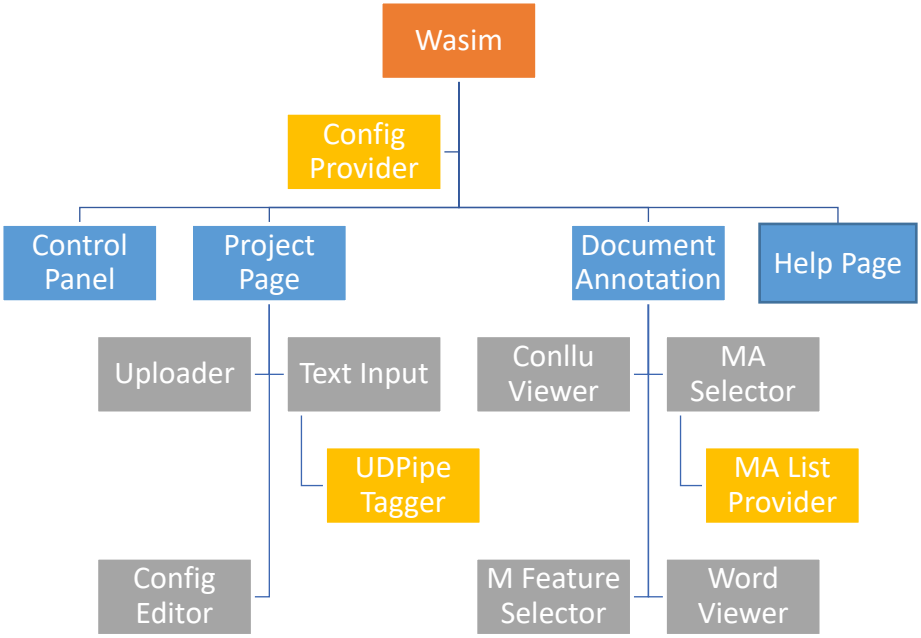


Figure 9.4 The overview design of Wasim.

Wasim has four pages: a control panel for managing all projects and is only authorised to the Wasim administrator (). The project page is used for managing project documents and properties (and Figure 9.7). The last page is the main page for annotation (). Project pages can be shared with other annotators by a direct URL link.

[4] Ionic is an open-source free SDK for developing native and progressive web apps using familiar web technologies (HTML, CSS, and JavaScript). https://ionicframework.com/

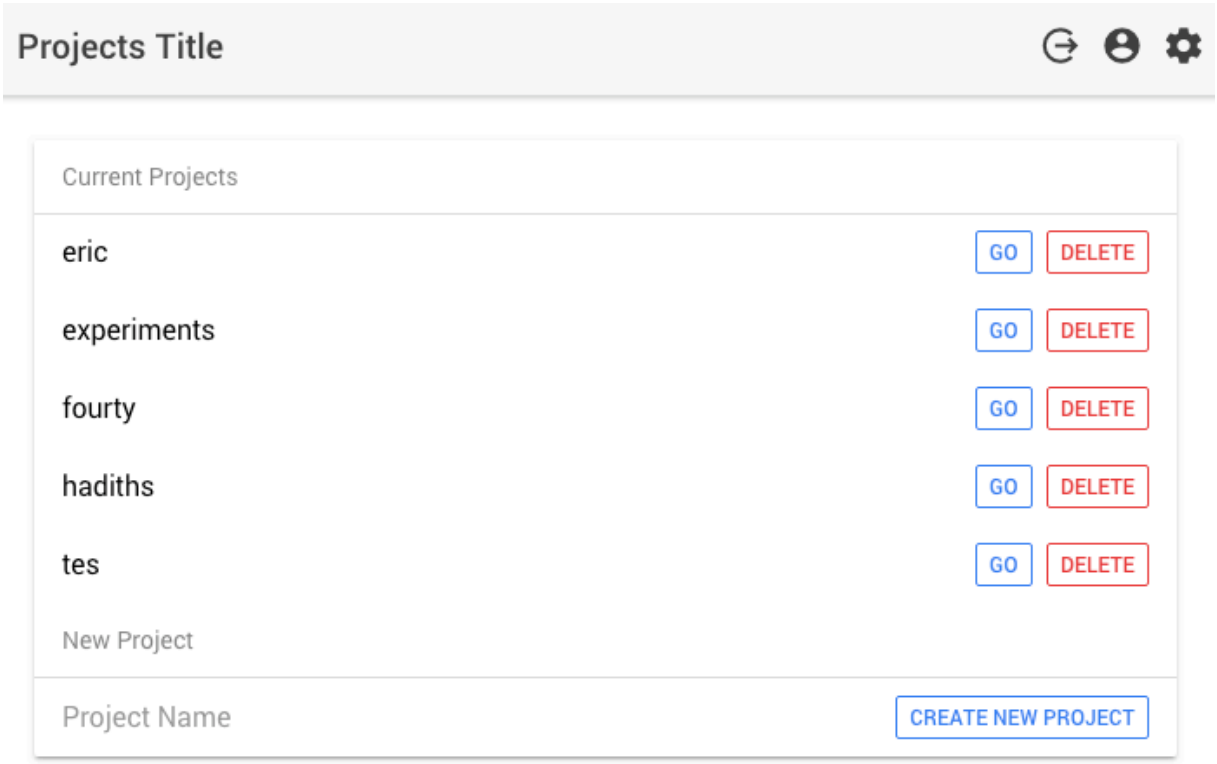


Figure 9.5 The page for managing top-level projects.

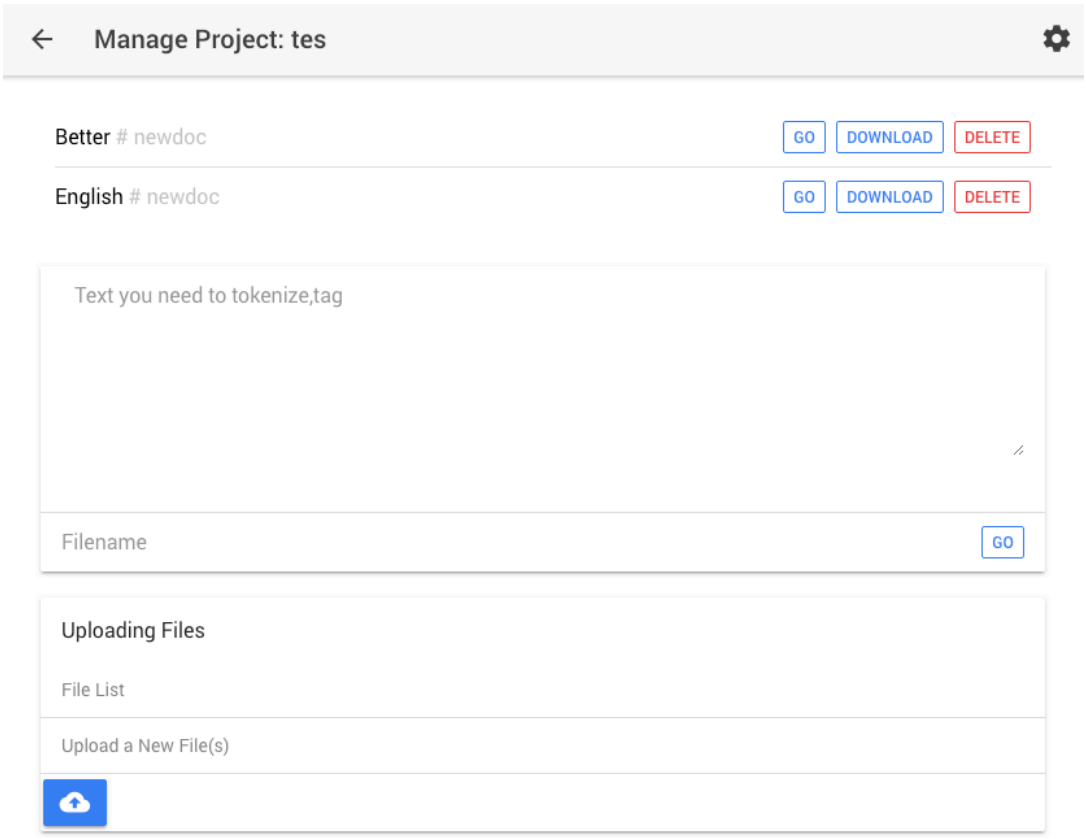


Figure 9.6 The page for managing Project's documents.

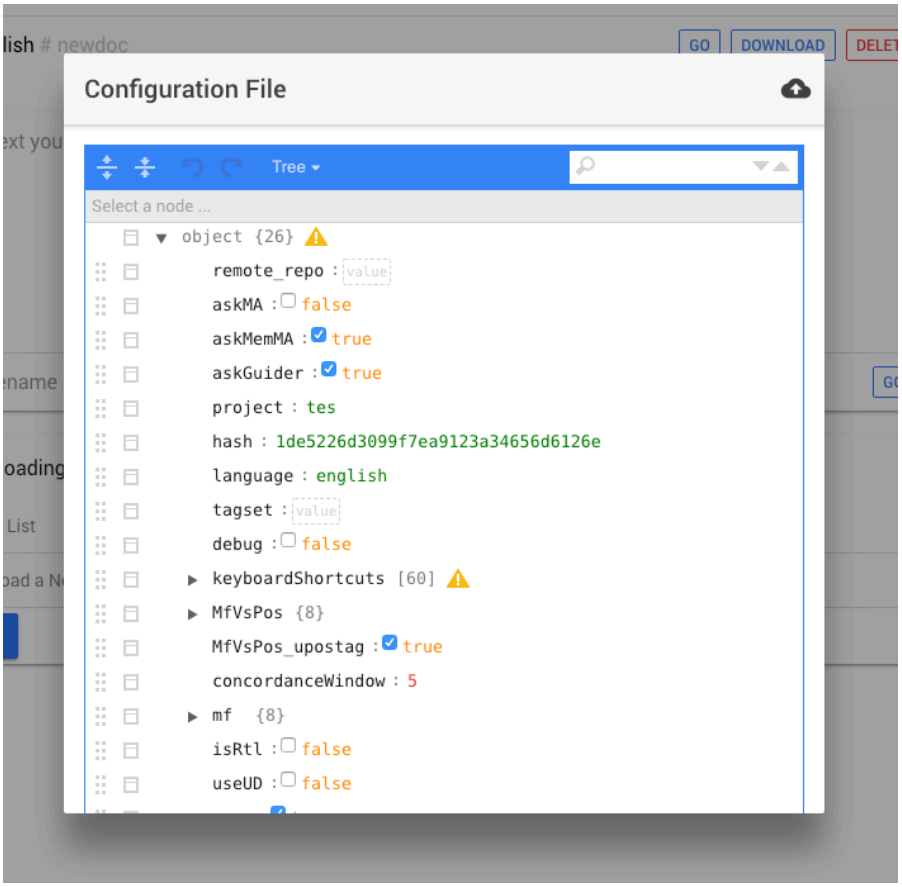


Figure 9.7 Project's Settings Editor

9.9.1 Control Panel

In this page, an administrator with the proper authentication can control the projects and users. When a new project is initiated, a folder in the server is created, and *git* repository is initiated to track all changes to the project.

9.9.2 Project Page

In this page, a list of all documents is shown and searchable. Project documents can be downloaded as a CoNLL-U formatted document. Project moderator (with proper authentication access) can manage the documents and change project's settings as well. Project settings include customising keyboard shortcuts, project language, access credentials, MA settings, POS tagset, morphological features, and the mapping from MA tagset and values to project settings.

9.9.3 Document Annotation page

The Document annotation page has four main components: 1) morphological analyser selector, 2) morphological feature selector, 3) word sequence viewer, and 4) manual CoNLL-U format editor.

The first component is the morphological analyser selector. A text input for listing the morphological analyses is used which allows the annotator to search quickly through the list using the POS tag, morphological feature, form, number of segments and lemma. Once a text is entered, the list will be filtered to only those that match the input.

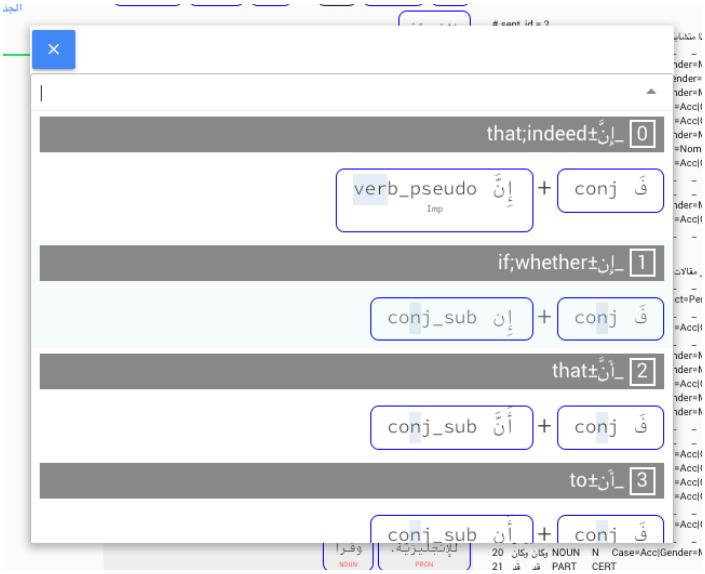


Figure 9.8 illustrate the use of the component when tagging one word in Arabic. It shows as well the search functionality.

This component contacts the morphological analyser by the MA list provider. It is used to send requests (an HTTP request) to the morphological analyser and handle its response. The response is a slightly modified CoNLL-U format of the sentence. We add the position of the word in the sentence as a value to the miscellaneous column. The word provider then assigns each word in the sentence with the list of analyses that match word's position.

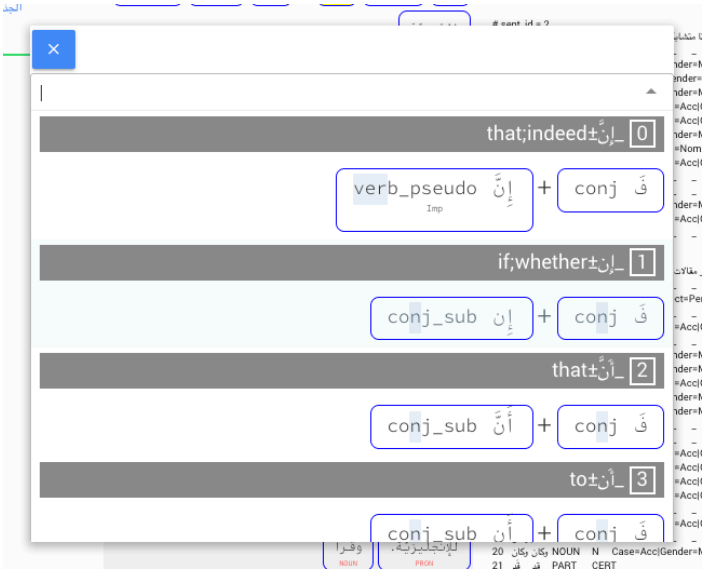


Figure 9.8 Morphological Analyser selector component.

The second component is the morphological feature selector. It allows the annotator to quickly choose the correct value of all morphological features that are compatible with the chosen POS tag. Similar to the last component, it filters the list once a text input is entered. Morphological features are ordered in columns, and once a value of a feature is chosen, the column disappears.

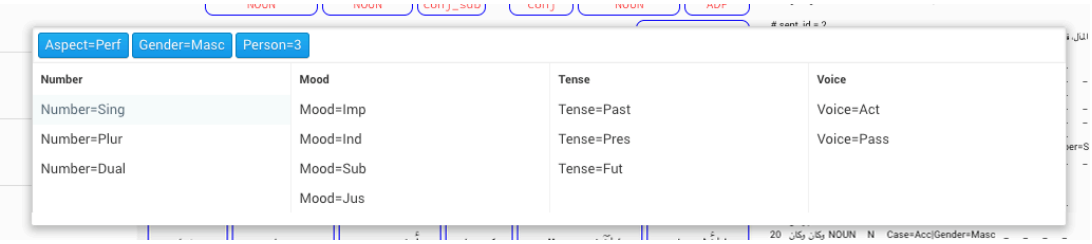


Figure 9.9 Morphological feature selector component.

The third component is the CoNLL-U viewer and editor. This component periodically syncs the internal representation of document to its representation in CoNLL-U format. It allows the annotator to double check that their edits are reflected in the CoNLL-U representation. It allows manual editing of the CoNLL-U text. Edits are parsed and validated to make sure it does not violate the formal format rules[5] using a publicly available validator and parser[6].

[5] http://universaldependencies.github.io/docs/format.html

[6] https://github.com/spyysalo/conllu.js

Figure 9.10 CoNLL-U viewer and editor.

This component maintains an internal object-oriented representation of the document. A Document object has an array of Sentence object. A Sentence object has an id, comments (both parsed from the comment at the beginning of the sentence) and an array of Elements. An Element can be one of several types: Multiword, word, or a segment (part of a multiword element). A Multiword has only a form and range of segments ids with reference to its segments. A Word has an id (used later for syntactic representation), form, lemma, two levels of POS tags (universal and detailed), a list of key-value pairs for morphological features in addition to three properties for syntactic. A Segment is a Word except the form is modified as it will appear if not attached and a reference to its parent (a Multiword object) and the segment's position in the parent is saved. The original text of the sentence can be covered by the form of Multiword and Word elements.

The last component is the main viewer of the annotator. A sentence is represented as a sequence of words. If a word is a Multiword object, its segments are shown separated by a plus sign. One word is active, and its list of properties (lemma, form, parent form (if a segment), morphological features, etc.) is shown on the side. Each sentence is separated from the next sentence.

9.10 Wasim Back-End Design

The backend part of the tool has two major components: document management, morphological analyser, tagger (UDPipe), result parser, and result mapper.

Wasim used the Git tool for managing documents and versioning. Git[7] is a free and open-source system that track changes of files (or documents). In Wasim, we use Git for several reasons: It offers off-the-shelf file tracking and maintenance.

Wasim clones the original repository for every user. For every save to the document, Wasim pushes the changes to the remote repository.

Besides, we use Git to show the difference between two annotations using the git diff subcommand. This feature is handy for project moderators, as it allows a simple curation. The differences are highlighted, and one can be chosen as the best human analysis.

9.11 Conclusion

We presented Wasim, an open-source web-based tool efficiency-oriented for semi-automatic annotation of inflectional languages resources. It supports multiple tasks including segmenting tokens, diacritizing and labelling tokens and segments. It is integrated with UDPipe to kick-start the annotation process. It can be integrated with a morphological analyser to speed up the annotation process.

For future work, we might add additional layers for syntax, co-referencing, and named entities. We also might as well support other formats (e.g. XML) in the future. Unlike deterministic (i.e. one-tag) morphological taggers, morphological analysers and lexicon, which produce multiple possible solutions, do not have an official format for encoding results. However, recent work done by More *et al.* (2018) , which encodes morphological analyses as lattices, seems appealing. We might consider adapting this format in future or any other standard format, if it is adopted by the UD community.

[7] https://git-scm.com/

10 CONCLUSION

10.1 Overview

The Classical variant of Arabic has received less attention in the field of Arabic NLP. Although it is considered the father of Modern Standard Arabic (MSA) and it has wide liturgical usage by Muslims around the world, this variant is under-resourced and underexplored, especially classical texts beside the Quran.

Most approaches have involved the development of new corpora, tools, and standards for classical Arabic. These approaches are limited in terms of resource size, because the creation of new resources is usually costly. Some approaches ease the contribution of the public and allow them to be involved in the resource development, but it shows that the quality of this approach is not optimal. Generally, the development of new resources is usually costly in terms of money and time. We aimed in this thesis to follow a different approach: reuse available resources in MSA. MSA and classical Arabic share many aspects in the language, and not benefiting from these resources is wasteful.

Although there are a number of great and thorough resources in MSA, these resources are not optimal for classical Arabic, and need some adaptation. This thesis explored computational ways that *adopt* existing available heterogeneous resources in Modern Standard Arabic and *combine* and *adapt* them. This adoption tackles two types of adaptation: annotation-style adaptation and domain adaptation. Firstly, existing MSA taggers are not standard in their underlying linguistic theories, nor the computational implementation. They use different annotation tagsets and adversarial segmentation schemas. They are implemented as well in different programming languages, and their output format is not standard. Secondly, these resources are trained to be used optimally in MSA language. The two languages have different

distributions, and we aim at learning from the MSA distribution an accurate model for classical Arabic.

The first part of this thesis explored and surveyed the literature and implemented a systematic way for evaluating available MSA taggers. Although the results should be taken cautiously, this evaluation reaffirmed other works in the literature: classical Arabic texts vary greatly from newswire data, i.e. MSA. The drop in the accuracies of these taggers varies from 10% to 20%.

Using best-scoring taggers in the first part, we designed a systematic approach that combines and adapts several taggers to other domains and annotation-styles. This systematic approach is done through several stages of format standardisation, tagset and segmentation conversion, and advanced techniques for prediction and disambiguation. Each stage has its own challenges and different techniques were compared and contrasted.

Several experimental studies were conducted throughout the thesis. One experiment showed that cross mapping of tagsets is mostly n-to-n and tagsets cannot be easily contained or mapped to one very-fine-grained tagset. Another successful experiment utilised freely available naturally-annotated texts to reduce the ambiguity level by increasing the diacritisation level of words. A third experiment illustrated how tagsets are not compatible in a case study of tagging adjectives.

Although the ensemble of heterogeneous taggers is shown to be challenging, especially when used to adapt to another domain, an ensemble of four MSA-based taggers that uses a relatively small corpus (~25k) for adaptation is found to be effective in terms of robustness and accurateness. The best ensemble method does not require prior alignment rules and scored an accurate POS tagging (90.2%).

The third part introduced two contributions to the Arabic linguistic resources. A new data resource is publicly released, the 144K words morphosyntactically-annotated Sunnah Arabic corpus (SAC) where 5k of the corpus is annotated manually using an extended QAC tagset. In addition, an open-source annotation tool that aims to speed up the tedious annotation process through four major features is introduced and shown to be effective.

10.2 Thesis Achievements

At the commencement of this thesis, we aimed to answer three research questions:

1 Do MSA-based taggers perform well on CA texts? Can the annotation of CA benefit from existing MSA or unsupervised resources?
2 Is it feasible to transfer knowledge from MSA-based taggers to tag classical Arabic texts through combining heterogeneous POS taggers?
3 Does aligning and mapping different segmentation and labelling schemas help ensemble taggers?

10.2.1 First Research Question

This thesis tries to answer these three questions, through novel scientific approaches. For the first question:

- A new framework that runs a comprehensive list of MSA taggers is introduced. The framework can install, run, and standardise the output of these taggers. The framework covers almost all open access and downloadable MSA taggers and analysers.
- Tested on some classical Arabic sentences and words, taggers performed below their published accuracy. Taggers differ, however, on their performance on classical Arabic and were shown to make different errors, which motivated us to combine these resources.
- An experimental study on mapping one tagger's tagset to a classical Arabic tagset shows that mapping is mostly n-to-n, and the underlying linguistic theories are different, which is illustrated on the case of tagging adjectives.
- An experimental study shows that classical Arabic texts can greatly benefit from the availability of large resources of diacritised classical texts. Word ambiguity was reduced greatly by borrowing diacritics from similar contexts. MSA taggers do not fully exploit this opportunity, and some completely drop these diacritics.

10.2.2 Second Research Question

For the second question, this thesis introduced a systematic way for adapting heterogeneous taggers by combining and exploiting them to perform well on a new domain.

There are many low-resourced languages that share many aspects with some high-resources languages, and developing new corpora without exploiting existing recourses is wasteful. However, direct usage of these resources is not practical as

they face the problem of heterogeneity. We illustrate this heterogeneity on classical Arabic texts.

- Three robust systematic ways for reusing existing taggers to a new domain with heterogeneous annotation style are introduced: *morpheme-aligned* ensemble using labelling information, *character-aligned* ensemble using form information, and *joint end-to-end ensemble* using deep learning and neural networks.
- The thesis presented the SAWAREF ensemble tagger. It is the first heterogenous ensemble tagger for Arabic that can be used with unseen texts and adapt to any arbitrary annotation style. The tagger was able to tag 90.2% of the words with their POS tag correctly. Although this accuracy is not directly comparable to other approaches due to different forms of data, language and annotation, it is higher than any adapted-form of participating taggers.
- A new one-thousand-word corpus that is tagged and aligned by different heterogeneous taggers is introduced. It can be used for evaluating and aligning the output of four taggers. This dataset can serve as well to induce mapping between taggers. The dataset is shown conveniently in tabular format and is the first of its kind in Arabic.
- The Quranic Arabic corpus has been adapted to serve a broader use of classical Arabic. We modified the orthography and morphological representation and introduced a newer version of the corpus to make the corpus more usable in the sense of machine learning.
- Since the thesis targeted general classical Arabic, a different genre of classical Arabic texts other than Quranic texts is needed. We presented the Sunnah Arabic Corpus, an annotated linguistic resource that consists of 144K words of the Hadith narratives (an utterance attributed to prophet Mohammed) extracted from Riyāḍu Aṣṣāliḥīn book (aka The Meadows of the Righteous), a compilation of 1896 hadith narratives written by Al-Nawawi and compiled on 1334. Because the morphological annotation conforms to traditional Arabic grammar, this resource should be helpful to Arabic students. The book has been studied extensively in the literature and translated into several languages, and the corpus (and its website) presents a

framework that combines all these studies coupled with the morphological analysis of the Arabic hadith.

10.2.3 Third Research Question

For the third question, we present three models of combining heterogeneous taggers. The presented models use different intrinsic ways of handling heterogeneity.

- Although some work in the literature (Hughes, Souter and Atwell, 1995; Alabbas and Ramsay, 2012b) suggested combining existing taggers using mapping rules, the mapping strategy requires a considerable linguistic background and is prone to errors. An experimental study of mapping one tagset to another performed by two students in computational linguistics confirmed that the inter-agreement of the mapping is very low.
- The study also shows that mapping tagsets with different underlying theories is not effective. However, mapping morphological features to one standard representation is shown effective in ensembles. These morphological features are sometimes extracted from complex tagsets.
- A new web-based tool for helping linguists map one tagset to another is introduced. The tool is designed to learn possible mappings by running taggers on a corpus.
- Another web-based tool is created to develop our Parallel Aligned Corpus, which allows users to align the outputs of different taggers at the morpheme level.
- We presented different ways of aligning *morphemes* of input taggers. Alignment using mapping rules extracted from aligned corpus performed the best: 96.75%, then manually crafted mappings 92.62%. However, errors propagated from this alignment hurt the ensemble tagger, especially when aligning multiple taggers. The best ensemble tagger using these methods is 88.09% accurate.
- The form-based method used a novel approach for aligning taggers using their form. The ensemble tagger that used this approach performed slightly better: 88.73%.
- This thesis introduced an *end-to-end* method that does not require any prior alignment or mapping. It not only has the freedom from manual feature

engineering, the end-to-end model is superior to other models in almost all classes.

10.3 Challenges And Limitations

This section discusses the main obstacles during the research. We were able to handle some of these obstacles and some remain limitations of the study and need rethinking of different approaches. One of the main challenges of this study is the limited ability to download, run, and standarise different tools. Many of the tools are designed in research labs and not designed as an end-user product. They are usually not documented at all or the documentation falls short in many aspects. Some tools require specific environments and libraries to run, and figuring out how to install these tools and its input-shape expectancy is often time-consuming. In addition, standardizing these valuable tools requires a thorough understanding of the tool outputs, and with the lack of documents that describe its tagset, it become very challenging.

Although we were determined to combine not only deterministic taggers but also morphological analysers, we needed to focus on deterministic taggers because of time limits. The path for combining morphological analysers should be much easier now as this project has already shown how to run, map, and standardise several analysers. The alignment part of the combination is left for future work.

Another challenge was the availability of classical Arabic corpus texts. The only corpus that is large enough and reasonably documented is the Quranic Arabic corpus at the time of the research. However, the Quranic text does not constitute a valid corpus for our final goal of annotating a collection of the Hadith narratives, because of different text, orthography, and distribution. In addition, in contrast to other annotation projects, we did not have funds for annotating the Sunnah Arabic Corpus, which required us to develop it in our spare time.

Although there are many annotation projects in Arabic, the annotation tools used are generally not accessible or not available unfortunately. The development of the open-source Wasim tool is a necessity for efficient and consistent annotation of the Sunnah Arabic corpus. Although this development was time and effort consuming, we hope that it will speed up and help other annotation projects around the world, especially projects in inflectional languages. We aim to use it extensively in other projects in the future.

In contrast, we encounter different challenges during the research which we have not addressed. As mentioned before, the alignment between taggers assumes that they are deterministic and only one label is given per morpheme. However, this makes many taggers unusable as there are many taggers (i.e. morphological analysers) that are not deterministic and produces multiple analysis per morpheme.

Another challenge that we faced throughout the thesis is the absence of benchmark dataset. Beside the known split of the Arabic Treebank (PATB), we have not found any other data split that we can compare our results with, especially classical texts. The Arabic Treebank split even requires adhering to its labelling schema. The Arabic Treebank is not freely available and requires a membership of the Linguistic Data Consortium. Because of this absence, we implemented several approaches and compared them against each other. We are publishing our data split and code and hope it will be considered as a benchmark for classical Arabic.

10.4 Future Work

Many different adaptations, configurations, tests, and experiments are left for future work, and it is mainly because of lack of time. Experiments with larger/different datasets, and/or different configurations are usually very time and computational power consuming. We look forward to continuing exploring two topics in particular in addition to extending the annotation of the Sunnah Arabic Corpus and its website development.

The Sunnah Arabic Corpus is developed in this thesis in response to the lack of other annotated corpora beside the Quran. The Sunnah, being the second source of Islamic law and morals, is an under-resourced valuable text. We aim to continue the annotation of the Riyadh Asslaheen and include other books. Now that we have an annotation tool that is efficient in time and accuracy, we aim to access some funds to annotate the remaining parts of the corpus. We hope as well to syntactically annotate the corpus using available resources. In addition, the current website for SAC does not make the most of the resource for the end-users. We aim to enrich it with translations of the Hadith and utilise and group many scattered resources in an intuitive user interface.

In addition, the current ensemble approaches do not make use of morphological analysers. As shown before, the best analysers do not only rely on statistical methods of information extraction of training corpora, but they have

access to external resources such as lexicon and morphological databases. Since the number of classical Arabic resources are growing (e.g. Hadith Science Lexicon (Najeeb *et al.*, 2015) and Heritage corpus (Mohamed, 2018)) and our paradigm is to reuse and exploit available resources, we plan to incorporate them in our neural network architecture. Instead of only incorporating a single morphological analyser, we plan to continue our deferred work of combining multiple heterogeneous morphological analysers and use the ensemble instead. This work is halfway completed, and we have now experience of merging and aligning heterogeneous labels.

One important improvement is experimenting to exploit existing heterogenous annotated corpora, instead of exploiting heterogenous POS taggers, especially since we have a number of recently introduced classical Arabic annotated corpora. The two problems look similar but there are some critical differences. The adaptation of ensemble POS taggers is dependent on the quality of the POS taggers on the samples of the training dataset; however, annotated corpora are verified and assumed to have correct annotations. Exploiting heterogeneous annotated corpora can be converted to our problem: an ensemble of POS taggers, simply by training a tagger on each corpus. However, to fully exploit the differences, the training of these taggers can be done simultaneously and some information can be shared for the benefit of all taggers. A similar approach (Qiu, Zhao and Huang, 2013) has been done for Chinese which has some common features with Arabic. We have experienced a less similar approach when combining the two problems: segmentation and labelling in one network. Although they have different input and output, they share information (by encoding the sequence and concatenating the two encodings) that is useful for both tasks.

The ensemble methods are designed to be language neutral. We would like to experiment how our ensemble may be applied to other domains/languages. For example, we can make use of the AMALGAM project (Atwell *et al.*, 2000) which aggregated several existing rival taggers, and build an ensemble on top of these taggers. The Chinese language has rival segmentation schemas and we might compare our ensemble to related work on exploiting corpora-based heterogeneous annotation style.

The end-to-end approach using deep learning is actually a hot topic in the literature. It has been proven to be one of the most successful approaches in several

classification problems. In this thesis, the two ends of the approach start from the output of the input taggers, and conclude with the morphological analysis of one word. As a future work, we plan to extend this process from both ends. Using existing taggers that are tuned to some other domains is not optimal, so one possible adaptation is to tune these taggers within the ensemble process, thus making the end-to-end process start from the corpora itself. Another extension can be done to the other end, i.e. syntactic parsing of the text. One study in the literature shows that MSA and classical Arabic share similar syntax which can be exploited (Zhang *et al.*, 2015).

The current network architecture can be improved in several ways. The stacking of LSTM hidden layers has been successfully applied in POS tagging (thus earning the description as deep learning) (Goldberg, 2017). For example, we could map a word from its embedding representation. But, with a deeper network, a word can be represented more efficiently from its characters. Similarly, we would like to experiment with a deeper representation of input taggers by exploring their outputs in the different levels: character, morpheme, and word. Another way is to stack tasks where the output of one task (e.g. POS tag) can happen in the different layers, not only on the outermost layer. This approach has been reported in (Søgaard and Goldberg, 2016) that it is worthwhile to make higher-level tasks make use of lower-level representation, especially when a hierarchy between tasks exists. This might also apply to our task with POS tagging and morphological feature prediction.

Diacritics in the input text is not fully exploited in our ensemble. In almost all input taggers, these taggers are designed to ignore these marks (because they contribute to increasing word sparsity). We plan to retune these input taggers so that analyses are ranked based on the similarity in diacritics. In addition, the experiment of diacritization of classical Arabic can be enhanced by exploiting larger diacritics corpora, flexible fuzzy matching of words, and better representation of contexts.

LIST OF REFERENCES

Abandah, G. A., Graves, A., Al-Shagoor, B., Arabiyat, A., Jamour, F. and Al-Taee, M. (2015) 'Automatic diacritization of Arabic text using recurrent neural networks', *International Journal on Document Analysis and Recognition*. Springer Berlin Heidelberg, 18(2), pp. 183–197. doi: 10.1007/s10032-015-0242-2.

Abdel Ghafour, H. H., El-Bastawissy, A. and Heggazy, A. (2011) 'AEDA: Arabic edit distance algorithm - Towards a new approach for Arabic name matching', in *Proceedings - ICCES'2011: 2011 International Conference on Computer Engineering and Systems*, pp. 307–311. doi: 10.1109/ICCES.2011.6141061.

Abdul-Mageed, M., Diab, M. T. and Kübler, S. (2013) 'ASMA: A System for Automatic Segmentation and Morpho-Syntactic Disambiguation of Modern Standard Arabic.', in *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pp. 1–8.

Adda, G., Mariani, J., Lecomte, J., Paroubek, P. and Rajman., M. (1998) 'The GRACE French part-of-speech tagging evaluation task.', in *International Conference on Language Resources and Evaluation, Granada, May.*, pp. 433–441.

Akbari, A. A. (1986) *Grammatical Analysis of The Prohetic Hadith (Arabic)*. Edited by A. Nabhan. Damascus: The Center of Arabic Langage.

Al-jundi, A. (2016) *Towards an evalutation of POS taggers in view of Tamam's tagset*. Available at: http://www.hamassa.com/2016/09/19/للتحميل-أقسام-الكلم-العربي-نحو-تقييم/.

Al-saqi, F. (1975) *Arabic Part-of-Speech: Structure and Functionality (Arabic)*. Dar Al Uloom.

Al-Sughaiyer, I. A. and Al-Kharashi, I. A. (2004) 'Arabic morphological analysis techniques: A comprehensive survey', *Journal of the American Society for Information Science and Technology*. Wiley Subscription Services, Inc., A Wiley Company, 55(3), pp. 189–213. doi: 10.1002/asi.10368.

Al-Sulaiti, L. and Atwell, E. S. (2006) 'The design of a corpus of contemporary Arabic', *International Journal of Corpus Linguistics*. John Benjamins Publishing Company, 11(2), pp. 135–171.

Alabbas, M. A. S. (2013) *Textual Entailment for Modern Standard Arabic, PhD thesis*. The University of Manchester, UK.

Alabbas, M. and Ramsay, A. (2012a) 'Combining black-box taggers and parsers for modern standard Arabic', in *Federated Conference on Computer Science and*

Information Systems (FedCSIS). Wroclaw, Poland, pp. 19–26.

Alabbas, M. and Ramsay, A. (2012b) 'Improved POS-Tagging for Arabic by Combining Diverse Taggers', in *8th International Conference on Artificial Intelligence Applications and Innovations (AIAI)*. Halkidiki, Greece: Springer (Artificial Intelligence Applications and Innovations), pp. 107–116. doi: 10.1007/978-3-642-33409-2_12.

Alabbas, M. and Ramsay, A. (2014) 'Combining strategies for tagging and parsing Arabic', in *Proceedings of the EMNLP 2014 Workshop on Arabic NLP (ANLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 73–100.

Alashqar, A. M. (2012) 'A comparative study on Arabic POS tagging using Quran corpus', in *Informatics and Systems (INFOS)*. IEEE, pp. 29–33.

Albared, M. and Hazaa, M. A. S. (2015) 'N-attributes stochastic classifier combination for Arabic morphological disambiguation', *Saba Journal Of information Technology And Networking (SJITN)-ISSN: 2312-4989*, 3(1).

Albared, M., Omar, N. and Ab Aziz, M. J. (2009) 'Arabic part of speech disambiguation: A survey', *International Review on Computers and Software*, 4(5), pp. 517–532.

Albogamy, F. and Ramsay, A. (2016) 'Fast and Robust POS tagger for Arabic Tweets Using Agreement-based Bootstrapping', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Portorož, Slovenia, pp. 1500–1506.

Alfahal, M. Y. (2007) *Riyad-us-Saliheen with commentary on Ahadith (Arabic)*. Dar Ibn Katheer, Damascus, Syria.

Algahtani, S. and McNaught, J. (2015) 'Joint Arabic Segmentation and Part-Of-Speech Tagging', in *Second Workshop on Arabic Natural Language Processing*, p. 108.

Aliwy, A. H. (2013) *Arabic Morphosyntactic Raw Text Part of Speech Tagging System, PhD thesis*. University of Warsaw.

Aliwy, A. H. (2015) 'Combining Pos Taggers in Master-Slaves Technique for Highly Inflected Languages As Arabic', in *2015 International Conference on Cognitive Computing and Information Processing(CCIP)*, pp. 1–5. doi: 10.1109/CCIP.2015.7100682.

Alkhazi, I. S. and Teahan, W. J. (2017) 'Classifying and Segmenting Classical and Modern Standard Arabic using Minimum Cross-Entropy', *IJACSA) International*

Journal of Advanced Computer Science and Applications, 8(4), pp. 421–430. doi: 10.14569/IJACSA.2017.080456.

Almeman, K. A. (2015) *Reducing Out-of-Vocabulary in Morphology to Improve the Accuracy in Arabic Dialects Speech Recognition, PhD Thesis*. University of Birmingham.

AlOmari, O. (2005) *The Grammatical Analysis Of The Nawawi Forty Book (Arabic)*. Onizah.

Alosaimy, A. and Atwell, E. (2016) 'Ensemble Morphosyntactic Analyser for Classical Arabic', in *Second International Conference on Arabic Computational Linguistics*. Konya, Turkey.

Alosaimy, A. and Atwell, E. (2017) 'Sunnah Arabic Corpus: Design and Methodology', in *Proceedings of the 5th International Conference on Islamic Applications in Computer Science and Technologies*. Semarang, Indonesia.

Alosaimy, A. and Atwell, E. (2018) 'Diacritization of a Highly Cited Text: A Classical Arabic Book as a Case', in *2nd IEEE International Workshop on Arabic and derived Script Analysis and Recognition (ASAR 2018)*. London, UK.

Alrabiah, M. (2014) *Building A Distributional Semantic Model for Traditional Arabic & Investigating its Novel Applications to The Holy Quran, PhD thesis*. King Saud University.

Alrabiah, M., Al-Salman, A., Atwell, E. S. and Alhelewh, N. (2014) 'KSUCCA: a key to exploring Arabic historical linguistics', *International Journal of Computational Linguistics (IJCL)*. CSC Journals, 5(2), pp. 27–36.

Altabbaa, M., Al-zaraee, A. and Shukairy, M. (2010) *An Arabic Morphological Analyzer and Part-Of-Speech Tagger, M.S. thesis*. Arab International University, Damascus, Syria.

An-Nawawi, Y. I.-Š. (1976) 'An-Nawawi's Forty Hadith: Matn Al-arbaʿin An-nawawiya'. Holy Koran Publishing House.

Attia, M. (2006) 'An Ambiguity-Controlled Morphological Analyzer for Modern Standard Arabic Modelling Finite State Networks', in *The Challenge of Arabic for NLP/MT Conference*. London: The British Computer Society.

Attia, M., Pecina, P. and Toral, A. (2011) 'An open-source finite state morphological transducer for modern standard Arabic', in *Proceedings of the 9th International Workshop on Finite State Methods and NLP*, pp. 125–133.

Attia, M., Rashwan, M. and Al-Badrashiny, M. (2009) 'Fassieh®, a Semi-Automatic

Visual Interactive Tool for Morphological, PoS-Tags, Phonetic, and Semantic Annotation of Arabic Text Corpora', *IEEE Transactions on Audio, Speech and Language Processing*, 17(5), pp. 916–925. doi: 10.1109/TASL.2009.2019298.

Atwell, E. (2008) 'Development of tag sets for part-of-speech tagging', *Corpus Linguistics: An International Handbook, Volume 1. Berlin, Germany: Walter de Gruyter*, pp. 501–526.

Atwell, E., Al-Sulaiti, L., Al-osaimi, S. and Shawar, B. A. (2004) 'A Review of Arabic Corpus Analysis Tools', in *Proceedings of JEP-TALN Arabic language processing*. Fez, Morocco, pp. 229–234.

Atwell, E., Demetriou, G., Hughes, J., Schiffrin, A., Souter, C. and Wilcock, S. (2000) 'A Comparative Evaluation of Modern English Corpus Grammatical Annotation Schemes', *ICAME Journal: International Computer Archive of Modern and Medieval English Journal*, (24), pp. 7–23.

Atwell, E., Hughes, J. and Souter, C. (1994) 'AMALGAM: Automatic Mapping Among Lexico-Grammatical Annotation Models', in *Proceedings of ACL workshop on The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pp. 11–20.

Azmi, A. M. and Almajed, R. S. (2015) 'A survey of automatic Arabic diacritization techniques', *Natural Language Engineering*, 21(03), pp. 477–495. doi: 10.1017/S1351324913000284.

Banko, M. and Brill, E. (2001) 'Scaling to Very Very Large Corpora for Natural Language Disambiguation', in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pp. 26–33. doi: 10.3115/1073012.1073017.

Beesley, K. R. (1998) 'Arabic Morphology Using Only Finite-State Operations', in *Proceedings of the Workshop on Computational Approaches to Semitic languages*, pp. 50–57. doi: 10.3115/1621753.1621763.

Belinkov, Y., Magidow, A., Romanov, M., Shmidman, A. and Koppel, M. (2016) 'Shamela: A Large-Scale Historical Arabic Corpus', in *LT4DH: Language Technology Resources and Tools for Digital Humanities workshop*.

Benajiba, Y. and Zitouni, I. (2010) 'Arabic Word Segmentation for Better Unit of Analysis', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Valletta, Malta, pp. 1346–1352.

Bin-Muqbil, M. S. (2006) *Phonetic And Phonological Aspects Of Arabic Emphatics*

And Gutturals, PhD thesis. University Of Wisconsin-madison.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2017) 'Enriching Word Vectors with Subword Information', *Transactions of the Association for Computational Linguistics*, 5(1), pp. 135–146. doi: 1511.09249v1.

Bollmann, M., Petran, F., Dipper, S. and Krasselt, J. (2014) 'CorA: A web-based annotation tool for historical and other non-standard language data', in *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*. Gothenburg, Sweden, pp. 86–90.

Bouamor, H., Habash, N., Salameh, M., Zaghouani, W., Rambow, O., Abdulrahim, D., Obeid, O., Khalifa, S., Eryani, F., Erdmann, A. and Oflazer, K. (2018) 'The MADAR Arabic Dialect Corpus and Lexicon', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Miyazaki, Japan, pp. 3387–3396.

Boudchiche, M., Mazroui, A., Bebah, M., Lakhouaja, A. and Boudlal, A. (2016) 'AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer', *Journal of King Saud University-Computer and Information Sciences*. Elsevier.

Boudlal, A., Lakhouaja, A., Mazroui, A., Meziane, A., Bebah, M. and Shoul, M. (2010) 'Alkhalil morpho sys1: A morphosyntactic analysis system for arabic texts', in *International Arab Conference on Information Technology*.

Brants, T. (1995) 'Tagset Reduction Without Information Loss', in *33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, MA, USA, pp. 287–289.

Breiman, L. (2001) 'Random Forests', *Machine Learning*, 45(1), pp. 5–32.

Brierley, C., Sawalha, M. and Atwell, E. (2012) 'Open-Source Boundary-Annotated Corpus for Arabic Speech and Language Processing', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Istanbul, Turkey, pp. 1011–1016.

Buckwalter, T. (2002a) 'Arabic Morphological Analyzer (AraMorph) version 1.2'.

Buckwalter, T. (2002b) 'Buckwalter Arabic Morphological Analyzer Version 1.0'.

Carletta, J. (1996) 'Assessing agreement on classification tasks: the kappa statistic', *Computational Linguistics*, 22(2), pp. 249–254. doi: 10.1.1.48.4108.

Chen, D. and Manning, C. (2014) 'A Fast and Accurate Dependency Parser using Neural Networks', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pp. 740–750. doi:

10.3115/v1/D14-1082.

Chen, H., Zhang, Y. and Liu, Q. (2016) 'Neural Network for Heterogeneous Annotations', *Emnlp 2016*, pp. 731–741.

Chen, X., Shi, Z., Qiu, X. and Huang, X. (2017) 'Adversarial Multi-Criteria Learning for Chinese Word Segmentation', in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1193–1203. doi: 10.18653/v1/P17-1110.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014) 'Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pp. 1724–1734. doi: 10.3115/v1/D14-1179.

Clark, A. (2007) 'Supervised and Unsupervised Learning of Arabic Morphology', in Soudi, A., Bosch, A. van den, and Neumann, G. (eds) *Arabic Computational Morphology*. Dordrecht: Springer Netherlands, pp. 181–200. doi: 10.1007/978-1-4020-6046-5_10.

Clive Holes, R. A. (2004) *Modern Arabic: Structures, Functions, and Varieties*. Revised Ed. Georgetown University Press (Georgetown Classics in Arabic Languages and Linguistics series).

Cohen, J. (1960) 'A coefficient of agreement for nominal scales', *Educational and psychological measurement*. Sage Publications Sage CA: Thousand Oaks, CA, 20(1), pp. 37–46.

Cohen, W., Ravikumar, P. and Fienberg, S. (2003) 'A Comparison of String Distance Metrics for Name-Matching Tasks', *Proceedings of IJCAI-03 Workshop on Information Integration on the Web*, pp. 73–78. doi: 10.1002/spe.4380120106.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L. and Jégou, H. (2018) 'Word Translation Without Parallel Data', in *International Conference on Learning Representations*, pp. 1–14.

Damerau, F. (1964) 'A technique for computer detection and correction of spelling errors', *Communications of the ACM*, pp. 171–176.

Darwish, K. and Abdelali, A. (2017) 'Arabic POS Tagging : Don't Abandon Feature Engineering Just Yet', in *Third Workshop on Arabic Natural Language Processing (WANLP 2017)*. Valencia, Spain, pp. 130–137.

Darwish, K., Abdelali, A. and Mubarak, H. (2014) ‘Using Stem-Templates to improve Arabic POS and Gender/Number Tagging’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Reykjavik, Iceland, pp. 2926–2931.

Darwish, K. and Mubarak, H. (2016) ‘Farasa: A New Fast and Accurate Arabic Word Segmenter’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Portorož, Slovenia, pp. 1070–1074.

Darwish, K., Mubarak, H. and Abdelali, A. (2017) ‘Arabic Diacritization: Stats, Rules, and Hacks’, in *Proceedings of The Third Arabic Natural Language Processing Workshop*, pp. 9–17.

Debili, F. and Achour, H. (1998) ‘Voyellation automatique de l’arabe’, in *Proceedings of the Workshop on Computational Approaches to Semitic Languages*. Montreal, Canada, pp. 42–49.

Diab, M. (2009) ‘Second generation AMIRA tools for Arabic processing: Fast and robust tokenization, POS tagging, and base phrase chunking’, in *Conference on Arabic Language Resources and Tools*. Cairo, Egypt, pp. 285–288.

Diab, M., Habash, N., Rambow, O. and Roth, R. (2013) *LDC Arabic Treebanks and Associated Corpora: Data Divisions Manual*.

Diab, M., Hacioglu, K. and Jurafsky, D. (2004) ‘Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks’, in *HLT-NAACL 2004: Short Papers*. Boston, USA, pp. 149–152.

Diab, M. T. (2007) ‘Improved Arabic base phrase chunking with a new enriched POS tag set’, in *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages Common Issues and Resources - Semitic ’07*. Prague, Czech Republic, p. 89. doi: 10.3115/1654576.1654592.

Dienes, P. and Oravecz, C. (2000) ‘Bottom–up tagset design from maximally reduced tagset’, in *Proceedings of the COLING-2000 Workshop on Linguistically Interpreted Corpora*. Centre Universitaire, Luxembourg, pp. 42–47.

Dimitrova, L., Ide, N., Petkevič, V., Erjavec, T., Kaalep, H. J. and Tufis, D. (1998) ‘Multext-East: Parallel and Comparable Corpora and Lexicons for Six Central and Eastern European Languages’, in *The 17th International Conference on Computational Linguistics*. Montreal, Canada, pp. 315–319. doi: 10.3115/980845.980897.

Dmitriev, K. (2016) *Open Arabic Project*, *GitHub repository*. GitHub. Available at:

https://github.com/OpenArabic/Annotation.

Dukes, K., Atwell, E. and Habash, N. (2013) 'Supervised collaboration for syntactic annotation of Quranic Arabic', *Language Resources and Evaluation*, pp. 33–62. doi: 10.1007/s10579-011-9167-7.

Dukes, K. and Habash, N. (2010) 'Morphological Annotation of Quranic Arabic', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Valletta, Malta.

Dyer, C., Chahuneau, V. and Smith, N. A. (2013) 'A Simple, Fast, and Effective Reparameterization of IBM Model 2', in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia, pp. 644–648.

El-haj, M. and Koulali, R. (2013) 'KALIMAT a Multipurpose Arabic Corpus', in *Second Workshop on Arabic Corpus Linguistics (WACL-2)*. Lancaster, UK, pp. 22–25.

El-Imam, Y. A. (2004) 'Phonetization of Arabic: rules and algorithms', *Computer Speech & Language*, 18(4), pp. 339–373. doi: https://doi.org/10.1016/S0885-2308(03)00035-4.

Elhadj, Y. (2009) 'Statistical Part-of-Speech Tagger for Traditional Arabic Texts', *Journal of Computer Science*, 5(11), pp. 794–800. doi: 10.3844/jcssp.2009.794.800.

Elhadj, Y., Abdelali, A. and Ammar, A. (2014) 'Revisiting Arabic Part of Speech Tagsets', in *IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, pp. 793–802.

Elhadj, Y., Al-Sughaiyer, I. A., Khorsi, A. and Alansari, A. (2010) 'The Morphological Analysis of the Holy Qur'an: A Database of the Entire Quranic Text (Arabic)', *International Journal of Computer Science and Engineering in Arabic*, 3(1).

Elming, J. and Habash, N. (2007) 'Combination of Statistical Word Alignments Based on Multiple Preprocessing Schemes', 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, pp. 25–28. doi: 10.3115/1614108.1614115.

Freeman, A. (2001) 'Brill's POS tagger and a Morphology parser for Arabic', *ACL 2001 Workshop on Data-Driven Machine Translation,* p. 7.

Freeman, A., Condon, S. and Ackerman, C. (2006) 'Cross Linguistic Name

Matching in English and Arabic', in *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, US, pp. 471–478. doi: 10.3115/1220835.1220895.

Gahbiche-Braham, S., Bonneau-Maynard, H., Lavergne, T. and Yvon, F. (2012) 'Joint Segmentation and POS Tagging for Arabic Using a CRF-based Classifier', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Istanbul, Turkey, pp. 2107–2113.

Gerdes, K. (2013) 'Collaborative Dependency Annotation.', in *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*. Prague, Czech Republic, pp. 88–97.

Giesbrecht, E. and Evert, S. (2009) 'Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus', in *Web as Corpus Workshop WAC5*. San Sebastian, Spain, pp. 27–35.

Goldberg, Y. (2017) *Neural Network Methods for Natural Language Processing*, *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Gomaa, W. and Fahmy, A. (2013) 'A survey of text similarity approaches', *International Journal of Computer Applications*, 68(13), pp. 13–18.

Graves, A. (2012) *Supervised Sequence Labelling with Recurrent Neural Networks*, *Image Rochester NY*. Springer-Verlag Berlin Heidelberg. doi: 10.1007/978-3-642-24797-2.

Habash, N. (2007) 'Arabic Morphological Representations for Machine Translation', in *Arabic Computational Morphology*. Springer, Dordrecht (Text, Speech and Language Technology), pp. 263–285. doi: 10.1007/978-1-4020-6046-5_14.

Habash, N. (2010) *Introduction to Arabic Natural Language Processing*, *Synthesis Lectures on Human Language Technologies*. doi: 10.2200/S00277ED1V01Y201008HLT010.

Habash, N., Diab, M. and Rambow, O. (2012) 'Conventional Orthography for Dialectal Arabic', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Istanbul, Turkey, pp. 711–718.

Habash, N., Rambow, O. and Roth, R. (2009) 'MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization', in *Proceedings of the Second International*

Conference on Arabic Language Resources and Tools. Cairo, Egypt, pp. 102–109.

Habash, N. and Roth, R. M. (2009) 'CATiB: the Columbia Arabic Treebank', in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Suntec, Singapore, pp. 221–224.

Habash, N. and Sadat, F. (2006) 'Arabic Preprocessing Schemes for StatisticalMachine Translation', in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. New York City, US.

Habash, N., Shahrour, A. and Al-Khalil, M. (2016) 'Exploiting Arabic Diacritization for High Quality Automatic Annotation', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Portoroz, Slovenia, pp. 4298–4304.

Hajic, J., Smrz, O., Zemánek, P., Šnaidauf, J. and others (2004) 'Prague Arabic dependency treebank: Development in data and tools', *Intern. Conf. on Arabic …*. ufal.mff.cuni.cz.

Halteren, H. Van, Zavrel, J. and Daelemans, W. (2001) 'Improving Accuracy in Word Class Tagging through the Combination of Machine Learning Systems', *Computational Linguistics*, 27(2), pp. 199–229. doi: 10.1162/089120101750300508.

Hassan, T. (1994) *Arabic language its sense and structure (Arabic)*. Casablanca: Dar Ath-Thaqafa.

Heintz, I. (2014) 'Language Modeling', in Zitouni, I. (ed.) *Natural Language Processing of Semitic Languages*. Springer, pp. 161–196. doi: 10.1007/978-3-642-45358-8.

Henrich, V., Reuter, T. and Loftsson, H. (2009) 'CombiTagger: A System for Developing Combined Taggers', in *Proceedings of the Twenty-Second International Florida Artificial Intelligence Research Society Conference*. Sanibel Island, USA: AAAI Press, pp. 254–259.

Hermena, E., Drieghe, D., Hellmuth, S. and Liversedge, S. P. (2015) 'Processing of Arabic Diacritical Marks: Phonological-Syntactic Disambiguation of Homographic Verbs and Visual Crowding Effects', *Journal of Experimental Psychology: Human Perception and Performance*, pp. 494–507.

Hifny, Y. (2012) 'Higher Order n-gram Language Models for Arabic Diacritics Restoration', in *Proceedings of the Twelfth Conference on Language Engineering (ESOLEC'12)*. Cairo, Egypt, pp. 1–5.

Hochreiter, S. and Urgen Schmidhuber, J. (1997) 'Long Short-Term Memory',

Neural Computation, 9(8), pp. 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

Hughes, J., Souter, C. and Atwell, E. (1995) 'Automatic Extraction of Tagset Mappings from Parallel-Annotated Corpora', in *Proceedings of the ACL-SIGDAT Workshop From Text to Tags: Issues in Multilingual Language Analysis*. Dublin, Ireland, pp. 10–17.

Hulden, M. (2009) 'Foma: a Finite-State Compiler and Library', in *Proceedings of the Demonstrations Session at EACL 2009*. Athens, Greece, pp. 29–32.

Hulden, M. and Samih, Y. (2012) 'Conversion of Procedural Morphologies to Finite-State Morphologies: a Case Study of Arabic', in *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*. San Sebastián, Spain: Association for Computational Linguistics, pp. 70–74.

Indurkhya, N. and Damerau, F. (2010) *Handbook of Natural Language Processing*. 2nd Editio, *Antimicrobial agents and chemotherapy*. 2nd Editio. Chapman and Hall/CRC.

Inoue, G., Shindo, H. and Matsumoto, Y. (2017) 'Joint Prediction of Morphosyntactic Categories for Fine-Grained Arabic Part-of-Speech Tagging Exploiting Tag Dictionary Information', in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 421–431.

Jiang, W., Huang, L. and Liu, Q. (2009) 'Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging', in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore, pp. 522–530. doi: 10.3115/1687878.1687952.

Jurafsky, D. and Martin, J. H. (2008) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Second Edi. doi: 10.1162/089120100750105975.

Khoja, S. (2001) 'APT: Arabic part-of-speech tagger', in *Proceedings of the Student Workshop at NAACL*. Pittsburgh, PA, USA, pp. 20–25.

Kim, Y.-B., Snyder, B. and Sarikaya, R. (2015) 'Part-of-speech Taggers for Low-resource Languages using CCA Features', in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1292–1302.

Kingma, D. P. and Ba, J. (2014) 'Adam: A Method for Stochastic Optimization', in

Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10. New York, New York, USA: ACM Press, p. 103. doi: 10.1145/1830483.1830503.

Kiraz, G. A. (2001) *Computational Nonlinear Morphology: with Emphasis on Semitic Languages*. Cambridge University Press.

Kobyliński, L. (2014) 'PoliTa: a Multitagger for Polish', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Reykjavik, Iceland, pp. 2949–2954.

Kübler, S. and Mohamed, E. (2012) 'Part of speech tagging for Arabic', *Natural Language Engineering*. Cambridge Univ Press, 18(04), pp. 521–548. doi: 10.1017/S1351324911000325.

Kübler, S. and Zinsmeister, H. (2015) *Corpus linguistics and linguistically annotated corpora*. Bloomsbury Publishing.

Kudo, T. and Matsumoto, Y. (2001) 'Chunking with support vector machines', in *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*. Pittsburgh, Pennsylvania. doi: http://dx.doi.org/10.3115/1073336.1073361.

Kulick, S., Gabbard, R. and Marcus, M. (2006) 'Parsing the Arabic treebank: Analysis and improvements', in *Proceedings of the 5th International Treebanks and Linguistic Theories*. Prague, Czech Republic, pp. 31–42.

Kuncheva, L. I. (2014) *Combining Pattern Classifiers*. Hoboken, NJ, USA: John Wiley & Sons, Inc. doi: 10.1002/9781118914564.

Lee, Y.-S. (2004) 'Morphological analysis for statistical machine translation', in *Proceedings of HLT-NAACL 2004: Short Papers*. Boston, Massachusetts, pp. 57–60. doi: 10.3115/1613984.1613999.

Lee, Y., Papineni, K. and Roukos, S. (2003) 'Language model based Arabic word segmentation', in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Sapporo, Japan, pp. 399–406. doi: 10.3115/1075096.1075147.

Leech, G. and Wilson, A. (1996) 'Eagles Recommendations for the Morphosyntactic Annotation of Corpora, EAG-TCWG-MAC/R', in *EAGLES Guidelines*. Pisa: Consiglio Nazionale delle Ricerche. Istituto di Linguistica Computazionale.

Levenshtein, V. (1966) 'Binary Codes Capable of Correcting Deletions, Insertions and Reversals', *Soviet Physics Doklady*, 10.

Maamouri, M. and Bies, A. (2004) ‘Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools’, in *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*. Geneva, Switzerland, pp. 2–9.

Maamouri, M., Bies, A., Buckwalter, T., Jin, H. and Mekki, W. (2005) *Arabic Treebank: Part 3 (full corpus) v 2.0 (MPG + Syntactic Analysis) LDC2005T20.* Available at: https://catalog.ldc.upenn.edu/LDC2005T20.

Maegaard, B. (2004) ‘NEMLAR-An Arabic Language Resources Project’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Lisbon, Portugal, pp. 109–112.

Malmasi, S. and Dras, M. (2018) ‘Native Language Identification With Classifier Stacking and Ensembles’, *Computational Linguistics*, 44(3), pp. 403–446. doi: 10.1162/coli_a_00323.

Mandhour, I. (1994) *Lisan Al-Arab (Arabs Tongue Dictinary,Arabic)*. Third Edit. Turath For Solutions.

Marquez, L., Rodriguez, H., Carmona, J. and Montolio, J. (1999) ‘Improving POS Tagging Using Machine-Learning Techniques’, in *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. College Park, USA, pp. 53–62.

Marton, Y., Habash, N. and Rambow, O. (2010) ‘Improving Arabic Dependency Parsing with Lexical and Inflectional Morphological Features’, in *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*. Los Angeles, CA, USA, pp. 13–21.

Marton, Y., Habash, N. and Rambow, O. (2013) ‘Dependency Parsing of Modern Standard Arabic with Lexical and Inflectional Features’, *Computational Linguistics*, 39(1), pp. 161–194. doi: 10.1162/COLI_a_00138.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013) ‘Distributed Representations of Words and Phrases and their Compositionality’, in *Advances in Neural Information Processing Systems 26 (NIPS 2013)*. Lake Tahoe, USA, pp. 1–9.

Mohamed, E. (2012) ‘Morphological Segmentation and Part of Speech Tagging for Religious Arabic’, in *Fourth Workshop on Computational Approaches to Arabic Script-based Languages (CAASL4)*. San Diego, USA, pp. 65–71.

Mohamed, E. (2018) ‘Morphological Segmentation and Part-of-Speech Tagging for the Arabic Heritage’, *ACM Transactions on Asian and Low-Resource Language*

Information Processing. New York, NY, USA: ACM, 17(3), pp. 1–13. doi: 10.1145/3178459.

Mohamed, E., Kübler, S., Sandra, K. and Hall, M. (2010) 'Is Arabic Part of Speech Tagging Feasible Without Word Segmentation?', in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California, pp. 705–708.

Monroe, W., Green, S. and Manning, C. D. (2014) 'Word Segmentation of Informal Arabic with Domain Adaptation', in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland, pp. 206–211.

More, A., Etino˘ Glu, O. Ç., A˘ Grı Oltekin, Ç., Habash, N., Sagot, B., Seddah, D., Taji, D. and Tsarfaty, R. (2018) 'CoNLL-UL: Universal Morphological Lattices for Universal Dependency Parsing', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Miyazaki, Japan.

Mueller, T., Schmid, H. and Schütze, H. (2013) 'Efficient Higher-Order CRFs for Morphological Tagging', in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 322–332.

Najeeb, M., Abdelkader, A., Al-Zghoul, M. and Osman, A. (2015) 'A Lexicon for Hadith Science Based on a Corpus', *International Journal of Computer Science and Information Technologies*, 6(2), pp. 1336–1340.

Needleman, S. B. and Wunsch, C. D. (1970) 'A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins', *Journal of Molecular Biology*, 48(3), pp. 443–453. doi: 10.1016/0022-2836(70)90057-4.

Nguyen, T. and Vogel, S. (2008) 'Context-based Arabic morphological analysis for machine translation', in *Proceedings of the Twelfth Conference on Computational Natural Language Learning - CoNLL '08*. Morristown, NJ, USA: Association for Computational Linguistics, pp. 135–142. doi: 10.3115/1596324.1596348.

Nivre, J., Agic, L. ˘Zeljko, Agić, Ž., Ahrenberg, L., Aranzabe, M. J., Asahara, M., Atutxa, A., Ballesteros, M., Bauer, J., Bengoetxea, K., Bhat, R. A., Bick, E., Bosco, C., Bouma, G., Bowman, S., Candito, M., Cebiroğlu Eryiğit, G., Celano, G. G. A., Chalub, F., Choi, J., Çöltekin, Ç., Connor, M., Davidson, E., de Marneffe, M.-C., de Paiva, V., Diaz de Ilarraza, A., Dobrovoljc, K., Dozat, T., Droganova, K., Dwivedi, P., Eli, M., Erjavec, T., Farkas, R., Foster, J., Freitas, C., Gajdošová, K., Galbraith, D., Garcia, M., Ginter, F., Goenaga, I., Gojenola, K., Gökırmak, M., Goldberg, Y.,

Gómez Guinovart, X., Gonzáles Saavedra, B., Grioni, M., Grūzītis, N., Guillaume, B., Habash, N., Hajič, J., Hà Mỹ, L., Haug, D., Hladká, B., Hohle, P., Ion, R., Irimia, E., Johannsen, A., Jørgensen, F., Kaşıkara, H., Kanayama, H., Kanerva, J., Kotsyba, N., Krek, S., Laippala, V., Lê Hồng, P., Lenci, A., Ljubešić, N., Lyashevskaya, O., Lynn, T., Makazhanov, A., Manning, C., Mărănduc, C., Mareček, D., Martínez Alonso, H., Martins, A., Mašek, J., Matsumoto, Y., McDonald, R., Missilä, A., Mititelu, V., Miyao, Y., Montemagni, S., More, A., Mori, S., Moskalevskyi, B., Muischnek, K., Mustafina, N., Müürisep, K., Nguyễn Thị, L., Nguyễn Thị Minh, H., Nikolaev, V., Nurmi, H., Ojala, S., Osenova, P., Øvrelid, L., Pascual, E., Passarotti, M., Perez, C.-A., Perrier, G., Petrov, S., Piitulainen, J., Plank, B., Popel, M., Pretkalniņa, L., Prokopidis, P., Puolakainen, T., Pyysalo, S., Rademaker, A., Ramasamy, L., Real, L., Rituma, L., Rosa, R., Saleh, S., Sanguinetti, M., Saulīte, B., Schuster, S., Seddah, D., Seeker, W., Seraji, M., Shakurova, L., Shen, M., Sichinava, D., Silveira, N., Simi, M., Simionescu, R., Simkó, K., Šimková, M., Simov, K., Smith, A., Suhr, A., Sulubacak, U., Szántó, Z., Taji, D., Tanaka, T., Tsarfaty, R., Tyers, F., Uematsu, S., Uria, L., van Noord, G., Varga, V., Vincze, V., Washington, J. N., Žabokrtský, Z., Zeldes, A., Zeman, D. and Zhu, H. (2017) *Universal Dependencies 2.0*. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics ({{\\'U}FAL}), Faculty of Mathematics and Physics, Charles University. Available at: http://hdl.handle.net/11234/1-2184 (Accessed: 22 September 2017).

Obeid, O., Khalifa, S., Habash, N., Bouamor, H., Zaghouani, W., Oflazer, K. and Ag, J. A. (2018) 'MADARi : A Web Interface for Joint Arabic Morphological Annotation and Spelling Correction', in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Miyazaki, Japan: European Language Resources Association (ELRA).

Odebrecht, C. (2018) *MKM – ein Metamodell für Korpusmetadaten (German), PhD Thesis*. Humboldt-Universität zu Berlin, Sprach- und literaturwissenschaftliche Fakultät. doi: http://dx.doi.org/10.18452/19407.

Paroubek, P. (2007) 'Evaluating Part-of-Speech Tagging and Parsing', in Dybkjær, L., Hemsen, H., and Minker, W. (eds) *Evaluation of Text and Speech Systems*. Dordrecht: Springer Netherlands (Text, Speech and Language Technology), pp. 99–124. doi: 10.1007/978-1-4020-5817-2_4.

Pasha, A., Al-Badrashiny, M., Diab, M., Habash, N., Pooleery, M., Rambow, O. and

Roth, R. (2015) ‘MADAMIRA v2.0 User Manual’. Center for Computational Learning Systems, Columbia University.

Pasha, A., Al-Badrashiny, M., Diab, M., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O. and Roth, R. M. (2014) ‘Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Reykjavik, Iceland.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011) ‘Scikit-learn: Machine Learning in Python’, *Journal of Machine Learning Research*, 12, pp. 2825–2830.

Petrov, S., Das, D. and McDonald, R. (2012) ‘A Universal Part-of-Speech Tagset’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Istanbul, Turkey, pp. 2089–2096.

Pîrvan, F. and Tufi, D. (2006) ‘Tagset mapping and statistical training data cleaning-up’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Genoa, Italy, pp. 385–390.

Plank, B., Søgaard, A. and Goldberg, Y. (2016) ‘Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss’, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 412–418. doi: 10.18653/v1/P16-2067.

Qiu, X., Zhao, J. and Huang, X. (2013) ‘Joint Chinese Word Segmentation and POS Tagging on Heterogeneous Annotated Corpora with Multiple Task Learning.’, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, (October), pp. 658–668.

Rashwan, M. A. A., Al Sallab, A. A., Raafat, H. M. and Rafea, A. (2015) ‘Deep Learning Framework with Confused Sub-Set Resolution Architecture for Automatic Arabic Diacritization’, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3), pp. 505–516. doi: 10.1109/TASLP.2015.2395255.

Rashwan, M., Al-Badrashiny, M., Attia, M. and Abdou, S. M. (2009) ‘A Hybrid System for Automatic Arabic Diacritization’, *2nd International Conference on Arabic Language Resources and Tools*, (June 2014), pp. 54–60. doi:

10.1109/TASL.2010.2045240.

Ryding, K. C. (2005) *A reference grammar of modern standard Arabic*, *Language*. Cambridge University Press. doi: 10.1353/lan.2008.0050.

S. Rabiee, H. (2011) 'Adapting Standard Open-Source Resources To Tagging A Morphologically Rich Language: A Case Study With Arabic', in *Proceedings of the Second Student Research Workshop associated with RANLP 2011*. Hissar, Bulgaria, pp. 127–132.

Salih, B. (2007) *A Detailed Grammatical Analysis of the Recited Quran using i'rāb (Arabic)*. Beirut: Dar Al-Fikr.

Al Sallab, A., Rashwan, M., M. Raafat, H. and Rafea, A. (2014) 'Automatic Arabic diacritics restoration based on deep nets', in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 65–72. doi: 10.3115/v1/W14-3608.

Samih, Y., Maier, W. and Kallmeyer, L. (2016) 'SAWT: Sequence Annotation Web Tool', in *Proceedings of the Second Workshop on Computational Approaches to Code Switching*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 65–70. doi: 10.18653/v1/W16-5808.

Sawalha, M. (2011) *Open-source resources and standards for Arabic word structure analysis: Fine grained morphological analysis of Arabic text corpora, PhD thesis*. University of Leeds.

Sawalha, M. and Atwell, E. (2013) 'A standard tag set expounding traditional morphological features for Arabic language part-of-speech tagging', *Word Structure*, 6(1), pp. 43–99. doi: 10.3366/word.2013.0035.

Sawalha, M., Atwell, E. and Abushariah, M. a M. (2013) 'SALMA: Standard Arabic Language Morphological Analysis', in *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. IEEE, pp. 1–6. doi: 10.1109/ICCSPA.2013.6487311.

Schmidt, T., Witt, A., Hinrichs, E., Rehm, G., Chiarcos, C. and Lehmberg, T. (2006) 'Avoiding Data Graveyards: From Heterogeneous Data Collected in Multiple Research Projects to Sustainable Linguistic Resources', in *E-MELD Workshop on Digital Language Documentation: Tools and Standards: The State of the Art*. Lansing, MI, USA.

Schneider, N., Mohit, B., Oflazer, K. and Smith, N. (2012) 'Coarse lexical semantic annotation with supersenses: an Arabic case study', in *Proceedings of the 50th*

Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Jeju Island, Korea, pp. 253–258.

Schroder, I. (2002) *A case study in part-of-speech- tagging using the ICOPOST toolkit*. Department of Computer Science, University of Hamburg.

Schuster, M. and Paliwal, K. K. (1997) 'Bidirectional recurrent neural networks', *IEEE Transactions on Signal Processing*, 45(11), pp. 2673–2681. doi: 10.1109/78.650093.

Sibawayh, A. B. A. (1988) *Kitab Sibawayh (Arabic)*. Third Edit. Edited by A. Haron. Ar Riyad - Saudi Arabia: Alkhanji.

Sinclair, J. and Ball, J. (1996) 'Eagles: Preliminary Recommendations on Text Typology, EAG-TCWG-TTYP/P', in *EAGLES Guidelines*. Pisa: Consiglio Nazionale delle Ricerche. Istituto di Linguistica Computazionale.

Sjöbergh, J. (2003) 'Combining POS-taggers for improved accuracy on Swedish text', in *Proceedings of 14th Nordic Conference of Computational Linguistics, NoDaLiDa 2003*. Reykjavik, Iceland.

Smrz, O. (2007) *Functional Arabic Morphology. Formal System and Implementation, PhD thesis*, *The Prague Bulletin of Mathematical Linguistics*. Charles University in Prague.

Śniatowski, T. and Piasecki, M. (2012) 'Combining Polish Morphosyntactic Taggers', in Bouvry, P., K\lopotek, M. A., Leprévost, F., Marciniak, M., Mykowiecka, A., and Rybi\'nski, H. (eds) *Security and Intelligent Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 359–369. doi: 10.1007/978-3-642-25261-7_28.

Søgaard, A. (2009) 'Ensemble-based POS tagging of Italian', in *11th Conference of the Italian Association for Artificial Intelligence*. Reggio Emilia, Italy.

Søgaard, A. and Goldberg, Y. (2016) 'Deep multi-task learning with low level tasks supervised at lower layers', in Intergovernmental Panel on Climate Change (ed.) *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 231–235. doi: 10.18653/v1/P16-2038.

Soudi, A., Bosch, A. van den, Ide, N., Jean, V. and Kiraz, G. (2007) 'Arabic Computational Morphology : Knowledge-Based and Empirical Methods', *Arabic Computational Morphology*, 38(11), pp. 3–14. doi: 10.1007/978-1-4020-6046-5.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S. and Tsujii, J. (2012)

‘BRAT: a web-based tool for NLP-assisted text annotation’, in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Avignon, France: Association for Computational Linguistics, pp. 102–107.

Straka, M. and Straková, J. (2017) ‘Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe’, in *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada: Association for Computational Linguistics, pp. 88–99.

Sutskever, I., Vinyals, O. and Le, Q. V. (2014) ‘Sequence to Sequence Learning with Neural Networks’, *Mathematical Programming*, 155(1–2), pp. 105–145. doi: 10.1007/s10107-014-0839-0.

Teahan, W. J., Wen, Y., McNab, R. and Witten, I. H. (2000) ‘A Compression-based Algorithm for Chinese Word Segmentation’, *Computational Linguistics*, 26(3), pp. 375–393. doi: 10.1162/089120100561746.

Teufel, S. (1995) ‘A Support Tool for Tagset Mapping’, in *Proceedings of the Workshop SIGDAT: From Text to Tags (EACL95)*.

Toutanova, K., Klein, D., Manning, C. D. and Singer, Y. (2003) ‘Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network’, in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL ’03*. Morristown, NJ, USA: Association for Computational Linguistics, pp. 173–180. doi: 10.3115/1073445.1073478.

Tsarfaty, R., Seddah, D., Goldberg, Y., Kuebler, S., Versley, Y., Candito, M., Foster, J., Rehbein, I. and Tounsi, L. (2010) ‘Statistical Parsing of Morphologically Rich Languages (SPMRL): What, How and Whither’, in *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*. Los Angeles, CA, USA: Association for Computational Linguistics, pp. 1–12.

Ungurean, C., Burileanu, D., Popescu, V., Negrescu, C. and Dervis, A. (2008) ‘Automatic diacritic restoration for a TTS-based e-mail reader application’, in *UPB Scientific Bulletin, Series C*. Bucharest, Romania, pp. 3–12.

Vergyri, D. and Kirchhoff, K. (2004) ‘Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition’, in *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*. Stroudsburg, PA,

USA (Semitic ’04), pp. 66–73.
Wintner, S. (2014) ‘Morphological processing of Semitic languages’, in *Natural language processing of Semitic languages*. Springer, pp. 43–66.
Yaseen, M., Attia, M., Maegaard, B., Choukri, K., Paulsson, N., Haamid, S., Krauwer, S., Bendahman, C., Fersøe, H., Rashwan, M., Haddad, B., Mukbel, C., Mouradi, A., Shahin, M., Chenfour, N. and Ragheb, A. (2006) ‘Building Annotated Written and Spoken Arabic LR’s in NEMLAR Project’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Genoa, Italy, pp. 533–538.
Yimam, S. M., Biemann, C., Eckart de Castilho, R. and Gurevych, I. (2014) ‘Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno’, in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 91–96. doi: 10.3115/v1/P14-5016.
Yosef, H. A. (2003) *The Iarab of Al-Nawawi’s Forty Hadith (Arabic)*. Cairo: AlMukhtar.
Zaghouani, W., Bouamor, H., Hawwari, A., Diab, M., Obeid, O., Ghoneim, M., Alqahtani, S. and Oflazer, K. (2015) ‘Guidelines and Framework for a Large Scale Arabic Diacritized Corpus’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Portorož, Slovenia, pp. 3637–3643.
Zalmout, N. and Habash, N. (2017) ‘Don’t Throw Those Morphological Analyzers Away Just Yet: Neural Morphological Disambiguation for Arabic’, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 704–713. doi: 10.18653/v1/D17-1073.
Zavrel, J. and Daelemans, W. (2000) ‘Bootstrapping a Tagged Corpus through Combination of Existing Heterogeneous Taggers’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Athens, Greece, pp. 17–20.
Zeman, D. (2008) ‘Reusable Tagset Conversion Using Tagset Drivers’, in *LREC: Proceedings of the International Conference on Language Resources and Evaluation*. Marrakech, Morocco, pp. 213–218.
Zeman, D., Popel, M., Straka, M., Hajic, J., Nivre, J., Ginter, F., Luotolahti, J., Pyysalo, S., Petrov, S., Potthast, M., Tyers, F., Badmaeva, E., Gokirmak, M.,

Nedoluzhko, A., Cinkova, S., Hajic jr., J., Hlavacova, J., Kettnerová, V., Uresova, Z., Kanerva, J., Ojala, S., Missilä, A., Manning, C. D., Schuster, S., Reddy, S., Taji, D., Habash, N., Leung, H., de Marneffe, M.-C., Sanguinetti, M., Simi, M., Kanayama, H., DePaiva, V., Droganova, K., Martínez Alonso, H., Çöltekin, Ç., Sulubacak, U., Uszkoreit, H., Macketanz, V., Burchardt, A., Harris, K., Marheinecke, K., Rehm, G., Kayadelen, T., Attia, M., Elkahky, A., Yu, Z., Pitler, E., Lertpradit, S., Mandl, M., Kirchner, J., Alcalde, H. F., Strnadová, J., Banerjee, E., Manurung, R., Stella, A., Shimada, A., Kwak, S., Mendonca, G., Lando, T., Nitisaroj, R. and Li, J. (2017) 'CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies', in *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 1–19. doi: 10.18653/v1/K17-3001.

Zeroual, I. and Lakhouaja, A. (2016) 'A new Quranic Corpus rich in morphosyntactical information', *International Journal of Speech Technology*. Springer, pp. 1–8.

Zeroual, I. and Lakhouaja, A. (2017) 'Feature-rich PoS Tagging through Taggers Combination : Experience in Arabic', *Transactions on Machine Learning and Artificial Intelligence*, 5(4). doi: 10.14738/tmlai.54.2981.

Zeroual, I., Lakhouaja, A. and Belahbib, R. (2017) 'Towards a standard Part of Speech tagset for the Arabic language', *Journal of King Saud University - Computer and Information Sciences*, 29(2), pp. 171–178. doi: 10.1016/j.jksuci.2017.01.006.

Zerrouki, T. and Balla, A. (2017) 'Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems', *Data in Brief*. Elsevier, 11, pp. 147–151. doi: 10.1016/j.dib.2017.01.011.

Zhang, Y., Li, C., Barzilay, R. and Darwish, K. (2015) 'Randomized Greedy Inference for Joint Segmentation, POS Tagging and Dependency Parsing', in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 42–52. doi: 10.3115/v1/N15-1005.

Zhang, Y., Reichart, R. and Barzilay, R. (2012) 'Learning to Map into a Universal POS Tagset', in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju

Island, Korea, pp. 1368–1378.

Zribi, C., Torjmen, A. and Ahmed, M. (2007) ‘A Multi-Agent System for POS-Tagging Vocalized Arabic Texts.’, *The International Arab Journal of Information Technology*, 4(4), pp. 322–329.

Appendices

APPENDIX A: ANNOTATED HADITH EXAMPLE BY SEVERAL TAGGERS

In this appendix, we show a full sentence of one Hadith (prophet sayings) annotated in parallel by several morphological analysers and POS taggers. Columns represent 1) the abbreviation of the tool, 2) word id with morpheme id (if detected), 3) word form, 4) lemma, 5) assigned POS-tag, and 6) analysed morphological features such as gender (if available).

A.1 The Hadith Sentence (by MAs)

TOOL CODE	Word No.	Word Form	Lemma / Stem	POS tag	Morpho Features
AL	1	lA	lA_1	part_neg	-
AR	1	lA	-	part_neg	-
BP	1	lA	-	NEG_PART	-
EX	1	laA	laA	F-	-
KH	1	laA	laA	Hrf nfy	-
MS	1	laA	laA	HARF_NAFY	-
QT	1	lAa	-	pc	-
AL	2	yu&omin	|man_1	verb	Gender=M| Number=S| Aspect=IMPF| Voice=ACT| Person=3
AR	2	>Amn	-	verb	Gender=M| Number=S| Aspect=IMPF| Voice=ACT| Person=3
BP	2-0	yu	-	IV3MS	Gender=M| Number=S| Aspect=IMPF| Voice=ACT| Person=3
BP	2-1	&omin	|man_1	VERB_IMPERFECT	-
EX	2	yu&minu	|man	VI	Gender=M| Number=S| Mood=IND| Aspect=IMPF| Voice=ACT| Person=3
KH	2	yu&am~inu	>am~ana	fEl mDArE mbny llmElwm	Case=NOM| Aspect=IMPV| Person=3
MS	2-0	-	-	PREFIX_ YA2_ ANAIT_ MA3LOOM_ MAGHOOL	-
MS	2-1	yu&omin	yu&omin	FE3L_MODARE3_MAZEED	Aspect=IMPF
QT	2	UNK-WORD			
AL	3-0	>aHadkum	>aHad_1	noun	Gender=M| Number=S| Case=-
AL	3-1	-	-	2mp_poss	Gender=M| Number=P| Person=2

AR	3-0	>Hd	-	noun	Gender=M| Number=S
AR	3-1	_km	-	genpron	Gender=M| Number=P| Person=2
BP	3-0	>aHad	>aHad_1	NOUN	-
BP	3-1	kum	-	POSS_PRON_2MP	Gender=M| Number=P| Person=2
EX	3-0	>aHadu	>aHad	N-	Number=S| Case=NOM
EX	3-1	kum	huwa	SP	Gender=M| Number=P| Case=ACC| Person=2
KH	3-0	>aHadakumo	>aHad	Asm jAmd	Gender=M| Number=S| Case=ACC
KH	3-1	-	-	kumo: Dmyr AlmxATbyn	-
MS	3-0	>aHad~akumo	>aHad~a	AF3AL_TA3AGOB	-
MS	3-1	-	-	SUFFIX_ KUM_ MOKHATAB_ GAM3_ MOTHAKAR	Number=P| Person=2
QT	3	UNK-WORD			
AL	4	Hat~aY	Hat~aY_ 1	prep	-
AR	4	HtY	-	prep	-
BP	4	Hat~aY	-	PREP	-
EX	4	Hat~aY	Hat~aY	P-	-
KH	4	Hat~aY	Hat~aY	Hrf ETf	-
MS	4	Hat~aY	Hat~aY	HARF_GARR	-
QT	4	HatY~a	-	pp	-
AL	5	yakuwn	kAn_1	verb	Gender=M| Number=S| Aspect=IMPF| Voice=ACT| Person=3
AR	5	-	kaw~an	verb	Gender=M| Number=S| Aspect=IMPF| Voice=PASS| Person=3
BP	5-0	ya	-	IV3MS	Gender=M| Number=S| Aspect=IMPF| Voice=ACT| Person=3
BP	5-1	kuwn	kAn_1	VERB_IMPERFECT	-
EX	5	yakuwna	kaAn	VI	Gender=M| Number=S| Mood=SUBJ| Aspect=IMPF|

					Voice=ACT| Person=3
KH	5	yukowun~a	>akowaY	fEl mDArE m&kd mbny llmElwm	Aspect=IMPV| Person=3
MS	5-0	-	-	PREFIX_YA2_ANAIT_MA3LOOM	Voice=ACT
MS	5-1	yakuwn	yakuwn	FE3L_MODARE3_MOGARRAD	Aspect=IMPF
QT	5	UNK-WORD			
AL	6-0	hawAh	hawaY_1	noun	Gender=M| Number=S| Case=-
AL	6-1	-	-	3ms_poss	Gender=M| Number=S| Person=3
AR	6-0	hwY	-	noun	Gender=M| Number=S
AR	6-1	_h	-	genpron	Gender=M| Number=S| Person=3
BP	6-0	hawA	hawaY_1	NOUN	-
BP	6-1	hu	-	POSS_PRON_3MS	Gender=M| Number=S| Person=3
EX	6-0	hawaY	hawaY	N-	Number=S| Case=NOM
EX	6-1	hu	huwa	SP	Gender=M| Number=S| Case=ACC| Person=3
KH	6-0	hawaAhu	hawFY	Asm jAmd	Gender=M| Number=S| Case=NOM
KH	6-1	-	-	hu: Dmyr AlgA}b	-
MS	6-0	hawaAhu	hawaY	MASDAR_MOGARRAD	-
MS	6-1	-	-	SUFFIX_ HA2_ MODAF_ GHA2EB_ MOTHAKKAR	Gender=M| Person=3
QT	6	UNK-WORD			
AL	7	tabaEAF	tabaEAF_1	adv	Gender=M| Number=S| Case=ACC
AR	7	tbEAF	-	adv	-
BP	7-0	tabaE	tabaEAF_1	ADV	-
BP	7-1	AF	-	NSUFF_MASC_SG_ACC_INDEF	-
EX	7	tabaEFA	tabaE	N-	Number=S| Case=GEN
KH	7	tiboEFA	tiboE	Asm jAmd	Gender=M| Number=S| Case=ACC
MS	7-0	tabaEFA	tabaEFA	MASDAR_MOGARRAD	-

MS	7-1	-	-	SUFFIX_ALEF_TANWEEN	-
QT	7	UNK-WORD			
AL	8-0	li	-	prep	-
AL	8-1	mA	mA_1	pron_rel	Gender=M| Number=S| Case=-
AR	8-0	l_	-	prep	-
AR	8-1	mA	-	rel	Number=S
BP	8-0	li	-	PREP	-
BP	8-1	mA	limA_1	REL_PRON	-
EX	8-0	li	li	P-	-
EX	8-1	maA	maA	S-	-
KH	8-0	-	-	li : Hrf Aljr	-
KH	8-1	limaA	maA	Asm mwSwl	-
MS	8-0	-	-	PREFIX_LAM_GARR	-
MS	8-1	limaA	maA	ESM_MAWSOOL	-
QT	8	limaA	-	nc	Case=GEN
AL	9	ji}ota	jA'_1	verb	Gender=M| Number=S| Mood=IND| Aspect=PERF| Voice=ACT| Person=2
AR	9	jA'	-	verb	Aspect=PERF| Voice=ACT| Person=1
BP	9-0	ji}	jA'_1	VERB_PERFECT	-
BP	9-1	tu	-	PVSUFF_SUBJ:1S	Number=S| Aspect=PERF| Voice=ACT| Person=1
EX	9	ji}tu	jaA'	VP	Gender=M| Number=S| Aspect=PERF| Voice=ACT| Person=1
KH	9	ji}otu	jaA'a	fEl mAD mbny llmElwm	Person=1
MS	9-0	ji}otu	jaA'a	FE3L_MADI_MOGARRAD	Aspect=PERF
MS	9-1	-	-	SUFFIX_TA2_FA3EL_MOTA KALLEM	Person=1
QT	9	UNK-WORD			
AL	10-0	bihi	bi_1	prep	-
AL	10-	-	-	3ms_pron	Gender=M| Number=S|

	1				Person=3
AR	10-0	b_	-	prep	-
AR	10-1	_h	-	objcon	Gender=M| Number=S| Person=3
BP	10-0	bi	bi-_1	PREP	-
BP	10-1	hi	-	PRON_3MS	Gender=M| Number=S| Person=3
EX	10-0	bi	bi	P-	-
EX	10-1	hi	huwa	SP	Gender=M| Number=S| Case=ACC| Person=3
KH	10	bihi	bihi	jAr wmjrwr	-
MS	10-0	bihi	bi	HARF_GARR	-
MS	10-1	-	-	SUFFIX_ HA2_ MODAF_ GHA2EB_ MOTHAKKAR	Gender=M| Person=3
QT	10	UNK-WORD			

A.2 The Hadith Sentence (by POS taggers)

TOOL	Word No.	Word Form	Lemma / Stem	POS tag	Morpho Features
AM	1	lA	-	RP	-
FA	1	lA	-	PART	-
MA	1	lA	lA_1	part_neg	-
MD	1	lA	lA_1	part_neg	-
MR	1	lA	-	RP	-
ST	1	lA	-	RP	-
WP	1	lA	-	part_neg	-
MT	1	lA	-	Laa	
AM	2	y&mn	-	VBP	Aspect=IMPF|Voice=ACT|Person=2
FA	2	y&mn	-	V	-
MA	2	yu&omin	|man_1	verb	Gender=M|Number=S|Aspect=IMPF|Voice=ACT|Person=3
MD	2	yu&omin	|man_1	verb	Gender=M|Number=S|Aspect=IMPF|Voice=ACT|Person=3
MR	2	ymn	-	VBP	-
ST	2	y&mn	-	VBP	Aspect=IMPF|Voice=ACT
WP	2	yu'minu	-	verb	-
MT	2	yu&minu	-	V	Gender=M|Number=S|Aspect=IMPF|Voice=ACT|Mood=Ind
AM	3-0	>Hd	-	NN	-
AM	3-1	km	-	PRP	Person=2
FA	3-0	>Hd	-	NOUN	Person=1
FA	3-1	km	-	PRON	-
MA	3-0	>aHadakum	>aHad_1	noun	Gender=M|Number=S|Case=ACC
MA	3-1	-	-	2mp_poss	Gender=M|Number=P|Person=2
MD	3-0	>aHadkum	>aHad_1	noun	Gender=M|Number=S|Case=-
MD	3-1	-	-	2mp_poss	Gender=M|Number=P|Person=2

MR	3-0	AHd	-	NN	-
MR	3-1	+km	-	PRP$	-
ST	3-0	AHd	-	NN	Number=S
ST	3-1	km	-	PRP$	-
WP	3	AHadukum	-	noun	-
MT	3-1	>aHadukumo	-	Ed	
MT	3-2	-	-	N.Qnt	Gender=M|Number=S|Case=Nom|Qnt=True
MT	3-3	-	-	Poss	
AM	4	HtY	-	CJP	-
FA	4	HtY	-	PREP	-
MA	4	Hat~aY	Hat~aY_1	prep	-
MD	4	Hat~aY	Hat~aY_1	prep	-
MR	4	Hty	-	AN	-
ST	4	HtY	-	IN	-
WP	4	Hat~ay	-	noun	-
MT	4	Hata~Y	-	Prp	
AM	5	ykwn	-	VBP	Aspect=IMPF|Voice=ACT|Person=2
FA	5	ykwn	-	V	-
MA	5	yakuwn	kAn_1	verb	Gender=M|Number=S|Aspect=IMPF|Voice=ACT|Person=3
MD	5	yakuwn	kAn_1	verb	Gender=M|Number=S|Aspect=IMPF|Voice=ACT|Person=3
MR	5	ykwn	-	VBP	-
ST	5	ykwn	-	VBP	Aspect=IMPF|Voice=ACT
WP	5	yakwna	-	verb	-
MT	5	yakuwna	-	V	Gender=M|Number=S|Aspect=IMPF|Voice=ACT|Mood=Sub
AM	6-0	hwY	-	NN	-
AM	6-1	h	-	PRP	Person=2
FA	6-0	hwA	-	NOUN	Person=1
FA	6-1	h	-	PRON	-
MA	6-0	hawAh	hawaY_1	noun	Gender=M|Number=S|Case=-
MA	6-1	-	-	poss	Gender=M|Number=S|Person=3

MD	6-0	hawAh	hawaY_1	noun	Gender=M|Number=S|Case=-
MD	6-1	-	-	poss	Gender=M|Number=S|Person=3
MR	6-0	hwy	-	NN	-
MR	6-1	+h	-	PRP$	-
ST	6-0	hwA	-	NN	Number=S
ST	6-1	h	-	PRP$	-
WP	6	hawAhu	-	noun	-
MT	6-1	hawaAhu	-	Ed	
MT	6-2	-	-	N	Gender=M|Number=S|Case=Nom
MT	6-3	-	-	Poss	Gender=M|Number=S|Person=3
AM	7	tbEA	-	NN	-
FA	7-0	tbE	-	NOUN	Person=1
FA	7-1	A	-	CASE	-
MA	7	tabaEAF	tabaE_1	noun	Gender=M|Number=S|Case=ACC
MD	7	tabaEAF	tabaE_1	noun	Gender=M|Number=S|Case=ACC
MR	7	tbEA	-	NN	-
ST	7	tbEA	-	NN	Number=S
WP	7	tabaEAF	-	verb	-
MT	7	tabaEFA	-	N	Gender=M|Number=S|Case=ACC|Nunation=True
AM	8-0	l	-	IN	-
AM	8-1	mA	-	WP	-
FA	8-0	l+	-	PREP	-
FA	8-1	mA	-	PART	-
MA	8-0	li	-	prep	-
MA	8-1	mA	mA_1	pron_rel	Gender=M|Number=S|Case=-
MD	8-0	li	-	prep	-
MD	8-1	mA	mA_1	pron_rel	Gender=M|Number=S|Case=-
MR	8-0	l#	-	IN	-
MR	8-1	mA	-	WP	-
ST	8-0	l	-	IN	-
ST	8-1	mA	-	WP	-
WP	8	limA	-	noun_p	-

				rop	
MT	8-1	limaA	-	Le	
MT	8-2	-	-	RelMaa	Number=S
AM	9	j}t	-	VBD	Aspect=PERF|Voice=ACT|Person=2
FA	9-0	j}	-	V	-
FA	9-1	t	-	PRON	-
MA	9	ji}otu	jA'_1	verb	Gender=M|Number=S|Mood=IND|Aspect=PERF|Voice=ACT|Person=1
MD	9	ji}otu	jA'_1	verb	Gender=M|Number=S|Mood=IND|Aspect=PERF|Voice=ACT|Person=1
MR	9	jt	-	VBD	-
ST	9	j}t	-	VBD	Aspect=PERF|Voice=ACT
WP	9	ji'tu	-	noun_prop	-
MT	9-1	ji}out	-	V	Number=S|Aspect=PERF|Voice=ACT
MT	9-2	-	-	Subj	Number=S|Person=1
AM	10-0	b	-	IN	-
AM	10-1*	h	-	PRP	Person=2
FA	10-0	b+	-	PREP	-
FA	10-1	h	-	PRON	-
MA	10-0	bihi	bi_1	prep	-
MA	10-1	-	-	3ms_pron	Gender=M|Number=S|Person=3
MD	10-0	bihi	bi_1	prep	-
MD	10-1	-	-	pron	Gender=M|Number=S|Person=3
MR	10-0	b#	-	IN	-
MR	10-1	+h	-	PRP	-
ST	10-0	b	-	IN	-

ST	10-1	h	-	PRP	-
WP	10	bihi	-	noun_prop	-
MT	10-1	bihi	-	Prp	
MT	10-2	-	-	Poss	Gender=M|Number=S|Person=3

APPENDIX B: OUTPUT FORMAT DIFFERENCES

Figure 10.1 A sample of the output of AraMorph in two versions Java and Perl. In the Perl version, each solution has the vocalized word (in parenthesis), lemma (in square brackets), analyses of each segments where segments are separated by plus sign, and finally a helpful gloss in Engish.

```
SOLUTION #1
Lemma :       jA'
Vocalized as :  ji}tu
Morphology :
       prefix : Pref-0
       stem : PV_C
       suffix : PVSuff-t
Grammatical category :
       stem : ji}     VERB_PERFECT
       suffix : tu    PVSUFF_SUBJ:1S
Glossed as :
       stem : arrive/come/occur
       suffix : I <verb>

... 2 more solutions
```

(a) Java

```
INPUT STRING: j}t
LOOK-UP WORD: j}t
  SOLUTION 1: (ji}otu) [jA'_1
] ji}/VERB_PERFECT+tu/PVSUFF_SUBJ:1S
     (GLOSS): + arrive/come/occur + I <verb>
  SOLUTION 2: (ji}ota) [jA'_1
] ji}/VERB_PERFECT+ta/PVSUFF_SUBJ:2MS
     (GLOSS): + arrive/come/occur + you [masc.sg.] <verb>
  SOLUTION 3: (ji}oti) [jA'_1
] ji}/VERB_PERFECT+ti/PVSUFF_SUBJ:2FS
     (GLOSS): + arrive/come/occur + you [fem.sg.] <verb>
```

(b) Perl

Figure 10.2 Alkhalil output of one analysis of the word "ji}otu" is on the first row. We added a new row for translating the output shown in the first row. It is clear that the POS tags and the type of the word are not in a good reusable format.

Input جِئْتُ *ǧiʾtu*	Voweled Word جِئْتُ *ǧiʾtu*	Prefix #	Stem جـ*ǧʾ*ـت	Type فعل مَاض مبني لِلمعلوم *fʿl māḍ mbny llmʿlwm*	Pattern فِلْتُ *filtu*	Root جيء *ǧyʾ*	POS Tags ثلاثي مجرد مسند إلى المتكلم أنا ولازم متعد *tlā-ty mǧrd msnd ʾlā ālmtklm ʾnā mtʿd wlāzm*	Suffix تاء المتكلم *tāʾ ā-lmtklm*
ji}otu	ji}otu	#	j}t	Active perfect verb	1i2o3u	jy´	VIII Unaugmented first-Person Transitive and Intransitive	t: t of first-person

Figure 10.3 A sample of the output of AraComLex.

```
j}t     +verb+past+activejA'+1pers@
j}t     +verb+past+activejA'+2pers+sg+masc@
j}t     +verb+past+activejA'+2pers+sg+fem@
```

Figure 10.4 A sample of the output of Elixir FM. Each analysis has seven columns (e.g. first column is an eight-slot string that represent the POS tag and morphological features).

```
:::: ji}otu

 ::: <^gi'tu>

  :: (792,1)   ["arrive","come","occur"]
               Verb [] [FIL] []      [I]
               ^gA'    "^g y '"      FAL    jaA'   jA'
   : <^gi'tu>  ji}tu   j}t
     VP-A-1MS-- ^gi'tu "^g y '"      FiL |<< "tu"   ji}tu   j}t
   : <^gi'tu>  ji}tu   j}t
     VP-A-1FS-- ^gi'tu "^g y '"      FiL |<< "tu"   ji}tu   j}t
   : <^gi'tu>  ji}tu   j}t
     VP-P-1MS-- ^gi'tu "^g y '"      FiL |<< "tu"   ji}tu   j}t
   : <^gi'tu>  ji}tu   j}t
     VP-P-1FS-- ^gi'tu "^g y '"      FiL |<< "tu"   ji}tu   j}t
```

Figure 10.5 A sample of the XML output of the Qutuf System.

```
<Word number_of_possibilities="2" original_string="limaA">
       <SurfaceFormMorphemes certainty="0.8125" voweled_form="limaA">
              <Proclitcs>
                     <Proclitc arabic_description="Hrf, Hrf jr, ZAhr" tag="p,p"/>
              </Proclitcs>
              <Cliticless arabic_description="Asm, m*kr >w m&nv, mfrd >w mvnY >w jmE, ?, mjrwr, Asm mwSwl
                    m$trk, mErfp, ZAhr" tag="n,mf,sdp,?,g,c,d"/>
              <Enclitics/>
       </SurfaceFormMorphemes>
       <SurfaceFormMorphemes certainty="0.5" voweled_form="limaA">
              <Proclitcs>
                     <Proclitc arabic_description="Hrf, Hrf jr, ZAhr" tag="p,p"/>
              </Proclitcs>
              <Cliticless arabic_description="Asm, ?, ?, ?, mjrwr, Asm $rT, nkrp, ZAhr" tag="n,?,?,?,g,h,i
                    "/>
              <Enclitics/>
       </SurfaceFormMorphemes>
</Word>
```

Figure 10.6 A sample of the output of ALMORGEANA. The representation of the analysis is similar to MADA and MADAMIRA.

```
;;WORD j}t
diac:ji}ota lex:jA'_1 bw:+ji}/PV++ta/PVSUFF_SUBJ:2MS gloss:arrive/come/occur pos:verb prc3:0 prc2:0 prc1:0
      prc0:0 per:2 asp:p vox:a mod:i gen:m num:s stt:na cas:na enc0:0 rat:na source:lex stem:ji} stemcat:
      PV_C
diac:ji}oti lex:jA'_1 bw:+ji}/PV++ti/PVSUFF_SUBJ:2FS gloss:arrive/come/occur pos:verb prc3:0 prc2:0 prc1:0
      prc0:0 per:2 asp:p vox:a mod:i gen:f num:s stt:na cas:na enc0:0 rat:na source:lex stem:ji} stemcat:
      PV_C
diac:ji}otu lex:jA'_1 bw:+ji}/PV++tu/PVSUFF_SUBJ:1S gloss:arrive/come/occur pos:verb prc3:0 prc2:0 prc1:0
      prc0:0 per:1 asp:p vox:a mod:i gen:m num:s stt:na cas:na enc0:0 rat:na source:lex stem:ji} stemcat:
      PV_C
```

Figure 10.7 A sample of the output of MADA. It is identical to ALMORGEANA except its solutions are ranked. Starred solutions are the selected solution.

```
;;WORD j}t
;;SVM_PREDICTIONS: j}t asp:p cas:na enc0:0 gen:m mod:i num:s per:1 pos:verb prc0:0 prc1:0 prc2:0 prc3:0 stt:
      na vox:a
*1.000126 diac:ji}otu lex:jA'_1 bw:+ji}/PV++tu/PVSUFF_SUBJ:1S gloss:arrive/come/occur pos:verb prc3:0 prc2:0
       prc1:0 prc0:0 per:1 asp:p vox:a mod:i gen:m num:s stt:na cas:na enc0:0 rat:na source:lex stem:ji}
      stemcat:PV_C
_0.944387 diac:ji}ota lex:jA'_1 bw:+ji}/PV++ta/PVSUFF_SUBJ:2MS gloss:arrive/come/occur pos:verb prc3:0 prc2
      :0 prc1:0 prc0:0 per:2 asp:p vox:a mod:i gen:m num:s stt:na cas:na enc0:0 rat:na source:lex stem:ji}
      stemcat:PV_C
_0.910868 diac:ji}oti lex:jA'_1 bw:+ji}/PV++ti/PVSUFF_SUBJ:2FS gloss:arrive/come/occur pos:verb prc3:0 prc2
      :0 prc1:0 prc0:0 per:2 asp:p vox:a mod:i gen:f num:s stt:na cas:na enc0:0 rat:na source:lex stem:ji}
      stemcat:PV_C
```

Figure 10.8 A sample of the output of MADAMIRA: Like MADA output except for *sufgloss* (suffix gloss) feature.

```
;;WORD j}t
;;LENGTH 3
;;OFFSET 37
;;SVM_PREDICTIONS: j}t diac:ji}otu lex:jA' asp:p cas:na enc0:0 gen:m mod:i num:s per:1 pos:verb prc0:0 prc1
      :0 prc2:0 prc3:0 stt:na vox:a
*0.893935 diac:ji}otu lex:jA'_1 bw:ji}/PV+tu/PVSUFF_SUBJ:1S gloss:arrive/come/occur sufgloss:I_<verb> pos:
      verb prc3:0 prc2:0 prc1:0 prc0:0 per:1 asp:p vox:a mod:i gen:m num:s stt:na cas:na enc0:0 rat:na
      source:lex stem:ji} stemcat:PV_C
_0.856916 diac:ji}ota lex:jA'_1 bw:ji}/PV+ta/PVSUFF_SUBJ:2MS gloss:arrive/come/occur sufgloss:you_[masc.sg.]_
      <verb> pos:verb prc3:0 prc2:0 prc1:0 prc0:0 per:2 asp:p vox:a mod:i gen:m num:s stt:na cas:na enc0:0
      rat:na source:lex stem:ji} stemcat:PV_C
_0.830216 diac:ji}oti lex:jA'_1 bw:ji}/PV+ti/PVSUFF_SUBJ:2FS gloss:arrive/come/occur sufgloss:you_[fem.sg.]_<
      verb> pos:verb prc3:0 prc2:0 prc1:0 prc0:0 per:2 asp:p vox:a mod:i gen:f num:s stt:na cas:na enc0:0
      rat:na source:lex stem:ji} stemcat:PV_C
```

Figure 10.9 A sample of the output of MarMoT.

```
1       lA      _       _       _       PRT|RP _        NEG|PART
2       ymn     _       _       _       VRB|VBP _       IV3MS|IV|IVSUFF|MOOD|I
3       AHd     _       _       _       NOM|NN _        NOUN|CASE|DEF|ACC
4       km      _       _       _       NOM|PRP$        _       POSS|PRON|2MP
5       Hty     _       _       _       PRT|AN _        SUB|CONJ
6       ykwn    _       _       _       VRB|VBP _       IV3MS|IV|IVSUFF|MOOD|S
7       hwy     _       _       _       NOM|NN _        NOUN
8       h       _       _       _       NOM|PRP$        _       POSS|PRON|3MS
9       tbEA    _       _       _       NOM|NN _        NOUN|CASE|INDEF|ACC
10      l       _       _       _       PRT|IN _        PREP
11      mA      _       _       _       NOM|WP _        REL|PRON
12      jt      _       _       _       VRB|VBD _       PV|PVSUFF|SUBJ|3FS
13      b       _       _       _       PRT|IN _        PREP
14      h       _       _       _       NOM|PRP _       PRON|3MS
```

Figure 10.10 A sample of the output of SAPA.

```
# 0 0.554063
lA      part_neg+none+none+none part_neg+none+none+none/0.999943
y&mn    verb+none+none+none   verb+none+none+none/0.999972
>Hdkm   noun+none+none+none   noun+none+none+none/0.974859
HtY     prep+none+none+none   prep+none+none+none/0.682635
ykwn    verb+none+none+none   verb+none+none+none/0.950193
hwAh    noun+none+none+none   noun+none+none+none/0.969479
tbEA    noun+none+none+none   noun+none+none+none/0.979848
lmA     pron_rel+none+PREP+none pron_rel+none+PREP+none/0.922642
j}t     verb+none+none+none   verb+none+none+none/0.999986
bh      prep+none+PREP+none   prep+none+PREP+none/0.999839
```

Figure 10.11 A sample of the output of the Stanford POS Tagger, AMIRA, and Farasa. Standford does not mark segmented morphemes (e.g for regrouping later).

```
#ST
lA/RP y&mn/VBP_MS3 >Hd/NN +km/PRP_MP2 HtY/CJP ykwn/VBP_MS3 hwY/NN +h/PRP_MS3 tbEA/NN l#/IN mA/WP j}t/VBD_FS3
      b#/IN +h/PRP_MS3
#AM
lA/RP y&mn/VBP AHd/NN km/PRP$ HtY/IN ykwn/VBP hwA/NN h/PRP$ tbEA/NN l/IN mA/WP j}t/VBD b/IN h/PRP
#FA
S/S lA/PART y&mn/V >Hd/NOUN-MS +km/PRON HtY/PREP ykwn/V hwA/NOUN-MS +h/PRON tbE/NOUN-MS +A/CASE l+/PREP +mA/
     PART j}/V +t/PRON b+/PREP +h/PRON E/E
```

APPENDIX C: SOURCE TEXT OF WASIM CASE STUDIES

1 Modern Standard Arabic and Morphological Analyser

Arabic excerpt:

وبغضّ النّظر عن المآل، فإنّ اللغةَ والهويّةَ القوميّة كانتا متشابكتيْن بالنسبة إلى أورزيديل. فقد اشتبكَ، بعد أن استقرّ في أمريكا، مع ثقافة وطنه المُتبنَّى "أكثر بكثير من معظم الكتّاب المنفيين الألمان الآخرين"، كما أخبرني جوهان. وكان قد حصلَ لأورزيديل إدراكٌ وظيفيّ تماماً للإنجليزيّة. وقرأ بصرامةٍ الكتّاب الأمريكيين - مثل رالف والدو إيمرسون، وهنري ديفيد ثورو، وناثانيل هوثورن، ووالت ويتمان- ونشر مقالات عنهم بالألمانيّة. وقد ترجمَ الشاعرةَ الأمريكيّة (H.D) إلى لغته الأمّ. بيد أنّه لم ينشر في الإنجليزيّة على نحو خلّاق.

English translation.

Whatever the fate, language and national identity were intertwined with Orziedel. After settling in America, he clashed with the culture of his adopted homeland "much more than most other exiled German writers," Johan told me. Orzidl had a very functional grasp of English. He read the books of the American writers - such as Ralph Waldo Emerson, Henry David Thoreau, Nathaniel Hawthorne, and Walt Whitman - and published articles about them in German. The American poet (H.D.) was translated into his mother tongue. However, it was not published in English creatively.

Reference:

http://midan.aljazeera.net/intellect/literature/2017/5/27/%D9%84%D9%85%D8%A7%D8%B0%D8%A7-%D9%86%D8%B1%D8%B3%D9%85-%D8%A8%D9%84%D8%AF%D8%A7-%D8%AE%D9%8A%D8%A7%D9%84%D9%8A%D8%A7-%D8%AD%D9%8A%D9%86%D9%85%D8%A7-%D9%86%D9%81%D9%82%D8%AF-%D8%A7%D9%84%D9%88%D8%B7%D9%86

2 Quranic Arabic and Consistency Reinforcement

Arabic verses from chapter 18 (Alkahf, the cave):

10. إِذْ أَوَى الْفِتْيَةُ إِلَى الْكَهْفِ فَقَالُوا رَبَّنَا آتِنَا مِن لَّدُنكَ رَحْمَةً وَهَيِّئْ لَنَا مِنْ أَمْرِنَا رَشَدًا
11. فَضَرَبْنَا عَلَى آذَانِهِمْ فِي الْكَهْفِ سِنِينَ عَدَدًا
12. ثُمَّ بَعَثْنَاهُمْ لِنَعْلَمَ أَيُّ الْحِزْبَيْنِ أَحْصَى لِمَا لَبِثُوا أَمَدًا
13. نَّحْنُ نَقُصُّ عَلَيْكَ نَبَأَهُم بِالْحَقِّ إِنَّهُمْ فِتْيَةٌ آمَنُوا بِرَبِّهِمْ وَزِدْنَاهُمْ هُدًى

14. وَرَبَطْنَا عَلَى قُلُوبِهِمْ إِذْ قَامُوا فَقَالُوا رَبُّنَا رَبُّ السَّمَاوَاتِ وَالْأَرْضِ لَن نَّدْعُوَ مِن دُونِهِ إِلَهًا لَّقَدْ قُلْنَا إِذًا شَطَطًا
15. هَؤُلَاءِ قَوْمُنَا اتَّخَذُوا مِن دُونِهِ آلِهَةً لَّوْلَا يَأْتُونَ عَلَيْهِم بِسُلْطَانٍ بَيِّنٍ فَمَنْ أَظْلَمُ مِمَّنِ افْتَرَى عَلَى اللَّهِ كَذِبًا

English translation:

10. [Mention] when the youths retreated to the cave and said, "Our Lord, grant us from Yourself mercy and prepare for us from our affair right guidance."
11. So We cast [a cover of sleep] over their ears within the cave for a number of years.
12. Then We awakened them that We might show which of the two factions was most precise in calculating what [extent] they had remained in time.
13. It is We who relate to you, [O Muhammad], their story in truth. Indeed, they were youths who believed in their Lord, and We increased them in guidance.
14. And We made firm their hearts when they stood up and said, "Our Lord is the Lord of the heavens and the earth. Never will we invoke besides Him any deity. We would have certainly spoken, then, an excessive transgression.
15. These, our people, have taken besides Him deities. Why do they not bring for [worship of] them a clear authority? And who is more unjust than one who invents about Allah a lie?"

Reference:

http://tanzil.net/#18:10

3 Sunnah Arabic and Keyboard Navigation

Arabic hadith from The Book of Miscellany (Alkahf, the cave):

وعنه أن رسول الله صلى الله عليه وسلم قال:"بينما رجل يمشي في حلة تعجبه نفسه، مرجل رأسه، يختال في مشيته، إذ خسف الله به، فهو يتجلجل في الأرض إلى يوم القيامة" ((متفق عليه)).

English translation:

Messenger of Allah (peace be upon him) said, “While a man was walking, dressed in clothes admiring himself, his hair combed, walking haughtily when Allah caused the earth to swallow him. Now he will continue to go down in it (as a punishment) until the Day of Resurrection.”

[Muslim].

4 English and UDPipe

English excerpt:

“Brazil's government has abolished a vast national reserve in the Amazon to open up the area to mining.

The area, covering 46,000 sq km (17,800 sq miles), straddles the northern states of Amapa and Para, and is thought to be rich in gold, and other minerals.

The government said nine conservation and indigenous land areas within it would continue to be legally protected.

But activists have voiced concern that these areas could be badly compromised.”

Source: http://www.bbc.co.uk/news/world-latin-america-4103322