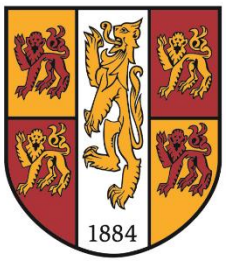
DOCTOR OF PHILOSOPHY

Compression-based Parts-of-Speech Tagger for the Arabic Language

Alkhazi, Ibrahim

Award date:
2019

Awarding institution:
Bangor University



PRIFYSGOL
BANGOR
UNIVERSITY

School of computer science

College of Physical & Applied Sciences

Compression-based Parts-of-Speech Tagger for the Arabic Language

---

Ibrahim Sulaiman B Alkhazi

Submitted in partial satisfaction of the requirements for the
Degree of Doctor of Philosophy
in Computer Science

Supervisor Dr. William J. Teahan

September 2019

Table of Content

List of Figures

List of Tables

Acknowledgement

All praise and gratitude to Allah The All-powerful for His blessings and guidance. Peace and blessings of Allah be upon the Prophet Muhammad.

I would like to express my sincere appreciation to my mother, Shama Saleh, for her constant motivation and prayers. I also would love to express my deep thankfulness for my wife, Bador, for supporting me to obtain great confidence and faith in myself. Many appreciations to my beloved children Alanoud and Alhanouf for being there for me throughout this journey and for drawing a smile on my face when I needed it the most. I would also like to show my appreciations to Ms Khawla S Alkhazi for the time and effort she spent contributing to this research.

I would love to express my heartfelt and genuine appreciation to my PhD supervisor Dr. William J. Teahan for his assistance, support and patience which led to the completion of this research. Thank you, Dr. William, for granting me the opportunity to complete this research.

Many thanks to my sponsor, the University of Tabuk, for the financial aid and for Tabuk Public Library, for providing the facilities required to complete the field trip.

I also would love to express my appreciation to my colleagues, Dr Mansor Alhgamdi and Mohammad Altamimi for their help and support.

Abstract

The Arabic language is a morphologically complex language that causes various difficulties for various NLP systems, such as POS tagging. The motive of this research is to investigate the development and training of a compression-based Arabic POS tagger using the PPM algorithm. The adoption of the algorithm for Arabic POS tagging may increase the efficiency and reduce the Arabic language ambiguity problem.

The best text compression algorithms can be applied to NLP tasks often with state-of-the-art results. This research examines the use of tag-based compression of larger Arabic resources to re-evaluate the performance of tag-based compression which may reveal POS linguistic aspects of the Arabic language. We also found that tag-based text compression for the Arabic text can be utilised as a means of evaluating the performance and quality of the Arabic POS taggers. The results of the experiments show that the tag-based compression of the text can effectively be used for assessing the performance of Arabic POS taggers when used to tag different types of the Arabic text, and also as a means of comparing the performance of two Arabic POS taggers on the same text.

With the rapid growth of Arabic text on the Web, studies that address the problems of classification and segmentation of the Arabic language are limited compared to other languages, most of which implement word-based and feature extraction algorithms. This research adopts a PPM character-based compression scheme to classify and segment Classical Arabic (CA) and Modern Standard Arabic (MSA) texts. An initial experiment using the PPM classification method on samples of text resulted in an accuracy of 95.5%, an average precision of 0.958, an average recall of 0.955 and an average F-measure of 0.954, using the concept of minimum cross-entropy. Segmenting the CA and MSA text using the PPM compression algorithm obtained an accuracy of 86%, an average precision of 0.869, an average recall of 0.86 and an average F-measure of 0.859.

This research describes the creation of the new Bangor Arabic Annotated Corpus (BAAC) which is a Modern Standard Arabic (MSA) corpus that comprises 50K words manually annotated by parts-of-speech. For evaluating the quality of the corpus, the Kappa coefficient

and a direct percent agreement for each tag were calculated for the new corpus and a Kappa value of 0.956 was obtained, with an average observed agreement of 94.25%. The corpus was used to evaluate the widely used Madamira Arabic POS tagger and to further investigate compression models for text compressed using POS tags. Also, a new annotation tool was developed and employed for the annotation process of the BAAC.

CHAPTER 1

Introduction

Contents:

1.1 Background & Motivation

The Arabic language "العربية" is among the most popular languages in use today, as shown in Figure 1.1. In the United Nations, it is among the five official languages and it is the primary language of 330 million people living in 22 countries in Asia, North Africa and the Middle East along with it being a secondary language of 1.4 billion people [185]. Arabic is a morphologically rich language having a mutual structure with Semitic languages such as Tigrinya, Hebrew and Amharic. It is a morphologically complex language that causes various difficulties for Natural Language Processing (NLP) [74], [161], [104], [38]. Diacritics are used in the Arabic language to disambiguate terms. However, Modern Standard Arabic text is very commonly written without diacritics and the contextual information is used by the reader of the text to disambiguate the meaning of the term. As a result of this Arabic language ambiguity problem, there has been an increase in the adoption of statistical approaches in the Arabic NLP field to solve the uncertainty of Arabic text [180].

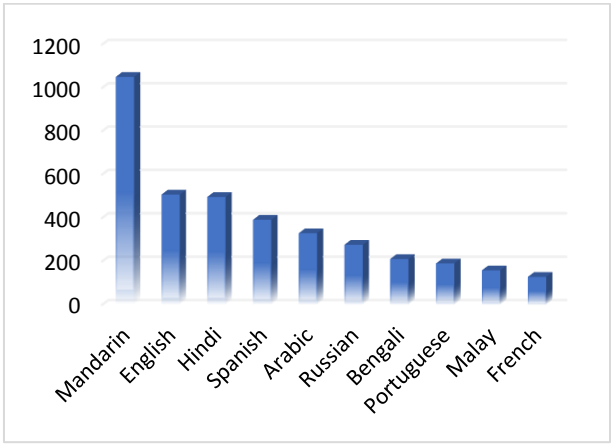


Figure 1.1. The most globally used languages [185], [135].

Natural language processing (NLP) is a computer science area of study which examines the process of understanding and manipulating human natural language speech or text to perform beneficial tasks, such as machine translation, part-of-speech tagging and speech recognition [70]. NLP started in the 1950s and involves research at the junction of linguistics and artificial intelligence [155].

A parts-of-speech (POS) tagger is a computer system that accepts text as input and then assigns a proper grammatical tag, such as VB for a verb, JJ for an adjective and NN for a noun,

as output for every token or term according to its appearance, position or order in the text. POS tagging is normally an initial step in any linguistic analysis and a very significant early step in the process of building several natural language processing (NLP) applications, such as information retrieval systems, spell auto-checking and correction systems and speech recognition systems [10].

The motive of this research is to investigate the development and training of a compression-based Arabic part-of-speech tagger. The new tagger utilises the Prediction by Partial Matching text compression scheme (PPM), which uses an adaptive statistical language model to make predictions about upcoming text and has been successfully applied to several Arabic NLP tasks, such as authorship attribution [46], [45], cryptology [15], text correction [19] and text compression [26], [29], but it has yet to have been applied to POS tagging. The adoption of the algorithm for Arabic POS tagging may increase the efficiency and reduce the Arabic language ambiguity problem.

PPM is an online adaptive text compression system that utilises the prior context to predict the coming symbol or character with given fixed context length. Previous experiments were performed by Alhawiti [26] to compare the three PPM models, character-based, word-based and tag-based, when used to compress the Arabic text and the size of resources used to estimate the tag-based text compression were small due to resource limitation. Since PPM is an online adaptive system that needs relatively large amounts of training data, this research investigates the use of the tag-based compression of larger Arabic resources as a method to evaluate the performance of different Arabic POS taggers.

Almost all Arabic language NLP tasks, such as part-of-speech tagging, are designed for Modern Standard Arabic text (MSA) [84]. Most of the popular Arabic POS taggers were trained on MSA text [141], [38], [37]. Contrastingly, tagging Classical Arabic text (CA) using MSA POS taggers will significantly reduce the quality of the tagging as reported in various studies [38], [37], [42], [40]. This research introduces the utilisation of compression-based techniques to classify and segment the two types of Arabic text to overcome the problem of code-switching in Arabic text and improve the performance of NLP tasks that are designed for specific type of Arabic text.

Corpora play a significant factor in the development, improvement and evaluation of many NLP applications. The limited availability of some existing resources, such as annotated corpora, and the cost of acquiring others are one of the main reasons that contribute to resource scarcity which prevents researchers from progressing further in their efforts. This need for annotated corpora, in particular, provided the motivation to create a manually POS annotated corpus for the Arabic language.

1.2 Aim and Objectives

The primary aim of this research is to investigate the development and training of a novel compression-based Arabic part-of-speech tagger using PPM. Therefore, this research's objectives are:

- Investigate the most efficient PPM compression method of Arabic text (see chapter 3).
- Investigate the applications of PPM tag-based compression to several Arabic NLP tasks (see chapters 3, 4, 6 and 7).
- Develop novel methods for classification and segmentation of Classical Arabic and Modern Standard Arabic text using PPM (see chapter 4).
- Create and evaluate a new POS manually annotated Arabic Corpus (see chapter 5).
- Develop novel compression-based criteria for evaluating Arabic part-of-speech taggers and use them to evaluate the new tagger (see chapter 6).
- Develop and train a novel compression-based Arabic part-of-speech tagger based on PPM (see chapter 7).

The main objective of this research is the development and training of a novel compression-based POS tagger for the Arabic language which is based on the PPM compression system (see chapter 7). The new tagger is evaluated with novel criteria based on the tag-based

compression results (see chapters 3 and 6). To train the new tagger, a new POS manually annotated Arabic Corpus must be created and evaluated (see chapter 5). Since the new tagger is developed to tag MSA text, the new corpus must be classified and segmented using a novel compression-based classification method (see chapters 3 and 4).

1.3 Research Questions

The specific research questions are as follows:

- Can the PPM compression models be used to help reveal linguistic universals across languages?

- What is the best PPM compression model for compressing Arabic text?

- Can the tag-based compression of the Arabic text be utilised to measure the performance of various Arabic POS taggers?

- Can two types of non-colloquial written text for the Arabic language be classified using the PPM compression models?

- Can a new POS annotated corpus be used to develop and train a new compression-based Arabic part-of-speech tagger that is effective at tagging Arabic text?

- Will the adoption of the PPM compression models to tag the Arabic text increase the performance of tagging MSA text compared to other Arabic taggers?

1.4 Contributions

The contributions of this research are as follows:

- A novel compression-based Arabic part-of-speech tagger based on PPM.

  The main contribution of this research is the development and training of a novel compression-based POS tagger for the Arabic language which is based on PPM compression system. The results of the tagger were presented in two experiments. The

first used models that were trained using silver-standard data from two different POS Arabic taggers, the Stanford [100] and the Madamira taggers [161]. The results of this experiment show that using silver-standard data to train the new tagger decreases the quality of the tag-based compression of both the CA and MSA text compared to the Madamira tagger. The second experiment trained a model using the corpus that was developed specifically for this research and forms the second contribution (see next point), where the new tagger achieved an accuracy of 93%.

- A new POS annotated corpus for the Arabic language.
  The second contribution is the creation of a manually annotated POS Arabic corpus. It is an MSA corpus that contains 50K words manually annotated by part-of-speech tags. The annotated corpus used the same tagset utilised by the Madamira tagger and followed the annotation guidelines proposed by Maamouri for annotating the POS tags. Also, a new annotation tool was developed and employed for the annotation process of the new corpus which obtained a Kappa value of 0.956, and an average observed agreement of 94.25%. The newly created corpus was used to train the new tagger and to evaluate it, and also to evaluate existing Arabic taggers.

- A new method of classifying CA and MSA text based on the PPM algorithm.
  The third contribution of this research is the development of a compression-based Arabic text classifier. This method was required to classify and segment the text of the newly developed corpus. The adoption of a PPM character-based compression scheme to classify and segment Classical Arabic (CA) and Modern Standard Arabic (MSA) texts resulted in an accuracy of 95.5%, an average precision of 0.958, an average recall of 0.955 and an average F-measure of 0.954, using the concept of minimum cross-entropy. Segmenting the CA and MSA text using the PPM compression algorithm resulted in an accuracy of 86%, an average precision of 0.869, an average recall of 0.86 and an average F-measure of 0.859.

- A novel compression-based method for evaluating the performance of Arabic POS taggers.
  The final contribution of this study is the development of a novel compression-based method for evaluating the performance of Arabic POS taggers. This method utilises the

quality of the tag-based compression of the tagged Arabic text as an indication for the quality of the tagger. This method was applied to evaluate the new tagger, and the results conclude that the use of the newly created corpus to train the new tagger increases the quality of the tag-based compression when the new tagger is used to tag MSA text.

1.5 Publications

Based on this research, three journal papers and two conference papers have already been published. All the publications are based on jointly-authored papers, where I'm the main contributor to all primary contributions presented in these publications and the co-author(s) worked in a consulting capacity, giving feedback, overall supervision and/or commentaries.

Table 1.1 shows the particular journal and conference papers which have been associated with this research. The first, entitled "Tag-based models for Arabic Text Compression", explores the approach of compressing the Arabic text using parts-of-speech (tags) along with the text to give significantly better compression results when compared to current variations of PPM, both word-based and the character-based. First, the paper explains the concept of Prediction by Partial Matching and its use for compressing natural language text. Secondly, it details the experiments on using PPM tag-based modelling to compress Arabic text. Finally, the paper mentions the results and limitations of those experiments. The paper was presented at the Intelligent Systems Conference 2017, held in London, UK, and published by IEEE. The conclusions of this paper were an essential basis for this research, as presented in chapters 3 and 7.

The second publication is titled "Compression-based Tag models for Evaluating Arabic Parts-of-speech taggers", which investigates the method of employing the compression results of the Arabic text that utilises both the POS (tags) and the text to evaluate the performance and the quality of two of the most commonly recognised Arabic POS taggers, the Madamira [161] and Stanford Arabic taggers [100]. First, the paper discusses details of the PPM tag-based compression experiments, then mentions the outcomes and limitations of these investigations. This conference paper was presented at the 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology Conference, held in Amman, Jordan.

The research in this paper was utilised to measure the performance of the main contribution of this thesis, as shown in chapter 7.

1	Title	Tag based models for Arabic Text Compression
	Authors	Ibrahim S Alkhazi, Mansoor A Alghamdi and William J. Teahan
	Submitted to	*Intelligent Systems Conference 2017*
	Year	2017
	Status	Published
2	Title	Classifying and Segmenting Classical and Modern Standard Arabic using Minimum Cross-entropy
	Authors	Ibrahim S Alkhazi and William J. Teahan
	Submitted to	*International Journal of Advanced Computer Science and Applications (IJACSA)*
	Year	2017
	Status	Published
3	Title	BAAC: Bangor Arabic Annotated Corpus
	Authors	Ibrahim S Alkhazi and William J. Teahan
	Submitted to	*International Journal of Advanced Computer Science and Applications (IJACSA)*
	Year	2018
	Status	Published
4	Title	Compression-based Tag models for Evaluating Arabic Parts-of-speech taggers
	Authors	Ibrahim S Alkhazi and William J. Teahan
	Submitted to	*2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology Conference*
	Year	2019
	Status	Published
5	Title	Compression-based Parts-of-speech tagger for the Arabic Language
	Authors	Ibrahim S Alkhazi and William J. Teahan
	Submitted to	*International Journal of Computational Linguistics (IJCL)*
	Year	2019
	Status	Published

Table 1.1. The journal and conference papers which have been associated with this research.

The third publication is titled "Classifying and Segmenting Classical and Modern Standard Arabic using Minimum Cross-entropy". This paper explores the approach of classifying Arabic text using PPM. First, the paper explains the PPM text compression scheme and its use for compressing, classifying and segmenting natural language text. Secondly, it details the findings of PPM character-based modelling experiments used to classify and segment Arabic text. Finally, the results and limitations of those experiments are discussed in detail. The paper was published in the International Journal of Advanced Computer Science and Applications (IJACSA) in 2017. This research was needed to find out the most effective way of classifying and segmenting the newly developed corpus, as presented in chapter 4.

The fourth paper titled "BAAC: Bangor Arabic Annotated Corpus" describes the creation of the new Bangor Arabic Annotated Corpus (BAAC) which is a Modern Standard Arabic (MSA) corpus that comprises 50K words manually annotated by parts-of-speech. In this paper, the new corpus was used to evaluate the widely used Madamira Arabic part-of-speech tagger and to further investigate compression models for text compressed using part-of-speech tags. Also, this paper presented a new annotation tool which was developed and employed for the annotation process of the new corpus. The paper was published in the International Journal of Advanced Computer Science and Applications (IJACSA) in 2018. This paper was needed to complete the main contribution of this thesis, as presented in chapters 5 and 6.

The main contribution of this thesis was published in the fifth paper, entitled "Compression-based Parts-of-speech tagger for the Arabic Language". The paper explores the use of compression-based models to develop and train a part-of-speech (POS) tagger for the Arabic language. The paper details the use of several models to train the new tagger. The paper also evaluates the performance of the new tagger on the two types of the Arabic text utilising the tag-based results and the newly annotated corpus, as presented in chapter 6. The paper was published in the International Journal of Computational Linguistics (IJCL) in 2019.

1.6 Organisation of this Dissertation

- Chapter 1 is an introduction to this research. It introduced the background and motivation of this study. It also introduced the aim and objectives of this research. Finally, the contributions and publication also have been listed.

- Chapter 2 surveys the literature associated with this study. First, it presents an Arabic language overview, followed by details on the PPM text compression of Arabic text. Then, the chapter reviews Arabic text classification and its applications. Next, a review on the status of the Arabic annotated resources is presented. Finally, the chapter reviews the status of the current Arabic part-of-speech taggers.

- Chapter 3 explores the approach of compressing Arabic text using parts-of-speech (tags) along with the text to give significantly better compression results when compared to current variations of PPM.

- Chapter 4 explores the approach of classifying and segmenting Classical and Modern Standard Arabic text using PPM.

- Chapter 5 describes the creation of the new Bangor Arabic Annotated Corpus (BAAC) which is a Modern Standard Arabic (MSA) corpus that comprises 50K words manually annotated by parts-of-speech.

- Chapter 6 investigates the method of employing the compression results of the Arabic text that utilises both the POS (tags) and the text to evaluate the performance and the quality of two of the most commonly recognised Arabic POS taggers.

- Chapter 7 explores the use of compression-based models to develop and train a part-of-speech (POS) tagger for the Arabic language. The chapter details the use of several models to train the new tagger and also evaluate the performance of the new tagger on the two types of Arabic text utilising tag-based results and the newly annotated corpus.

- Chapter 8 presents a summary of the thesis with suggestions for future work.

CHAPTER 2

Background and Related Work

Contents:

Arabic is a morphologically complex language that causes various difficulties for Natural Language Processing (NLP) [74], [161], [104], [38]. This chapter presents an overview of the Arabic language and surveys the literature associated with this study by investigating the application of the PPM compression system to several Arabic NLP tasks. Section 2.1 presents an overview of the Arabic language. Section 2.2 investigates PPM compression models and the three methods of compressing the text, character-based, word-based and tag-based. Specifically, it describes how to use PPM to compress Arabic text in section 2.2.1 and provides an overview of PPM in Section 2.2.2. Then, section 2.2.3 addresses the use of minimum cross-entropy concept to classify the two types of non-colloquial written text for the Arabic language, Modern Standard Arabic and Classical Arabic. Section 2.2.4 reviews language resources for the Arabic language with a focus on Modern Standard Arabic annotated resources for POS tagging. Finally, section 2.2.5 investigates the POS tagging of the Arabic language.

2.1 Arabic Language Background

2.1.1 An Overview

The Arabic language "العربية" is acknowledged to be one of the most commonly used languages, with 330 million people using the language as their first language, as shown in Table 2.1, plus 1.4 billion more using it as a secondary language [185]. Based on the number of countries and their writing system, the Arabic script is the second most popularly utilised writing system after Chinese and Latin [61]. The majority of the speakers are located across twenty-two nations, primarily in the Middle East, North Africa and Asia. The United Nations considers the Arabic language as one of its five official languages.

The Arabic language is part of the Semitic languages that includes Tigrinya, Amharic and Hebrew, and shares almost the same structure as those languages. It has 28 letters, two genders – feminine and masculine, as well as singular, dual and plural forms. The Arabic language has a right-to-left writing system with the basic grammatical structure that consists of verb-subject-object (VSO) and other structures, such as VOS, VO and SVO [24], [100], [13]. The Arabic language has had an affect on Indo-European languages such as Spanish and Portuguese, and vice versa; for example, some Arabic words were borrowed from Romance languages [204].

Rank	Language	Users (millions)
1	Mandarin	1051
2	English	508
3	Hindi	497
4	Spanish	392
5	Arabic	330
6	Russian	277
7	Bengali	211
8	Portuguese	191
9	Malay	159
10	French	129

Table 2.1. The Most Universally Used Languages [185], [135].

The Arabic script is cursive, as most of the Arabic letters are connected by methods of ligatures, and the appearances of several letters within a term depend on their location [91]. The Arabic script has 22 letters which can be connected with previous and next letters by small straight lines while the rest of the letters can be connected only to a previous letter. All the non-connecting letters of the Arabic language, which are "ادذرزو", are used in a classical Arabic poem shown in Figure 2.1.

زُرْ دَارَ وُدٍّ إِنْ أَرَدْتَ وُرُودَا

زَادُوكَ وُدّاً أَنْ رَأَوْكَ وَدُودَا

Figure 2.1. A classical Arabic poem which is written only with the non-connecting Arabic letters "ادذرزو".

2.1.2 Arabic Internet Users

Presently, the information used, collected and sent by Arabic Internet users is growing fast. Between 2000 and 2016, the increase of Internet usage in Arab countries was 4,207.4% [115]. According to the Marketing website [116], the English language comes first as the most commonly used language on the Internet representing 25.5% of the Internet users, then the Chinese language followed by the Spanish language. The Arabic language comes in fourth place with 173 million users coming from 23 countries. According to the Marketing website [116], the largest Arabic Internet users are from Egypt which accounts for 19.4% of the total number followed by Saudi Arabia then Morocco as shown in Figure 2.2.

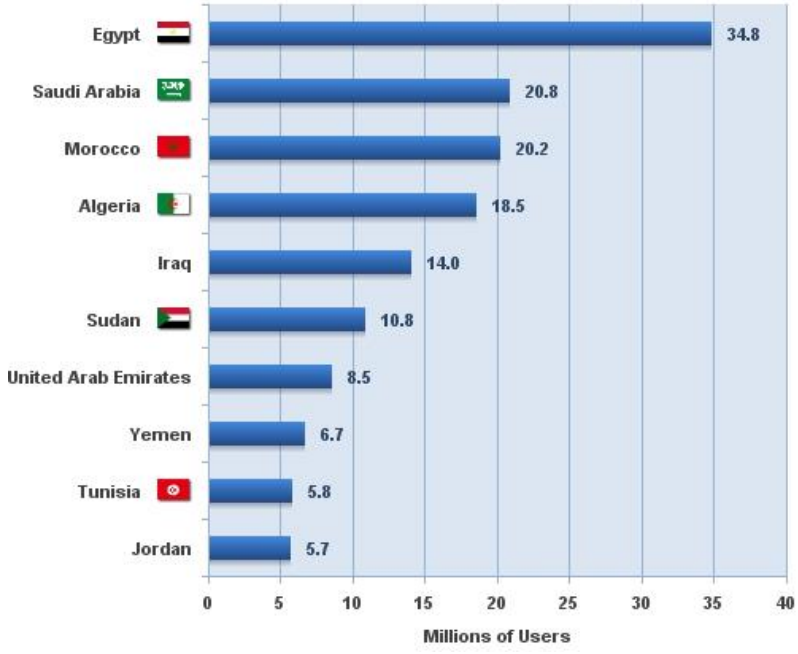


Figure 2.2. The largest 10 Arabic Internet users by countries [116].

2.1.3 Formal Written Types of Arabic language

The non-colloquial written text for the Arabic language can be divided into two types: Classical Arabic and Modern Standard Arabic [75], [156], [173], [25]. The Classical Arabic (CA) epoch is usually measured from the sixth century which is the start of Arabic literature. It is the language of the Holy Quran, the 1,400-year-old primary religious book of Islam with 77,430 words [86] and other ancient Islamic books from that era, such as the Hadith books [42]. With the beginning of journalism and the spread of literacy in the eighteenth century came Modern Standard Arabic

or MSA. MSA is the language of current printed Arabic media and most Arabic publications. (See Figure 2.3 for an example). Although MSA derives some of its attributes, such as syntax, from CA, however, MSA has a wider modern lexicon. Bin-Muqbil [57] argues that the stylistics of CA and MSA are different. Both CA and MSA are written and not spoken languages, whereas dialectal Arabic is spoken and not formally written [38], [46].

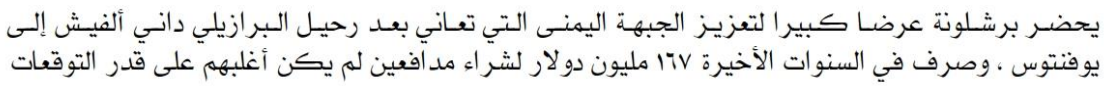

يحضر برشلونة عرضا كبيرا لتعزيز الجبهة اليمنى التي تعاني بعد رحيل البرازيلي داني ألفيش إلى يوفنتوس ، وصرف في السنوات الأخيرة ١٦٧ مليون دولار لشراء مدافعين لم يكن أغلبهم على قدر التوقعات

Figure 2.3. A sport news from aljazeera.net [28] in MSA text.

Most Arabic natural language processing (NLP) tasks perform better for MSA [84]. One example of those tasks is parts-of-speech tagging (POS) of the Arabic language as reported in [37], [42], where the performance of the taggers is best when tagging MSA text. The reason for the variation in performance between MSA and CA is that most Arabic language NLP systems were trained using MSA text [84]. More effort is currently being made, such as the creation of manually annotated CA corpora [85] and the evaluation of different Arabic POS taggers on CA text by Alosaimy and Atwell [37], to fill this gap in the research.

2.1.4 Arabic Encoding methods

The most common encoding system for the Arabic language, and for different languages as well, is UTF-8. The encoding system is able to encode all possible characters and combines various languages. The system is usually applied in multiple language applications and websites, such as Facebook and YouTube [29], [99]. Figure 2.4 shows that from 2001 to 2010, the use of other encoding systems, such as ASCII, has declined. UTF-8 uses only one byte to represent English letters, and for other languages such as the Arabic language, the system uses one to four bytes.

Microsoft developed the Windows-1256 encoding system that utilises 8-bits to represent a single Arabic character. The system can be used to represent other languages that utilise Arabic characters in their written forms, such as Kurdish, Persian and Urdu [203]. ISO 8859-6 is one of the popular character encodings systems which can be used to represent Arabic characters. Similar to Windows-1256, ISO 8859-6 utilises 8-bits to represent a single Arabic character, but

unlike Windows-1256, ISO 8859-6 is only designed for the Arabic language and cannot be used to represent other languages that utilise Arabic characters such as Kurdish, Persian and Urdu.

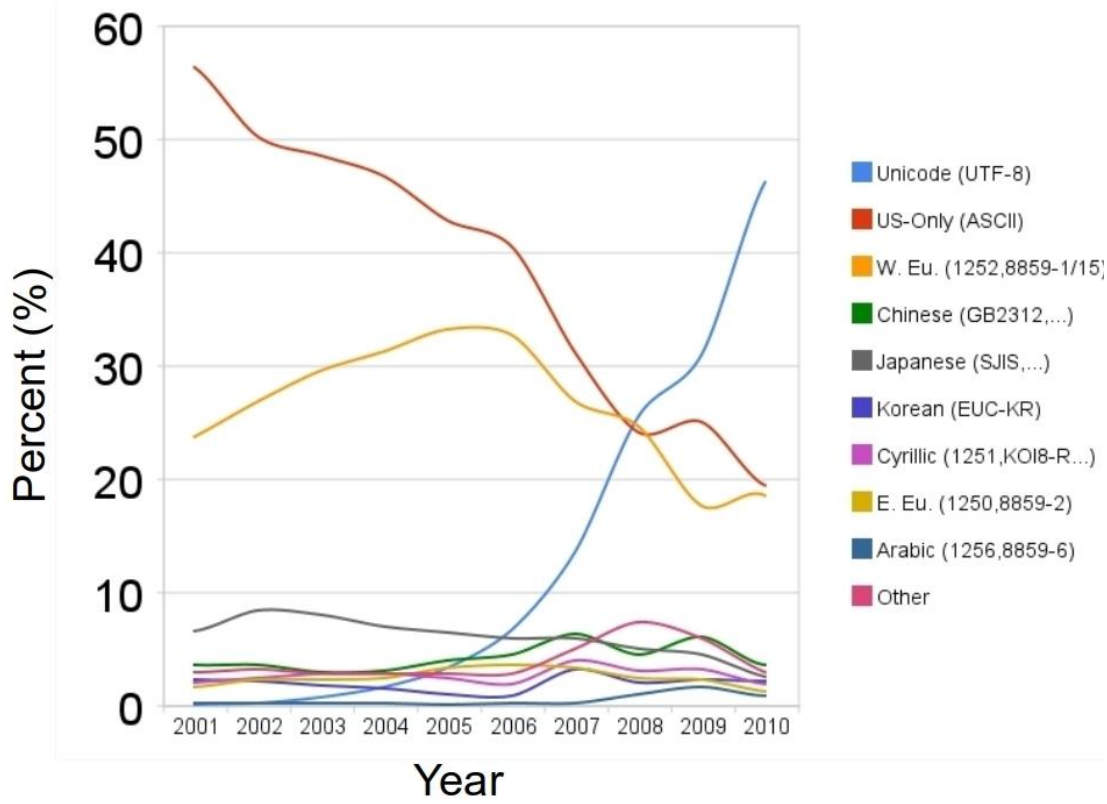


Figure 2.4. The growth of UTF-8 compared to other encoding systems [99].

2.1.5 Arabic morphology

Arabic is a morphologically complex language that causes various difficulties for Natural Language Processing (NLP) [74], [161], [104], [38]. Diacritics are used in the Arabic language to disambiguate terms. The presence of the four diacritics, which are FatHa, Dhamma, Kasra and Sukuun, in the text help in the lexical disambiguation of the word, as some words share identical component letters but different diacritics. An example of the use of diacritics to disambiguate the meaning of an Arabic term is the number of variants that the Arabic word "علم" can have with diacritics. Figure 2.5 shows 15 variants of the Arabic word "علم" where every form or variant of the term has a different meaning represented by a different use of the diacritics, and as stated before, that caused the rise in the adoption of statistical approaches to disambiguate the uncertainty of Arabic text [180].

Figure 2.5. Fifteen variants of the Arabic word "Alam" [58], which according to the diacritics used can be translated as “flag”, “scientist”, “a known place” and other translations.

2.2 Literature Review

2.2.1 PPM Text Compression of Arabic Text

2.2.1.1 Overview

The growth of exchanged information amongst Arabic users has sparked the need to reduce both the space required to store the information and the time needed to transmit that information, which thereby will help to overcome the problem of bandwidth limitation that some Arabic countries are having. This can be accomplished by compressing the stored and transmitted text [94].

2.2.1.2 Lossless Text Compression

Text compression can be defined as the process of decreasing the size needed to store the encoded text file by removing redundant information from the text, which will also reduce the time required to transmit that information. The lossless process can be reversed via decoding, to reproduce the exact original text without missing any part of the information [196]. The best lossless text compression algorithms are those that adapt to the compressed text [53]. Commonly used adaptive techniques can be classified into two classes. The first class is one that matches phrases in the text using a dictionary of phrases from already viewed text, then converts that text into a list of indexes into the dictionary. This type of technique is usually expressed as Ziv-Lempel compression [214], [215]. This way of compression does not produce

the best compression results, but is still widely used for its faster execution speed and lower resources consumption.

The best compression results are obtained using the second class that applies an adaptive statistical modelling technique. This class goes through two main steps, as shown in figure 2.6. First, a statistical model of the string seen so far in the compressed text is accumulated, and as the character is encoded, a probability distribution of the upcoming character is maintained. Then arithmetic coding is applied to encode the character which actually comes next in a near optimum way [149], [207], [171]. During the past three decades, the lossless text compression performance standard has been set by Prediction by Partial Matching (PPM) [196], which is a type of adaptive statistical modelling system.

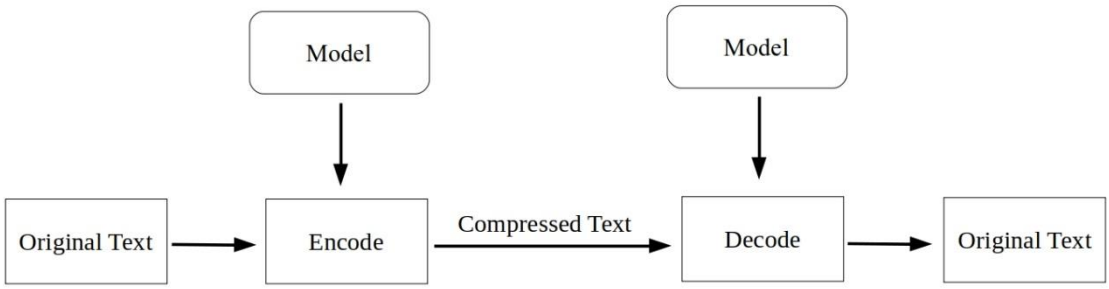


Figure 2.6. Utilising a model for text compression [53].

Text compression can be achieved in three main ways using the PPM algorithm. The first way is the use of character-based models in which the preceding context of observed symbols or characters is applied to foretell the next one. Another method of applying PPM is to use the word-based modelling of the text in which the trained model utilises the previous context of observed word or words to foretell the imminent word. The final method employs tag-based models that utilise the previously foretold tags (that represent the parts of speech) and words to predict the imminent terms (both tags and words) [196]. The concept of the tag-based method, as shown in Figure 2.7, is that recognising the tag of the term aids in predicting it. The principal advantage of employing the tag to foretell the imminent term is that the tag will in all probability have appeared many more times previously, and consequently be a better foreteller for the forthcoming tags plus terms [64], [119], [131].

The main benefit of utilizing the tag to predict the upcoming word is that the tag will in all likelihood have occurred many times beforehand, and therefore be a better predictor for the upcoming tags plus words [63], [119], [131]. On the other hand, the number of times that an individual word may have occurred is often small, and therefore is not as helpful for predicting the upcoming terms.

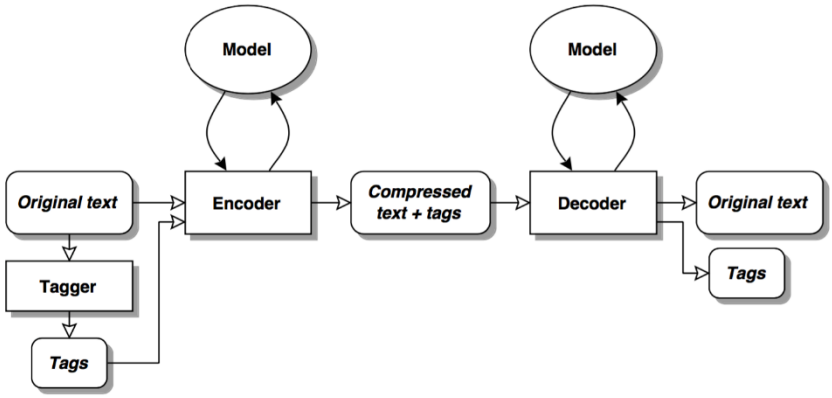


Figure 2.7. Using a tagger to compress text [190].

Previous experiments were performed by Alhawiti [26] to compare the three models on Arabic text. The experiments used various texts with different file sizes to estimate the quality of both the character and word-based text compression, whereas the text file sizes to estimate the quality of the tag-based compression were relatively small [26]. Alhawiti concluded that the character-based text compression of the Arabic text outperforms both the word-based and the tag-based compression.

The following section discusses PPM in more detail.

2.2.2 Prediction by Partial Matching

2.2.2.1 Overview

PPM is an online adaptive text compression system that foretells the upcoming character or symbol by using the previous context with given fixed length. It uses a Markov-based n-gram procedure which applies a back-off mechanism alike to that suggested by Katz [123].

Nevertheless, PPM refers to the backing-off as "escaping" and it was developed before Katz's proposed mechanism. It was first proposed by Cleary and Witten [72] in 1984 when they developed the character-based PPM variants, PPMA and PPMB. Then came Moffat and Howard, in 1990 and 1993, and introduced two further variants of PPM, PPMC and PPMD [208]. The main distinction among these variants of PPM, PPMA, PPMB, PPMC and PPMD, is the estimation of the escape probability that the smoothing mechanism requires for backing off to a reduced model's order. Experiments for character streams have shown that PPMD usually delivers better compression results when compared to other variants of PPM [124].

As stated, PPM has been successfully applied to many areas of NLP. It performs state-of-the-art compression of the text written in many languages, with results reported in [196], [26]. Another NLP application of PPM involves word segmentation of Chinese text, in this case by adding spaces to Chinese text that has no spaces [195]. Many other NLP tasks in other languages, such as code switching [46], [195], authorship attribution [190], text correction [19], cryptology [15] and speech recognition , were reported in various studies [26], [29].

The following equation is used to determine the probability p of the following character φ using PPMD [17]:

$$p(\varphi) = \frac{2c_d(\varphi) - 1}{2T_d} \tag{2.1}$$

where the coding order currently used is indicated by d , the number of times where the current context has happened or occurred in total is represented by T_d and the number of the symbol φ occurrences in the current context is represented by $c_d(\varphi)$. The estimate of the escape probability e by PPMD is as follows:

$$e = \frac{t_d}{2T_d} \tag{2.2}$$

where t_d represents the number of times in total where unique symbols occur following the current context. In most experiments, the use of 5 as a maximum order has proven to be efficient, as PPMD starts with the model's maximum order first to encode the forthcoming symbol [195].

If the forthcoming symbol was predicted by the current model and the model contained it, then its probability in current maximum order, 5 in this case, will be used to transmit it. If the

forthcoming symbol was not found in the model, then the encoder will escape to the next lower order model, 4 in this case, by encoding an escape. This process of escaping will be repeated until the model finds that symbol or prediction. If the model does not contain the symbol, then the encoder will back off to a default order of -1 [195].

2.2.2.2 Blending Techniques of PPM

PPM applies an approximate blending procedure named *full exclusion* that combines the prediction of every character and symbol of length smaller than or equivalent to m, where m is the model's chosen maximum order. The name of the technique comes from the application of the escape mechanism that escapes from the highest order prediction to lower orders until the upcoming character being predicted has been seen before. Commonly, the order 0 model is used to predict a character on the basis of its unconditional likelihood, whilst the order -1 model is applied to ensure that every potential symbol and character is assigned a finite probability [196].

PPM combines context predictions by assigning a weight to every context model, then calculating the weighted total of the probabilities. According to Bell, Cleary and Witten [53], the blended probability of character s is produced by

$$p(\mathrm{s}) = \sum_{i=-1}^{m} q_i p_i\,(\mathrm{s}) \tag{2.3}$$

where p_i represents every probability given to the order i model whereas q_i describes the weight given to model. The probabilities $p_i(s)$, which are rational, are estimated using the repetition counts $c_i(s)$. In order to prevent a probability of zero, lower order contexts are used to assign non-zero weights to the predictions.

From the escape probabilities, equivalent weights are estimated by

$$w_o = (1 - e_o) \prod_{i=o+1}^{l} e_i \qquad -1 \leq \mathrm{o} < l \tag{2.4}$$

and

$$w_l = (1 - e_l) \tag{2.5}$$

where e_o represents an escape at level *o* and the highest order context is represented by l which makes a non-zero prediction. Therefore, according to every order's escape probability, the lower orders weight is decreased one after another. Given that the escape probabilities are within 0 and 1 and e_{-1} = 0, the weights will be normalized. The weighted contribution of the model to the probability of the character s is

$$w_o p_o(\mathrm{s}) = (1 - e_o)\, p_o(\mathrm{s}) \prod_{i=o+1}^{l} e_i. \tag{2.6}$$

The sum of all weighted probabilities over each value of *o* determines the probability of the character s.

In 1990, Bell, Cleary and Witten [53] introduced the *full exclusion* mechanism which is an improvement to PPM's blending algorithm where the mechanism excludes each character predicted by higher-order contexts. The mechanism adds a small computational cost by checking all symbols for exclusion. Moffat [148] introduced the *update exclusion* mechanism which enhances the execution time of the program by not updating the counts if they are predicted by a higher order context [53] and can also lead to a slight improvement in compression by a few percent as with the full exclusion mechanism.

The next sections will discuss the three PPM methods of modelling.

2.2.2.3 Character-based Modelling

To explain character-based encoding in more detail, Table 2.2 presents the way PPM models a given string. The example in this case is how the PPMD prediction method models the string `tobeornottobe`. The model in this example uses a maximum order of 2 for illustration purposes (although normally it would be order 5). In the table, *c* indicates the count, *p* symbolizes the probability and |*A*| is the size of the alphabet that is used [195]. Let the imminent character for this example be letter *o*. The letter has been seen once before ('*be*' → *o*) for the order two context '*be*' and therefore it has a probability of $\frac{1}{2}$ (applying equation (1) since the

count is 1). Therefore, the letter *o* will be encoded using 1 bit. But, if the upcoming letter in the order two context had not been seen before, (i.e. suppose the next letter was *t* rather *o*), then the model would need to escape to a lower order, the escape probability will be $\frac{1}{2}$, and the model will back off to the order 1 context.

Order 2				Order 1				Order 0				Order -1		
Prediction		*c*	*p*	Prediction		*c*	*p*	Prediction		*c*	*p*	Prediction	*c*	*p*
'be'	→ o	1	1/2	'b'	→ e	2	3/4	→ b		2	3/26	→ *A*	1	1/|*A*|
	→ esc	1	1/2		→ esc	1	1/4	→ e		2	3/26			
'eo'	→ r	1	1/2	'e'	→ o	1	1/2	→ n		1	1/26			
	→ esc	1	1/2		→ esc	1	1/2	→ o		4	7/26			
'no'	→ t	1	1/2	'n'	→ o	1	1/2	→ r		1	1/26			
	→ esc	1	1/2		→ esc	1	1/2	→ t		3	5/26			
'ob'	→ e	2	3/4	'o'	→ b	2	3/8	→ esc		6	3/13			
	→ esc	1	1/4		→ r	1	1/8							
'or'	→ n	1	1/2		→ t	1	1/8							
	→ esc	1	1/2		→ esc	3	3/8							
'ot'	→ t	1	1/2	'r'	→ n	1	1/2							
	→ esc	1	1/2		→ esc	1	1/2							
'rn'	→ o	1	1/2	't'	→ o	2	1/2							
	→ esc	1	1/2		→ t	1	1/6							
'to'	→ b	2	3/4		→ esc	2	1/3							
	→ esc	1	1/4											
'tt'	→ o	1	1/2											
	→ esc	1	1/2											

Table 2.2. Processing the string `tobeornottobe` using PPM [205].

When the model backs off, the new order will be used to estimate the probability, and in this case, there is no letter *t* that comes after *e*. As a result, the model will encode another escape using a probability of $\frac{1}{2}$, and the context will be reduced to the null (order 0) context. Letter *t* will be encoded using this order, where the probability will be $\frac{5}{26}$. The total cost of predicting the last letter will be $\frac{5}{26} \times \frac{1}{2} \times \frac{1}{2}$, which in this case will be over 4 bits (since $-\log_2 \frac{5}{104} = 4.28$ bits). Moreover, if the following letter has not been seen before in the context, such as the letter *x*, the escape probability will be encoded three times from the maximum order of 2 down to -1 with the

following probabilities: $\frac{1}{2} \times \frac{1}{2} \times \frac{3}{13} \times \frac{1}{256}$ since order -1 will be used to encode this letter as we are encoding English characters using ASCII (with an alphabet size of 256), and this will require over 12 bits [195].

2.2.2.4 Word-based Modelling

For word-based encoding, a similar PPM-based approach is used to make the predictions, but one word at a time rather than a character at a time. A bigram word model is as follows:

$$p(s) = \prod_{i=1}^{m} p(w_i \mid w_{i-1}) \tag{2.7}$$

where $s = w_1 \ldots w_m$ is the text of m words being predicted. When a new bigram sequence $w_{i-1}w_i$ or a new word w_i is encountered, the model will escape to an order 0 model, and if needed to a standard PPM character model (where its characters are encoded one character at a time) in order to predict the unseen word or bigram.

2.2.2.5 Tag-based Modelling

The tag-based model can be represented as follows:

$$p(s) = \prod_{i=1}^{m} p(t_i|t_{i-1}t_{i-2}) \times p(w_i \mid t_i w_{i-1}) \tag{2.8}$$

where again $s = w_1 \ldots w_m$ is the text of m words being predicted. The tag-based model uses two streams, a tag stream and a word stream, to predict the upcoming word as shown in Figure 2.7. First, it will use an order 2 PPM model to predict the tag given the two previous tags, then predict the upcoming word given its tag along with the previously seen word. If the model has not seen the $t_i w_{i-1}$ sequence or its prediction w_i, an escape probability will be encoded and the model will try to continue predicting the following word using only the current tag. Lastly, if the prediction fails, it will escape to a character-based model [196]. The method requires that the text sequence of words is tagged first, and then effectively both the tag and word sequences need to be encoded together with the extra tag information also becoming available to the decoder as shown in Figure 2.7. If the extra tags improve the compression (compared to a word-based or character-based compression which do not need to encode the extra tag

information), then this helps support the linguistic validity of the tag information. Prior experiments [196] show that it is possible to get better compression results using tag-based compression compared to both word and character-based compression for tagged English text, but previous experiments with both Chinese [208] and Arabic tagged text [26] have not been able to reproduce the English results for these languages.

2.2.2.6 The Tawa toolkit

The Tawa toolkit [192] can be used to apply PPM modelling to many different NLP tasks. According to Teahan [192], "*The aim of the toolkit is to simplify the conceptualisation and implementation for a broad range of text mining and NLP applications involving textual transformations*". The toolkit can be used to implement a wide spectrum of NLP applications and it comprises eight principal applications, as shown in Figure 2.8, such as `train`, `encode`, `decode` and `classify`. It adopts a 'noiseless channel model approach', as illustrated in Figure 2.9, where every application is conceived as an encoding process without loss of any information and any procedure is reversible. The algorithms and pseudo-code of the encoding, decoding, training and six other applications are described in detail by Teahan [192]. Other details, such as the implementation aspects and search algorithms applied in the toolkit, are also addressed by the developer.

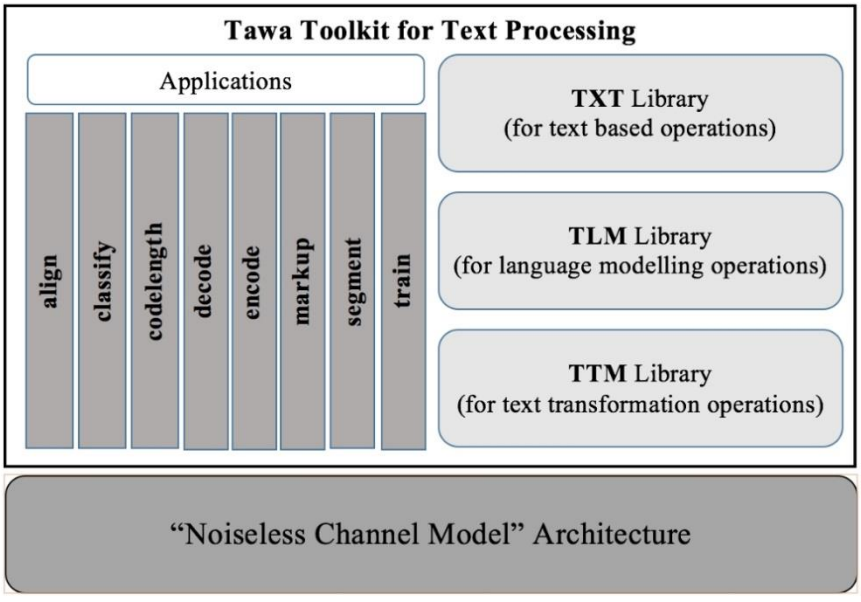


Figure 2.8. A diagrammatic representation of the Tawa Toolkit design [192].

Previous experiments were performed by Alhawiti [26] to compare the three models on Arabic text. Alhawiti used various text file sizes to estimate the quality of both the character and word-based text compression, whereas the only resource used to estimate the tag-based text compression was only the first part of the Arabic Treebank Corpus (ATC). As stated before, PPM is an online adaptive text compression system that needs relatively large amounts of training data to learn from and build the tag-based models. The reason for the use of the ATC corpus in the previous experiments is the fact that the resources for manually tagged Arabic corpus are limited, and the existing manually tagged corpora are usually relatively small [26]. Therefore, the effect of using the tag-based models to compress larger Arabic tagged text needs be investigated further to re-estimate the performance of the previous three PPM methods of compressing the Arabic text and to produce more comparative results.

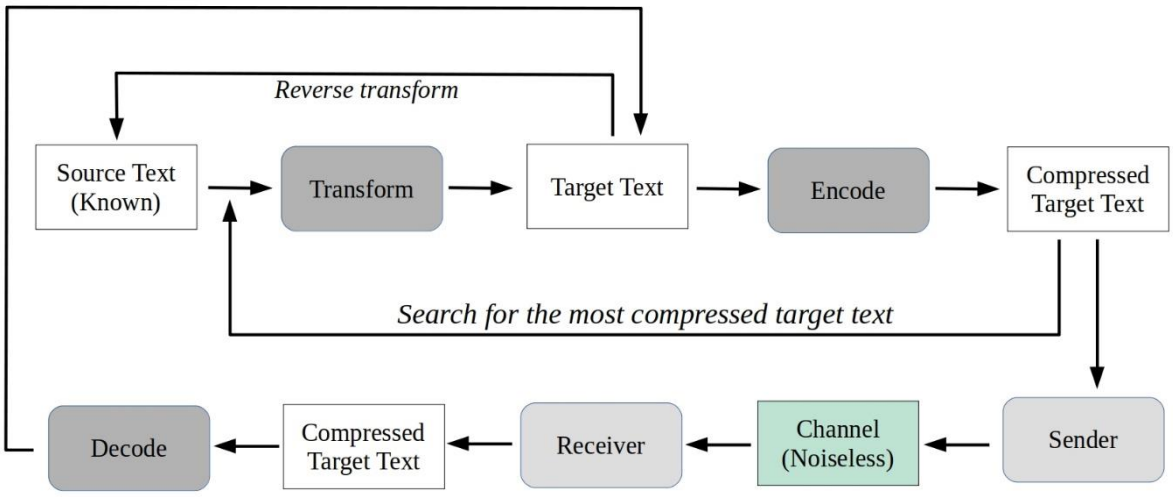


Figure 2.9. The encoding-based 'Noiseless Channel Model' used by the Tawa Toolkit [192].

2.2.3 Arabic Text Classification

2.2.3.1 Overview

Text classification is the process of automatically assigning a document to different predefined classes or categories to reflect their contents [194]. Text classification is important in various areas such as natural language processing (NLP), text mining, information retrieval, machine learning, etc. [188]. It also can be applied in a large variety of applications such as spam filtering

[12], author identification [121], [186], [47], gender identification [68], [44], sentiment analysis [4], [7], [14], [44], dialects identification [212], [143], and so on.

The massive increase in the size of text accessible on the internet during the last two decades has drawn the attention to the importance of text classification [188]. This increase of data on the Web has produced the need for methods to extract the required information from text documents, and therefore, generating unique difficulties for the text classification problem especially when considering applications requiring analysis of big data [129], [188].

Text classification can be implemented using various algorithms, for example, Naïve Bayes and the chain augmented Naïve Bayes probabilistic classifier [95], [163]. Other algorithms such as support vector machines, or SVM, [113], generalized instance sets [132], k-nearest neighbors algorithm [113], neural networks [172] and Generalized Discriminant Analysis, or GDA, [136] have been used to classify English text. Various algorithms have also been applied to other languages such as Chinese [113], [201] although there has been noticeably less research done with the Arabic language.

Most of these text classification algorithms handle text documents as a “bag-of-words” where a set of words or tokens are used to interpret the text and which rely on using their frequency in some manner [83], [179]. The traditional approach to text classification goes through four steps: first, pre-processing of the text where the words (or tokens) and sentences in the training files are segmented [194], [87]; second, using word/token counts to extract or select different features; thirdly, applying one of the machine learning algorithms mentioned earlier; and finally, performing the same feature extraction on the test data and applying the learned model to the extracted features to predict the class for the test data [194], [87].

During the process of analyzing the text, a complication occurs when the phenomenon of code-switching arises. This is where a text contains more than one language or variations of the same language. This phenomenon has been the subject of extensive linguistically oriented study in the past [98], [177], and the problem of mixed texts must be tackled by segmenting those variations. Text segmentation is the task of automatically separating the text into identified or coherent parts [52]. Compared to text classification, text segmentation can be used to produce

a more accurate estimate of each class, category or topic located inside the text rather than assigning a class or set of classes to the entire text as a whole.

Many segmentation algorithms, such as the TextTiling algorithm [114] and the dotplotting algorithm [169] rely on measuring the variation in word usage to predict potential boundaries in the text, where a vast difference in word usage is a positive sign. Kozima [130] introduced an algorithm that traces the coherence of a document by applying a semantic grid in a “lexical coherence profile”. A statistical approach was proposed by [52] for text segmentation, where the algorithm builds a model from selected informative features, then the model is used to predict where boundaries happen in the text.

Compared to the traditional way of text classification, compression-based language modelling uses a character-based approach, whereas traditional text classification is a word-based approach which is language-dependent that tends to overlook both the contextual information of the text and the word order [194], [188]. The use of language modelling for text classification takes into consideration the contextual information in the text when building the language model and avoids the need for pre-processing of the text usually required by most classification algorithms [194], [188]. The use of Markov-based approximations standard in character-based language modelling avoids the issue of explicit feature selection that is applied in traditional classification and segmentation algorithms which may discriminate some important features of the text [194], [120]. Algorithms that adopts a Viterbi-based algorithm produces an accurate estimate of each class, category or topic located in the text. [197].

2.2.3.2 Minimum Cross-entropy as a Text Classifier

The concept of minimum cross-entropy as a text classifier has been adopted in various NLP tasks that utilises the PPM algorithm [35], [46], [196], [191], [71]. The basis of the classification and segmentation schemes in the PPM algorithm uses the character-based approach for compressing the Arabic text [194]. The essence of this approach depends on the concept of entropy as a measurement of the message's “information content” [182], and on the notion that the upper bound of the entropy can directly be estimated by compressing the text [64].

The fundamental coding theorem in information theory [182] states that an entropy of a sequence of text, or message, is the lower bound to the average number of bits per character required to encode that message [197].

$$H(P) = -\sum_{i=1}^{k} p\,(x_i) \log p(x_i) \tag{2.9}$$

where there are k number of potential characters with the probability distribution $P = p(x_1), p(x_2),\ldots,p(x_k)$ and the probabilities sum to 1 and are independent. The measurement of the uncertainty associated with the selection of the characters is represented by the entropy, where the higher the entropy, the higher the uncertainty. The message's "information content" can also be measured by the entropy, as the more probable the messages, the less information is conveyed compared to less probable ones [197].

For simplification purposes, the sums displayed in following formulas are considered to be made over all potential sequences. A general case for a language with probability distribution can be extended from the previous equation for a text sequence $T = x_1, x_2, \ldots, x_m$ of length m:

$$H(L) = \lim_{m\to\infty} -\frac{1}{m}\sum p\,(x_1, x_2, \ldots, x_m) \log p(x_1, x_2, \ldots, x_m). \tag{2.10}$$

This describes the entropy of a language which is defined to be the limit of the entropy when the size of the message becomes large. The probability distribution for the source language L is usually not identified or known. Nevertheless, applying a model M as an approximation to the probability distribution gives the upper bound to $H(L)$ [197]:

$$H(L, M) = -\sum P_M\,(x_1, x_2, \ldots, x_m)\; log\; P_M(x_1, x_2, \ldots, x_m) \tag{2.11}$$

where $P_M(x_1, x_2, \ldots, x_m)$ is used to estimate the probabilities. $H(L, M)$ is described as the cross-entropy which is higher than or equivalent to the entropy $H(L)$, as this is based on the source itself which is the best possible language model:

$$H(L) \leq H(L, M).$$

Compressing the text can be used to estimate an upper bound to the entropy of a message [64]. Considering the number of bits needed to encode a sequence of text to be $b_M(x_1, x_2, \ldots, x_m)$, when using some model M to estimate the probabilities, then:

$$H(L,M,T) = \lim_{m\to\infty} \frac{1}{m} b_M(x_1, x_2, \dots, x_m) \quad (2.12)$$

where the number of bits per character needed to encode a long text message T formed from L is $H(L,M,T)$.

The cross-entropy is important as it presents a measurement of how great the estimated model is performing on the test text; the less inexact the model is, the closer the cross-entropy is to $H(L)$. Furthermore, by measuring the cross-entropy for every possible model, the cross-entropy provides a valuable measure for analysing the correctness of the competing models. The model that has the least cross-entropy is judged to be the "best" or most appropriate. The information is derived from a semantic label which is associated with each model which reflects the class or type of data that was used to train the model. Simply, the label linked with the "best" model is selected and used to classify the text:

$$\hat{\theta}(T) = \operatorname{argmin}_i H(L,M,T). \quad (2.13)$$

2.2.3.3 Minimum Cross-entropy as an Arabic Text Classifier

Almahdawi and Teahan [35] have successfully adopted a PPM character-based text compression scheme for coarse-grained and fine-grained classification of emotions in the text that includes the six Ekman's emotions (Sadness, Disgust, Anger, Surprise, Happiness and Fear). They reported that utilising the PPM as a classifier outperformed the conventional word-based text classification schemes. Altamimi and Teahan [46] have successfully classified gender and authorship of Arabic tweets using an order 11, PPMD model achieving an accuracy of 90% and 96% respectively.

Some Arabic corpora, such as the Bangor Arabic Compression Corpus (BACC), is a mixture of both CA and MSA text. An example is the BACC sub-corpus *'Arabic_book1'*, which contains both recent novels with ancient Arabic poems. (See Figure 2.10 for one example). The results of using such a corpus in order to perform various NLP tasks, such as POS tagging, as stated before, will vary and will not be consistent and reliable. Consequently, NLP applications should

treat these texts separately and use different training data for each or process them differently. Therefore, there arises a need to accurately classify CA from MSA within the text.

يُبكون من تبكي السماواتِ يومَه ومن قدِ بكته الأرض فالناس أكمد
وهل عدلت يـوما رزيةُ هالك رزيـــة يـــوم مـــات فيـــه محمد

Figure 2.10. A Classical Arabic eulogy poem from the BACC.

2.2.4 Arabic Annotated Corpus

2.2.4.1 Overview

The term *corpus* can be defined as a computerised set of genuine texts or discourses provided by language speakers that is saved in a machine-readable form [117], [147], [209], [9]. Xiao [210] argues that a corpus is not a randomly selected collection of texts nor an archive, but a file that manifests four essential aspects as follows: a corpus is a set of (1) machine-readable (2) genuine texts (that includes transcripts of spoken data) that are (3) tested to be (4) representative of a specific or a group of languages.

Corpora play a significant factor in the development, improvement and evaluation of many NLP applications such as machine translation [30], [211], part-of-speech tagging [180] and text-classification [211]. The design of any corpus depends on its intended applications [48]. Some corpora are for general use and can be utilised in many applications, and others may serve a specific purpose, such as building dictionaries or examining the language of a specific author or duration of time [42].

There are several kinds of annotations which could be applied to corpora, and each annotation is usually designed to handle a certain aspect of the language [146]. One type of corpora annotation is the structural annotation of the corpus by attaching descriptive information about the text, like mark-ups that specify the boundaries of the sentence, section and chapter, or a header file that names the author of the text or adds information about participants, such as the age and gender. Another type of annotation is the morphological annotation, where information

about the text, like the stems or root based in a language like Arabic, is added to the corpora. This type of annotation is the most common type of corpora annotation, and the most common type of morphological annotation is POS tagging of the text [146], where a tag, such as a noun, verb or particle is combined with each term in the corpus, and the number of tags used in the annotation varies from a few to 400 tags or more [105].

Based on the type of text and purpose(s) for being created, a corpus can be categorised into four categories: Raw Text Corpora; Annotated Corpora; Lexicon Corpora; and Miscellaneous Corpora. Examples of corpora for the Arabic language are provided below.

1. Raw Text Corpora can be divided into:

   A. **Monolingual corpora**, such as the BACC [26], Ajdir Corpora [5], the King Saud University corpus of Classical Arabic [43], Alwatan [2], Tashkeela [213] and the Al Khaleej Corpus [3]. Monolingual corpora consist of a raw text written in a single language.

   B. **Multilingual corpora**, also known as comparable corpora or parallel corpora, are corpora that are written in two or more languages. Multilingual corpora, such as the UN corpus [202] which is the most important and widely known free corpus [211], Corpus A [30], the Hadith Standard Corpus [34], [181] and MEEDAN Translation Memory [96], are used in NLP fields such as machine translation [30], [211].

   C. **Dialectal Corpora**, where the corpus is written in a specific language dialect, such as the Bangor Twitter Arabic Corpus for Egyptian, Gulf, Iraqi, Maghrebi and Levantine Arabic dialects [45]. Other well-known dialectal corpus for Arabic is the Shami corpus for Levantine Arabic dialects created by Abu Kwaik and others [8], and the Arabic Dialects Dataset collected by El-Haj [88]. Such corpora are used in fields such as text-classification.

   D. **Web-based corpora**, such as the KACST Arabic Corpus [17], the Leeds Arabic Internet Corpus [16] and the International Corpus of Arabic [20], where the corpora are only accessible online by an inquiry interface and the corpora cannot be downloaded.

2. The second type is Lexicon corpora, that can be divided into:

A. **Lexical Databases**, such as the BAMA 1.0 English-Arabic Lexicon [138] and the Arabic-English Learner's Dictionary [164].

B. **Words Lists** such as the Word Count of Modern Standard Arabic [43] and the Arabic Wordlist for Spellchecking [49], [77].

These types of corpora act like a vocabulary or a list of words and can be employed by linguists to study many aspects of a language or combined with the lexicons of systems, like spell checking applications, to improve their performance [211].

3. **Miscellaneous Corpora**, such as Speech Corpora [36], Handwriting Recognition Corpora [139], are beneficial for a number of NLP tasks such as plagiarism detection [56], speech recognition systems [36] and question answering [55].

4. **Annotated corpora** are essential for the development of many NLP systems, such as part-of-speech tagging [180], text parsing [69]. Annotated corpora are divided into:

A. **Named Entities Corpora** such as JRC-Names [187] and ANERCorp [54]. Most corpora of this type include the names of persons with the company or organisation name and the locations.

B. **Error-Annotated Corpora**, such as the KACST Error corpus [33], is a beneficial resource for systems such as spelling correction and machine translation corrected output [118].

C. **Miscellaneous Annotated Corpora**, such as the OntoNotes corpus [166] and the Arabic Wikipedia Dependency Corpus [151] which are semantically annotated corpora [166].

D. **Part-of-Speech (POS)** tagged corpora are an important resource for the training and development of POS systems [180]. Some of these resources will be presented in detail in the existing resources section below.

2.2.4.2 POS Arabic Annotated Corpora

POS annotated corpora are essential for the development of many NLP systems, such as part-of-speech tagging [180], statistical modelling [111]. The lack of such resources limits some researchers from progressing further in their efforts. The limited availability of some existing annotated corpora and the cost of acquiring others are one of the main reasons that contribute to resource scarcity. Several efforts have been made to overcome the lack of resources [37], [9], [85].

In 2001, the Linguistic Data Consortium (LDC) published the first versions of the Penn Arabic Treebank (ATB) [141], as illustrated in Table 2.3. This resource is widely used in many Arabic NLP applications such as the training of POS taggers, like the Madamira Arabic POS tagger [161] and the Stanford Arabic POS tagger [100]. The corpus consists of three parts with a total of 1 million annotated words. The first part v2.0 was a newswire text written in Modern Standard Arabic and consisted of 166K terms acquired from the Agence France Presse corpus. The second part was obtained from the Al-Hayat corpus which was distributed by Ummah Arabic News Text and consists of 144K terms [141]. The last part of the ATB corpus, part 3 v1.0, as shown Figure 2.11, is a newswire text obtained from the An-Nahar corpus and consists of about 350K morphologically annotated words. For non-members of the LDC, the cost of acquiring any part of the ATB corpus exceeds several thousand US dollars which prevents access to researchers with a limited budget [111], [141].

```
<Annotation id="DOC_ANN20020415.0083:AG2:Annotation14" type="solution"
            start="DOC_ANN20020415.0083:AG2:Anchor3"
                  end="DOC_ANN20020415.0083:AG2:Anchor4">
              <Feature name="solution">(ra}iysa) [ra}iys_1]
                    ra}iys/NOUN+a/CASE_DEF_ACC</Feature>
              <Feature name="gloss">president/head/chairman + [def.acc.]</Feature>
              <Feature name="number">3</Feature>
</Annotation>
```

Figure 2.11. A sample POS tag from the ATB Part 3 v 1.0.

Khoja [127], [125], [92] has published a 50,000 terms manually annotated POS tagged corpus written in MSA text. According to the author, the corpus is divided into two parts. The first part

is a newspaper text consisting of 1,700 terms that are manually tagged using a tagset that differentiates between the three moods of the verb and case structures of the noun [112]. The second part of the corpus is tagged using a simple tagset that includes only the following POS tags: noun, verb, particle, punctuation or number [125]. However, access to this resource was not provided for this study.

Corpus	Part	Text type	Number of terms	Source of text	Notes
Treebank (ATB)	One	MSA	166K terms	Agence France Presse corpus	Corpus fee is $4,500
	Two		144K terms	Al-Hayat corpus	Corpus fee is $4,000
	Three		350K terms	An-Nahar corpus	Corpus fee is $3,500
Khoja POS annotated corpus	One		1,700 terms	Newspaper text	Tagset consists of three moods of the verb and case structures of the noun
	Two		48,300 terms	Newspaper text	Tagset consists only of noun, verb, particle, punctuation or number
The AQMAR Arabic Wikipedia Dependency Tree Corpus	-		36K terms	Arabic Wikipedia articles	The tagset used in this corpus contains a small number of tags
The Columbia Arabic Treebank (CATiB)	-		-	Newswire feeds from 2004 to 2007	The tagset consists only of six POS tags.

Table 2.3. A table summary of different Arabic annotated corpora.

Another annotated corpus was published by Mohit [151]. The AQMAR Arabic Wikipedia Dependency Tree Corpus is a manually annotated corpus that contains 1262 sentences collected from ten Arabic Wikipedia articles and the 36K terms of the corpus are manually annotated using the Brat annotation tool [151]. The ten articles cover topics such as the Internet, Islamic Civilisation and Football and were annotated for named entities beforehand [178], [152], [153]. The tagset used in this corpus contains a small number of tags and therefore is not as

useful for the research concerning tag-based text compression described in Chapter 3 which requires much larger amounts of training data to be effective.

The Columbia Arabic Treebank (CATiB) [105] is another manually annotated Treebank corpus that consists of newswire feeds, from the year 2004 to 2007 and written in MSA. The corpus was initially tokenized and then POS tagged by the MADA&TOKAN toolkit [104], [105]. The TrEd annotation interface [165] was utilised in the annotation process. The number of tags used by CATiB is relatively small as it consists only of six POS tags, NOM, PROP, VRB, VRB-PASS and PRT, where each tag comprises a group of subtags; for example, the tag "NOM" can be used to tag nouns, adverbs, pronouns and adjectives.

There exist some annotated corpora for the Arabic language that cannot be utilised by many researchers, such as the tag-based text compression research applied by Alhawiti [26] due to availability, and cost issues, such as the Arabic Treebank corpus [141]. Other resources are designed to be used for particular research or annotated using a distinctive tagset produced for an explicit purpose. The Qur'anic Arabic Dependency Treebank is one example where the text is written in CA text and the corpus uses a tagset which is designed to tag CA text using traditional Arabic grammar [85], [30]. This need for annotated corpora, which are necessary for the development of many NLP systems, provided the motivation to create a manually annotated corpus for the Arabic language for this study (see Chapter 5).

2.2.5 Arabic Part-of-speech Tagging

2.2.5.1 Overview

A parts-of-speech (POS) tagger is a computer system that accepts text as input and then assigns a grammatical tag, such as VB for a verb, JJ for an adjective and NN for a noun, as output for every token or term according to its appearance, position or order in the text. POS tagging is normally an initial step in any linguistic analysis and a very important early step in the process of building several natural language processing (NLP) applications, such as information retrieval systems, spell auto-checking and correction systems and speech recognition systems [10]. Alabbas and Ramsay [18] argue that higher tagging accuracy improves the quality of all

subsequent stages and, therefore, assessing the tagger accuracy is an important step in the development of many NLP tasks.

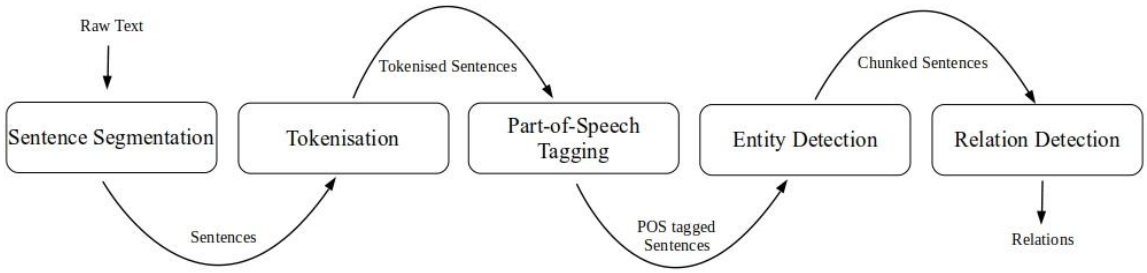


Figure 2.12. Simple information retrieval system pipeline architecture [158].

The tagging process can be achieved by one of the following general methods: (1) a statistical approach where a language model is trained using previously tagged corpora, such as the Arabic Treebank [111], and the model is then used to tag different text; (2) a rule-based approach where linguists define and develop rules or a knowledge base, as shown in Figure 2.13, which are used to assign POS tags; and (3) by combining the previous two approaches in a hybrid system [125], [38], [108], [93], [180].

2.2.5.2 Statistical POS tagging

The earliest approach used for developing POS taggers is the rule-based method [126], [125], [10], that was first developed in the 1960s. As stated before, this method utilises a collection of linguistic rules, where the number of rules ranges from hundreds to thousands, to tag the text. The development of a rule-based tagger is difficult, costly and the system is usually not robust [10]. Brill [60] developed the TBL rule-base tagger that obtained a tagging accuracy similar to that of statistical taggers. Unlike statistical taggers, the linguistic knowledge is created automatically as Brill's tagger trains simple non-stochastic rules [60]. Other examples of rule-base taggers are the CGC tagger developed by Klein and Simmons [128], the TAGGIT tagger which was produced by Greene and Rubin [102]. Nguyen and others have developed a rule-based POS tagger that utilises an SCRDR tree [170], as shown in Figure 2.13, to represent the rules used by the RDRPOSTagger [157]. RDRPOSTagger was utilised to tag two languages, English and Vietnamese, with a reported accuracy of 93.51% for the English language. The

tagger uses an error-driven procedure to build the knowledge base automatically in the form of a binary tree as shown in Figure 2.13.

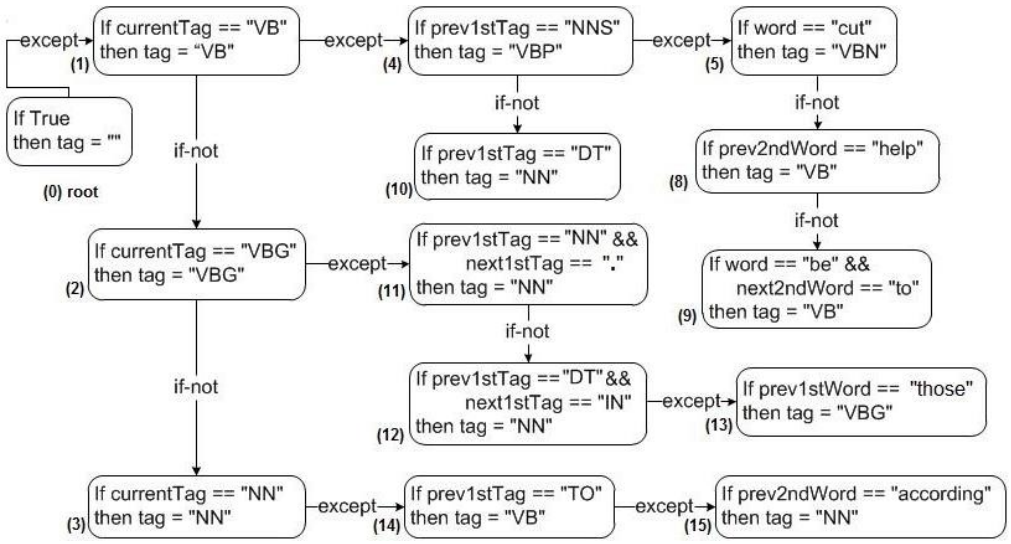


Figure 2.13. A sample of RDRPOSTagger tagging rules [157].

In the 1990s, the statistical approach of tagging the text started to replace the earliest approach used for developing POS taggers, and according to Martinez [144], the statistical approach also started to be adopted more with several other NLP tasks, reporting state-of-the-art results. For the Arabic language, the statistical method of tagging the Arabic text is largely utilised to solve the POS ambiguity of the Arabic text [180].

2.2.5.3 POS tagset

The tagset is a list of all the potential tags which could be assigned to the terms during the tagging process and it is regarded as a fundamental component for any POS tagger. For the English language, there are a number of common tagsets which have been developed and used by English POS taggers; for example, the Brown tagset used in the Brown corpus which comprises 226 tags, the LOB tagset used in the LOB corpus, which is based on the Brown tagset, containing 135 tags [97], and the Penn Treebank tagset which was used to tag the Penn Treebank corpus and contained 36 tags [189].

For the Arabic language, tagsets can be divided into traditional and English derived tagsets [38]. English derived tagsets arose when Arabic resources were limited, and a tagset was urgently

needed to develop new resources [111], [140], [82]. This type of tagset is usually a trivial modification of the standard English tagset, and this modification was considered problematic for Semitic languages as stated by Wintner [206], and illustrated by Alosaimy who showed that in some cases differentiation among adjectives and nouns is unclear [38]. Many more language specific tagsets for the Arabic language have been proposed; for example, the Khoja tagset utilised by the APT tagger includes 177 tags [125], [126], as shown in Figure 2.14. The El-Kareh and Al-Ansary [90] tagset comprises 72 tags used in their tagger. Al-Shamsi and Guessom [180] proposed a tagset that includes 55 tags, which was employed in the HMM tagger that they have developed. Finally, Al-Qrainy [39] proposed a tagset that was used in AMT tagger that comprises 161 detailed tags and 28 general ones, as displayed in Table 2.4.

Tagset name	**Utilised by**	**Number of tags**
Khoja tagset	APT tagger	177 tags
The El-Kareh and Al-Ansary tagset	The El-Kareh and Al-Ansary tagger	72 tags
Al-Shamsi and Guessom tagset	Al-Shamsi and Guessom HMM tagger	55 tags
Al-Qrainy tagset	AMT tagger	161 detailed tags and 28 general tags
SALMA tagset	-	Five main POS categories

Table 2.4. A table summary of different Arabic tagsets.

Sawalha and Atwell [176] introduced the SALMA tagset, which according to the authors, "captures long-established traditional morphological features of Arabic, in a compact yet transparent notation". The tagset includes 22 characters where each position serves a feature and the character at that position serves a morphological feature value or attribute. Figure 2.15 shows the main POS category of the SALMA tagset described at position 1. The tagset is bound to a particular tagging algorithm and other tagsets can be mapped onto the SALMA tagset standard according to the authors. The tagset was validated and utilised in various Arabic language processing systems [174], [175].

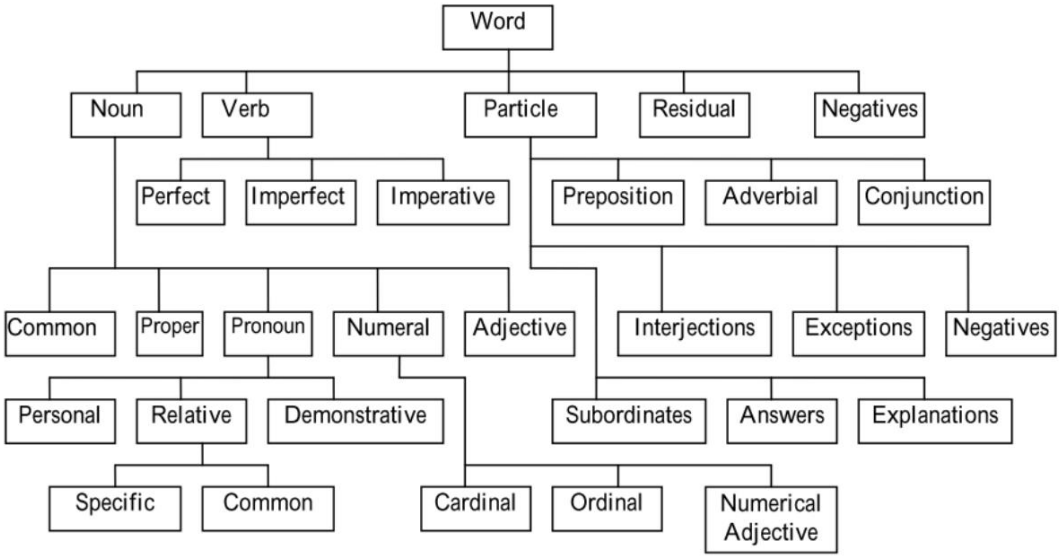


Figure 2.14. The main POS category of the Khoja's Tagset [27], [38].

Table 2.5 shows a sample of Arabic text tagged by three tagsets, the Madamira tagset, the Stanford tagset and Farasa tagset. The Farasa tagset [38], [6] consists of 16 primary tags. Pasha and others [161] introduced the Madamira tagset, which was used initially by the MADA tagger [104]. The tagset is the subset of the English tagset which was presented with the English Penn Treebank and consists of 32 tags and was initially proposed by Diab, Hacioglu and Jurafsky [80]. The Stanford tagset consists of 24 tags. Those tags are derived by manually decreasing the 135 tags obtained from the Arabic Treebank distribution [93].

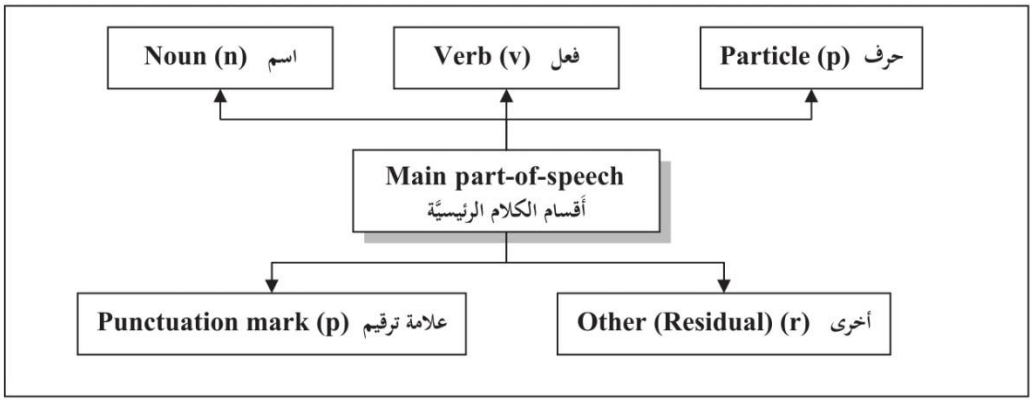


Figure 2.15. The main POS category of the SALMA tagset [176].

Term	Madamira Tag	Stanford Tag	Farasa Tag
وقد	part_verb	NN	PART
اتخذت	verb	VBD	V
خطوات	noun	NNS	NOUN-FP
بإنشاء	noun	NN	NOUN-MS
لجنة	noun	NN	NOUN-FS
الحقيقة	noun	DTNN	NOUN-FS
والمصالحة	noun	NN	NOUN-FS
واللجنة	noun	NN	NOUN-FS
الوطنية	adj	DTNN	ADJ-FS
المستقلة	adj	DTJJ	ADJ-FP
لحقوق	noun	NN	NOUN-FS
الإنسان	noun	DTNN	NOUN-MS

Table 2.5. Samples of various Arabic tagsets.

2.2.5.4 Statistical Arabic POS Taggers

The Madamira tagger is a disambiguation and morphological analysis system which can perform various natural language processing tasks for the Arabic language such as tokenization, part-of-speech tagging, phrase chunking and other tasks [161]. According to Pasha and others [161], Madamira blends and improves some of the best services that the previously two used systems, MADA [103], [104], [106] and AMIRA [81], provide. The system was trained using the first three parts of the Penn Arabic Treebank, ATC, as shown in Table 2.6. It supports both XML and plain text as input and output file type, and an online demo [161] of Madamira is made available at [162]. The Madamira tagset consists of 32 tags. There are several steps in Madamira's pre-processing of the text. First, it transliterates the text using the Buckwalter transliterator [65]. Then, it utilises the SAMA and CALIMA Analysers to morphologically analyse the text. Next, it creates SVM language models. Then, Madamira uses the morphological features to tokenise the text. The final step performs the phrase chunking and named entity recognition of the text by utilising SVM models [93].

System name	Approach used	Trained by	Accuracy
The Madamira tagger	-	Penn Arabic Treebank (ATB)	95.9%
The Stanford Arabic tagger	Statistical	Penn Arabic Treebank (ATB)	96.49%
Farasa system by Abdelali	Statistical	-	97.43% for MSA and 84.44% for CA
APT Arabic tagger	Hybrid	-	86%
Al Shamsi and Guessoum	Statistical	-	97%
Darwish tagger	-	Arabic dialects tagged tweets	89.3%

Table 2.6. A table summary of different Arabic POS taggers.

The Stanford Arabic tagger was developed by Toutanova and Manning at Stanford University [200]. It is an open-source, multi-language, Java-based tagger that utilises a maximum entropy modelling technique, which according to Green, Marneffe and Manning [101] can achieve a tagging accuracy of 95.49% [93]. The Stanford tagger was also trained to tag other languages such as German, Spanish, French and Chinese and provides a command-line interface and an API. The first three parts of the Arabic Penn Treebank were used to train the Stanford Arabic tagger [199].

Abdelali and others [6] have developed the Farasa segmenter for the Arabic language. The tool provides various tasks such as segmentation, POS tagging, Arabic text diacritisation, tokenisation and dependency parser. The developer used an SVM-rank approach that utilises linear kernels. For evaluation, the developer created a unique test set made of 70 WikiNews articles which include a diversity of themes published between 2013 and 2014. According to the developer, the tagger achieved an accuracy of 97.43% for tagging MSA text and 84.44% for tagging CA text.

In 2002, the APT Arabic tagger was developed by Khoja [125], [127]. The tagger uses the hybrid approach with a tagset that is based on the BNC English tagset and consists of 131 tags. According to the author, the tagger reached an accuracy of 86%. Mohamed and Kübler [150] have developed an Arabic POS tagger that utilises two approaches, the first requires no

segmentation of the word and the second applies the basic POS word segmentation. According to Mohamed and Kübler, the first approach achieved an accuracy of 93.93% and the second approach achieved an accuracy of 93.41%. Al Shamsi and Guessoum [180] used a statistical method which employs HMMs to train an Arabic POS tagger. The tagger, which utilises Buckwalter's stemmer, as illustrated in Figure 2.16, and uses a tagset that includes 55 tags, achieved an accuracy of 97%. Darwish and others [76] have developed a POS tagger that tags four different Arabic dialects, which are Gulf, Maghrebi, Egyptian and Levantine. The tagger, which was trained by a new dataset that contains Arabic tagged tweets, has achieved an accuracy of 89.3%.

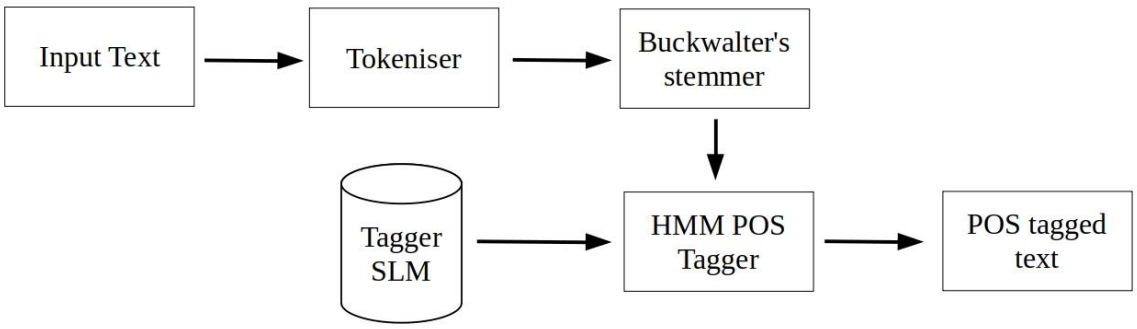


Figure 2.16. Al Shamsi and Guessoum HMM POS Tagger architecture [180].

As stated, Arabic is a morphologically complex language that causes various difficulties for Natural Language Processing (NLP) [74], [161], [104], [38]. Diacritics are used in the Arabic language to disambiguate terms. The presence of the four diacritics in the text help in the lexical disambiguation of the word, as some words share identical component letters but different diacritics. Modern Standard Arabic text, which most of Arabic NLP tasks are designed for [84], is very commonly written without diacritics and the contextual information is used by the reader of the text to disambiguate the meaning of the term. As a result of the ambiguity problem, the use of the Rule-based approach to tag the text increases the number of unanalyzed and mistagged terms [109]. The statistical method of tagging the Arabic text is broadly utilised to solve the POS uncertainty of the Arabic text [180]. PPM is a statistical language model algorithm that was applied in several Arabic NLP tasks and the adoption of the algorithm in the Arabic POS tagging may increase the efficiency and reduce the ambiguity problem.

2.3 Summary and Discussion

This chapter investigated the use of the PPM compression system for several Arabic NLP tasks, such as Arabic text compression and Arabic text classification.

A survey of Arabic text compression was given with a focus on PPM text compression models. First, an overview of compressing the Arabic text was presented. Then a survey of the three text compression methods using the PPM algorithm was introduced, ending with a focus on the limitations of the previous experiments conducted to estimate the performance of the tag-based method.

Next, this chapter introduced a survey of Arabic text classification. Then, a focus on the use of minimum cross-entropy as a text classifier was presented. Also, current applications of classifying Arabic text using the PPM algorithms was introduced with a focus on the need for classifying the two types of Arabic text, MSA and CA.

This chapter then presented an overview of language resources, for the Arabic language in particular. First, we reviewed language resources in general, followed by an overview of the annotated corpora for the Arabic language and the need to fill various gaps in annotated corpora.

Finally, this chapter has also reviewed the field of POS tagging of Arabic text. First an overview of the topic was provided along with the three main approaches used for tagging text. Then, the statistical POS tagging approach was reviewed followed by a survey of some existing statistical Arabic POS taggers and their tagsets.

The next chapter explores the approach of compressing Arabic text using parts-of-speech (tags) along with the text to give significantly better compression results when compared to current variations of PPM, word-based and character-based.

CHAPTER 3

Tag based Models for Arabic Text Compression

Contents:

3.1 Introduction

The previous chapter presented an overview of the Arabic language and surveyed the literature associated with this study by investigating the application of the PPM compression system to several Arabic NLP tasks. This chapter examines the use of tag-based compression of larger Arabic resources to re-evaluate the performance of tag-based compression which may reveal POS linguistic aspects of the Arabic language (as per research questions 1 and 2). The best text compression algorithms can be applied to natural language processing tasks often with state-of-the-art results [196], [193], [195], [197], [15]. Therefore, improved tag-based compression has applications beyond the specific compression application. For example, compression of co-translated parallel text produces compressed text of similar sizes which leads to a more effective method for sentence alignment of parallel corpora [31].

The focus of this specific chapter is on compressing Arabic text (encoded using the UTF-8 encoding scheme). The Arabic language poses many challenges for the NLP community due to interesting linguistic features that the language has, such as complicated morphology, dialect varieties and frequent code-switching [80] but to date, most of these unique Arabic NLP tasks have not been satisfactorily addressed. PPM can be successfully applied to many of these challenges in other languages such as English, Chinese and Welsh (see [193], [195], [196], [197], for example) but have yet to have been applied comprehensively to Arabic. One stumbling block is the need for more effective text compression algorithms for Arabic that can be applied to different Arabic NLP.

The rest of the chapter is formed as follows. Section 3.2 mentions details about experiments on using PPM tag-based modelling to compress Arabic text. Section 3.3 discusses the results and limitations of those experiments. A summary and discussion are presented in section 3.4.

A portion of this chapter has been published in a conference paper (Alkhazi, I. S., Alghamdi, M. A., & Teahan, W. J. (2017, September). “Tag based models for Arabic text compression”. In 2017 Intelligent Systems Conference (IntelliSys) (pp. 697-705). IEEE.)

3.2 Tag Based Compression Experimental Setup

The quality of the compression results depends on the quality and correctness of the tagging process. Currently, the resources for manually tagged Arabic corpus are limited, and the existing manually tagged corpora are small [26] as also shown in Table 3.1. Also, the size of the text being compressed affects the effectiveness of the compression, as will be mentioned later. So, to get a sufficient amount of tagged text to make the method effective, existing Arabic taggers, the Madamira Arabic POS tagger [161] and the Stanford Arabic POS tagger [100] were utilised. Since the two taggers were trained by the Arabic Treebank corpus [141], therefore, the corpora which have been used in these experiments were mostly written in MSA, which will reduce the amount of mistagged terms.

This chapter used five different corpora. The first corpus is Corpus A [31]. It covers several subjects such as politics, opinions, legal issues, economics, conferences, business, cinema and books. The text in the corpus was gathered from the Al-Hayat website, a bilingual newspaper, and from the open-source online corpus, OPUS [30].

Data Set	Corpus Size	Character encode size	Tag encode size	Improvement
AFP	138,223	23,512	32,844	-28.4%
UMH	426,811	64,420	70,824	-9.0%
XIN	158,997	25,974	29,189	-11.0%
ALH	600,091	108,196	120,928	-10.5%
ANN	195,043	39,196	56,765	-31.0%
XIA	431,474	71,183	77,261	-7.9%

Table 3.1. A sample of the tag-based compression results for the Prague Arabic Dependency Treebank.

The second corpus used is the Bangor Arabic Compression Corpus (BACC) that contains 31-millon words which was collected from several sources such as magazines, websites and books. The BACC consists of 16 files divided based on genre and size [26]. The third source is the King Saud University Corpus of Classical Arabic (KSUCCA) which is a relatively large corpus containing over 50 million words, divided into six genres. The main goal for the creation

of this corpus is analyzing the lexical meaning of the Holy Quran [40]. In this chapter, we used the parts that were mostly written in Modern Standard Arabic such as the Science sub-corpus.

The fourth source is the Arabic in Business and Management Corpora (ABMC). El-Haj [89], [122] created several Arabic corpora such as EASC, KALIMAT, Arabic Dialects Dataset and ABMC. The corpora are articles obtained from WikiNews, newspapers and summaries of the articles, and is mostly written in MSA text. The fifth and last source is the Arabic Learner Corpus [21], which includes 282,732 terms, gathered from students of Arabic in Saudi Arabia. The corpus covers spoken and written text created by 942 pupils, from 67 various nationalities enrolled at pre-university and university levels.

There are two steps to perform the tag-based compression experiments. First, you must tag each word in the text using, for example, the Madamira tagger or the Stanford tagger. Second, using the tag-based model, the tags should be encoded with the text itself. Even with the extra contextual information, a tag for each term, which has been added to the text, the hope is that the tag-based compression outperforms both the word-based and character-based compression of the original text. As stated, the experiments were done using Corpus A [31], BACC corpus [26], the King Saud University Corpus of Classical Arabic (KSUCCA) [40], the ABMC corpus [89], [122] and the Arabic Learner Corpus [21]. The text was tagged using Madamira [161] and Stanford [101] Arabic taggers. The PPM modelling was done using the Tawa toolkit [192].

The processing steps were as follow:

- The text being compressed is first preprocessed to produce a tagger input file in XML for the Madamira tagger and raw text for Stanford tagger.

- The text is then segmented and tagged using the Madamira tagger and the Stanford tagger.

- Then, the segmented and tagged text from the output of both taggers, where every prefix/infix/suffix is tagged, is processed for the first part of the experiment.

- For the second part of the experiment, the tag for only the root, as shown in Figure 3.1 for the Madamira tagger output, is selected as a tag for the entire term (not split by prefix/infix/suffix).

```
<word id="25" word="الدولة">
        <svm_prediction>
                <morph_feature_set diac="الدَّوْلَةِ" lemma="دَوْلَة dawolap" pos="noun" />
        </svm_prediction>
                <analysis rank="0" score="0.8840385512557991">
                        <morph_feature_set diac="الدَّوْلَةُ" lemma="دَوْلَة_١"
                         gloss="state/country" pos="noun" stem="دَوْل"/>
                </analysis>
        <tokenized scheme="ATB">
                 <tok id="0" form0="الدولة"/>
        </tokenized>
        <tokenized scheme="MyD3">
                   <tok id="0" form0="ال+"/>
                   <tok id="1" form0="دولة"/>
        </tokenized>
</word>
```

Figure 3.1. The Madamira segmentation and tagging output for the term "الدولة" which translates to "the country".

- Three compressed files are generated using tools provided by the Tawa toolkit [192]. The original text is first extracted from the tagged file, where every tag is removed and only the original text remain, then compressed using the order 5 PPMD character-based model but with the Arabic characters as defined by the UTF-8 encoding first converted to equivalent symbol numbers as described in [26].

- The same text is then compressed using an order 1 word-based model.

- Finally, for the first part of the experiment, the two segmented and tagged text files, one for Madamira and the other is for Stanford tagger, are compressed using the Tawa toolkit [192] that uses the model described by equation (2.8). For the second part of the experiment, the two unsegmented and tagged text files are compressed using the same tool.

3.3 Compression Results

The experimental results for Corpus A can be described into two parts (as shown in tables 3.2, 3.3, 3.4 and 3.5) based on how words in the text were segmented when it was tagged. Each table lists the results for the three different types of text compression (character, word and tag-based). In tables 3.2 and 3.4, the name of the text file being compressed and its size in bytes is shown in the first two columns. The compression output size (also in bytes) is then listed in the next four columns – for character-based compression first, then word-based compression, then tag-based compression in two columns using the tags generated by the two taggers. In tables 3.3 and 3.5, the results are converted to a bits-per-character ("bpc") compression ratio by dividing the encode output size in bytes multiplied by 8 (to determine the number of bits) then dividing by the number of Arabic characters in the text being compressed.

	Corpus Size (bytes)	Character encode size (bytes)	Word encode size (bytes)	Madamira Tag encode size (bytes)	Stanford Tag encode size (bytes)
Books	10,111,728	992,932	1,768,654	869,342	900,598
Business	23,794,220	2,954,882	4,583,372	2,809,082	2,878,069
Cinema	48,405,798	7,693,241	10,413,103	7,599,585	7,786,930
Conferences	19,683,004	2,463,988	3,810,208	2,349,752	2,406,579
Crimes	9,200,719	1,341,031	1,891,607	1,356,279	1,386,562
Decisions	15,172,319	1,465,637	2,650,080	1,261,969	1,307,147
Economy	23,617,015	2,950,719	4,563,009	2,803,885	2,872,965
Geographies	14,788,494	1,859,902	2,855,610	1,795,240	1,840,775
Issues	9,775,694	1,248,016	1,882,561	1,217,472	1,248,534
Law	14,459,749	1,816,820	2,784,756	1,756,683	1,800,847
Politics	20,398,361	2,547,920	3,950,894	2,422,574	2,482,772
Reports	14,568,403	1,853,284	2,822,415	1,795,010	1,839,161
Stories	28,362,790	4,213,806	5,930,949	4,208,840	4,314,284

Table 3.2. The compression output sizes using unsegmented text for Corpus A.

The percentage improvement of the tag-based compressors compared to the previously best state-of-the-art PPMD character-based compressor (as shown in the tables) are shown in the final two columns in tables 3.3 and 3.5. The character and word-based results in tables 3.2, 3.3,

3.4 and 3.5 are the same since the text was processed in the same way – only the tag-based text, as described earlier, was processed differently (split by prefix/infix/suffix, as shown in figures 3.1 and 3.2, for tables 3.4 and 3.5, and not for tables 3.2 and 3.3). However, the character and word-based results have been duplicated in both tables to allow for ease of comparison. Note that the results for the word-based compression is noticeably worse than the results for both the character and tag-based methods which may be a reflection of the morphologically rich nature of Arabic text. This requires further investigation as results for other languages indicate that word-based compression is usually very competitive [196]. However, this investigation is beyond the scope of this chapter which was focused on tag-based compression.

	Character encode (bpc)	Word encode (bpc)	Madamira Tag encode (bpc)	Stanford Tag encode (bpc)	Madamira Improve. (%)	Stanford Improve. (%)
Books	0.79	1.40	**0.69**	0.71	14.22%	10.25%
Business	0.99	1.54	**0.94**	0.97	5.19%	2.67%
Cinema	1.27	1.72	**1.26**	1.29	1.23%	-1.20%
Conferences	1.00	1.55	**0.96**	0.98	4.86%	2.39%
Crimes	**1.17**	1.64	1.18	1.21	-1.12%	-3.28%
Decisions	0.77	1.40	**0.67**	0.69	16.14%	12.12%
Economy	1.00	1.55	**0.95**	0.97	5.24%	2.71%
Geographies	1.01	1.54	**0.97**	1.00	3.60%	1.04%
Issues	1.02	1.54	**1.00**	1.02	2.51%	-0.04%
Law	1.01	1.54	**0.97**	1.00	3.42%	0.89%
Politics	1.00	1.55	**0.95**	0.97	5.17%	2.62%
Reports	1.02	1.55	**0.99**	1.01	3.25%	0.77%
Stories	1.19	1.67	**1.19**	1.22	0.12%	-2.33%

Table 3.3. The compression ratios (in bpc) when compressing the unsegmented text for Corpus A.

Figure 3.2. Sample segmented verse of the Holy Quran which translates to "merciful among themselves, you see them bowing and prostrating".

	Corpus Size (bytes)	Character encode size (bytes)	Word encode size (bytes)	Madamira Tag encode size (bytes)	Stanford Tag encode size (bytes)
Books	10,111,728	992,932	1,768,654	915,434	916,565
Business	23,794,220	2,954,882	4,583,372	2,721,276	2,718,995
Cinema	48,405,798	7,693,241	10,413,103	7,318,652	7,402,514
Conferences	19,683,004	2,463,988	3,810,208	2,274,826	2,273,345
Crimes	9,200,719	1,341,031	1,891,607	1,268,526	1,272,853
Decisions	15,172,319	1,465,637	2,650,080	1,348,677	1,349,478
Economy	23,617,015	2,950,719	4,563,009	2,720,672	2,719,921
Geographies	14,788,494	1,859,902	2,855,610	1,721,999	1,721,422
Issues	9,775,694	1,248,016	1,882,561	1,162,636	1,163,376
Law	14,459,749	1,816,820	2,784,756	1,686,617	1,685,836
Politics	20,398,361	2,547,920	3,950,894	2,348,775	2,348,913
Reports	14,568,403	1,853,284	2,822,415	1,720,206	1,719,935
Stories	28,362,790	4,213,806	5,930,949	3,981,845	4,004,927

Table 3.4. The compression output sizes using segmented text for Corpus A.

The words for part one of the experiment (i.e. whose results are shown in tables 3.2 and 3.3) were not segmented (by prefix, infix and/or postfix), and the tag of the term's root is assigned to the entire term. For example, the term 'وإبراهيم' might have been assigned the tag 'prop_noun' for the entire term. Whereas in part two of the experiment, the term 'وإبراهيم' was segmented into the prefix 'و', which is a conjunction, and the term 'إبراهيم', which is a 'prop_noun'. Another example of text segmentation is the term 'لعنصرها' which has the tag 'noun' assigned to it in the first part. But in part two of the experiment, this term was segmented into three individual parts:

prefix, which is 'ل' with the tag 'prep' assigned to it, the term 'عنصر' with the tag 'noun' assigned to it and a suffix, which is 'ها' that has the tag 'pron_dem' assigned to it. Also, in both parts of the experiment, we used the default tagset adopted by the Madamira and Stanford Arabic taggers.

	Character encode (bpc)	Word encode (bpc)	Madamira Tag encode (bpc)	Stanford Tag encode (bpc)	Madamira Improve. (%)	Stanford Improve. (%)
Books	0.79	1.40	**0.72**	0.73	8.5%	8.3%
Business	0.99	1.54	0.91	**0.91**	8.6%	8.7%
Cinema	1.27	1.72	**1.21**	1.22	5.1%	3.9%
Conferences	1.00	1.55	0.92	**0.92**	8.3%	8.4%
Crimes	1.17	1.64	**1.10**	1.11	5.7%	5.4%
Decisions	0.77	1.40	**0.71**	0.71	8.7%	8.6%
Economy	1.00	1.55	0.92	**0.92**	8.5%	8.5%
Geographies	1.01	1.54	0.93	**0.93**	8.0%	8.0%
Issues	1.02	1.54	**0.95**	0.95	7.3%	7.3%
Law	1.01	1.54	0.93	**0.93**	7.7%	7.8%
Politics	1.00	1.55	**0.92**	0.92	8.5%	8.5%
Reports	1.02	1.55	0.94	**0.94**	7.7%	7.8%
Stories	1.19	1.67	**1.12**	1.13	5.8%	5.2%

Table 3.5. The compression ratios (in bpc) when compressing the segmented text for Corpus A.

Tables 3.2 and 3.3 show that compressing the text which is not segmented and tagged with the Madamira Arabic tagger improves the compression of the original text over the character-based compressor by an average of 4.9%, whereas using the Stanford tagger improves the compression by an average of 2.2%. As stated, the tagging in this stage represents the whole term (with postfixes and prefixes).

The second part of the results are shown in tables 3.4 and 3.5 which represents the improvement of the compression as a result of segmenting the words in the text, with each postfix and prefix being tagged and compressed as a separate term. By segmenting each word

into prefixes, infixes and postfixes, and then tagging each as a term, the compression results were improved significantly. For the text that was tagged by Madamira, the compression improved by 7.6% compared to the character-based compression. As for the text that was tagged by the Stanford tagger, the compression improved by 7.4%.

The compression of the segmented text outperformed the same unsegmented text by 2.7%, for the text that was tagged by Madamira and 5.2% improvment for the text that was tagged by the Stanford tagger. This improvement of the compression reflects that the segmentation of the text is an essential step in the tagging process as stated by AlGahtani and McNaught [23], "Finding the correct tagging requires the correct segmentation in advance." Essentially, the correct tagging of the text made a better prediction of the upcoming term, and this has led to better compression as a result.

The same experiment was performed on the BACC corpus. Using PPM tag-based compression to compress the largest sub-corpus, *Book_collection*, the compression was improved by 2.5% using the Madamira tagger and 2.4% using the Stanford tagger. This corpus mainly consists of religious books which are mostly written in classical Arabic. As stated before, both the Madamira and Stanford Arabic taggers were trained on the first three parts of the Arabic Treebank Corpus. This proves that different NLP applications should treat these texts separately by accurately classifying CA from MSA within the text.

	Corpus Size	Character encode size	Word encode size	Madamira Tag encode size	Stanford Tag encode size
Religion	140,112,368	19,057,454	28,380,731	19,189,175	19,209,861
Literature	73,892,199	12,828,194	16,462,334	12,348,603	12,406,021
Linguistics	64,085,357	10,877,334	14,138,294	10,713,179	10,768,099
Science	59,038,146	9,547,210	12,831,102	9,397,481	9,428,971

Table 3.6. The compression output sizes for the KSUCCA Corpus.

The KSUCCA corpus [40] is divided into many genres and most of those genres are written in classical Arabic, the largest of which is the *Religion* sub-corpus, which contains the Holy Quran and other ancient Islamic books. Tables 3.6 and 3.7 illustrate the results of the compression

experiments using the sub-corpora *Religion*, *Literature*, *Linguistics* and *Science*. The use of classical Arabic text in the ancient Islamic books which was included in the *Religion* sub-corpus decreased the compression rate compared to the second, third and fourth sub-corpora, and this may reflect the fact that these sub-corpora consist of relatively more recent books.

	Character encode bpc	Madamira Tag encode bpc	Stanford Tag encode bpc	Madamira Improve. (%)	Stanford Improve. (%)
Religion	**1.09**	1.10	1.10	-0.7%	-0.79%
Literature	1.39	**1.34**	1.34	3.9%	3.4%
Linguistics	1.36	**1.34**	1.34	1.5%	1.1%
Science	1.29	**1.27**	1.28	1.6%	1.3%

Table 3.7. The compression ratios (in bpc) when compressing the KSUCCA Corpus.

ABMC is a relatively small corpora that consists mostly of MSA text. The Arabic Learner Corpus is also a small corpus written mostly in CA text. Tables 3.8 and 3.9 show the tag-based compression of ABMC, and tables 3.10 and 3.11 show the results of the tag-based compression of the Arabic Learner Corpus.

Genre	Corpus Size	Char encode size	Madamira Tag encode size	Stanford Tag encode size
Economic News	2,201,462	305,360	**286,718**	289,712
Management	1,358,576	201,192	**197,154**	198,241
Stock News	1,070,320	89,391	**83,173**	83,640

Table 3.8. The compression output sizes for the ABMC Corpus.

The results in tables 3.8 and 3.9 indicate that the tag-based compression of ABMC outperforms the character-based compression by an average of 4.77% using Madamira tagged text and 3.90% using Stanford tagged text. The compression results for the Arabic Learner Corpus, as presented in tables 3.10 and 3.11, show that the character-based compression of the corpus surpasses the tag-based compression. The fact that the Arabic Learner Corpus is written in CA text may have caused a decrease in the quality of the tag-based compression.

Genre	Char_encode bpc	Madamira encode bpc	Stanford encode bpc	Madamira Improvement	Stanford Improvement
Economic News	1.11	**1.04**	1.05	6.50%	5.40%
Management	1.18	**1.16**	1.17	2.05%	1.49%
Stock News	0.67	**0.62**	0.63	7.48%	6.88%

Table 3.9. The compression ratios (in bpc) when compressing the ABMC corpus.

Genre	Corpus Size	Char encode size	Madamira Tag encode size	Stanford Tag encode size
Arabic Learner Corpus	2806467	**469502**	472541	477,506

Table 3.10. The compression output sizes for the Arabic Learner Corpus.

Genre	Char_encode bpc	Madamira encode bpc	Stanford encode bpc	Madamira Improvement	Stanford Improvement
Arabic Learner Corpus	**1.34**	1.35	1.36	-0.64%	-1.68%

Table 3.11. The compression ratios (in bpc) when compressing the Arabic Learner Corpus.

According to the authors of the ATC, the corpus was written using Modern Standard Arabic. Therefore, the quality of the tagging of classical Arabic will be effected compared to when Modern Standard Arabic is being tagged. Table 3.12 shows a sample of classical Arabic tagged by the Madamira tagger, where many of the tags being assigned are not correct. The sample was taken from the BACC sub-corpus, '*Arabic history*', where the text is written in classical Arabic. The compression results of both the BACC sub-corpora, '*Book collection', 'Arabic book1', 'Arabic book2', 'Arabic book3',* and '*Arabic history*', are shown in tables 3.13 and 3.14.

Term	Tag	Correct Tag
الفروسية	Adj	Noun
ماركوز	Verb	Proper Noun
شرانقا	Verb	Noun
فواصل	Verb	Noun

Table 3.12. Sample of miss-tagged words.

A random sample from a tagged classical Arabic text shows that a number of terms were miss-tagged. An example is term "الفروسية" that translates to “the sport of horse riding”, which was tagged as "adj" whereas the term should be a "noun". Other examples are "ماركوز" that translates to “Marcos”', "شرانقا" that translates to “a cover” and "فواصل" that translates to “commas”, which were all tagged as ‘verb’ when the right tag is "noun" as shown in Table 3.12.

Genre	Corpus Size	Character encode size	Word encode size	Madamira Tag encode size	Stanford Tag encode size
Book collection	197,935,882	30,959,688	42,477,508	**30,191,397**	30,235,460
Arabic History	30,251,137	**4,206,076**	5,937,257	4,267,257	4,286,946
Press	536,692	**100,879**	117,440	102,749	104,344
Arabic book1	829,036	**164,445**	187,353	170,881	173,793
Arabic book2	884,273	**176,896**	200,961	183,935	186,466
Arabic book3	977,286	**190,482**	219,284	199,225	202,239

Table 3.13. The compression output sizes for the BACC corpus.

Another limitation of tag-based compression is the size of the text file being compressed. Tag-based compression of smaller texts is less effective as shown in tables 3.13 and 3.14, where the result on the two smaller files (*Arabic History* and *Press*) is not as good as the character-based compression for these files. This is because PPM is an online adaptive system that needs relatively large amounts of training data to learn and build the tag-based models.

Genre	Character encode bpc	Madamira Tag encode bpc	Stanford Tag encode bpc	Madamira Improve. (%)	Stanford Improve. (%)
Book collection	1.25	**1.22**	1.22	2.5%	2.4%
Arabic History	**1.11**	1.13	1.13	-1.4%	-1.9%
Press	**1.50**	1.53	1.56	-1.8%	-3.3%
Arabic book1	**1.59**	1.65	1.68	-3.77%	-5.38%
Arabic book2	**1.60**	1.66	1.69	-3.83%	-5.13%
Arabic book3	**1.56**	1.63	1.66	-4.39%	-5.81%

Table 3.14. The compression ratios (in bpc) when compressing the BACC corpus.

Data Set	Original Data Provider	Related Corpora	Corpus Size	Character encode size	Tag encode size
AFP	Agence France Presse	Penn ATB Part 1	138,223	23,512	32,844
UMH	Ummah Press Service	Penn ATB Part 2	426,811	64,420	70,824
XIN	Xinhua News Agency	Arabic Gigaword	158,997	25,974	29,189
ALH	Al Hayat News Agency	Arabic Gigaword	600,091	108,196	120,928
ANN	An Nahar News Agency	Arabic Gigaword	195,043	39,196	56,765
XIA	Xinhua News Agency	Arabic Gigaword	431,474	71,183	77,261

Table 3.15. The compression output sizes the Prague Arabic Dependency Treebank.

Further results for compressing small sized texts are shown in tables 3.15 and 3.16. Currently, the resources for manually tagged Arabic corpora are limited. The Prague Arabic dependency treebank [110] is a collection of manually tagged Arabic text. It consists of the first part of the Penn ATB and four other parts of Arabic Gigaword, all of which are made available at the Arabic UD treebank website [160], [141], [1], [110]. The less effective compression results for these texts (where the tags being used by the compressor are manually edited rather than

automatically assigned by a tagger) illustrates one limitation that PPM tag-based compression has when compressing small sized texts. The Madamira and Stanford taggers were not used to tag this corpus for this reason.

Data Set	Character encode (bpc)	Tag encode (bpc)	Improvement
AFP	**1.36**	1.90	-28.4%
UMH	**1.21**	1.33	-9.0%
XIN	**1.31**	1.47	-11.0%
ALH	**1.44**	1.61	-10.5%
ANN	**1.61**	2.33	-31.0%
XIA	**1.32**	1.43	-7.9%

Table 3.16. The compression ratios (in bpc) when compressing the Prague Arabic Dependency Treebank.

Previous experiments [26], which were performed to compare the three models on Arabic text, produced similar results to those shown in tables 3.15 and 3.16. According to Alhawiti [26], the only used resource was the first part of the Arabic Treebank Corpus (ATC), and as stated before, PPM is an online adaptive text compression system that needs relatively large amounts of training data to learn and build the tag-based models. Therefore, using PPM tag-based model to compress text will produce less effective results when compressing such a small corpus. Tables 3.17, 3.18 and Figure 3.3 illustrate the relation between the size of the corpus and the compression ratio.

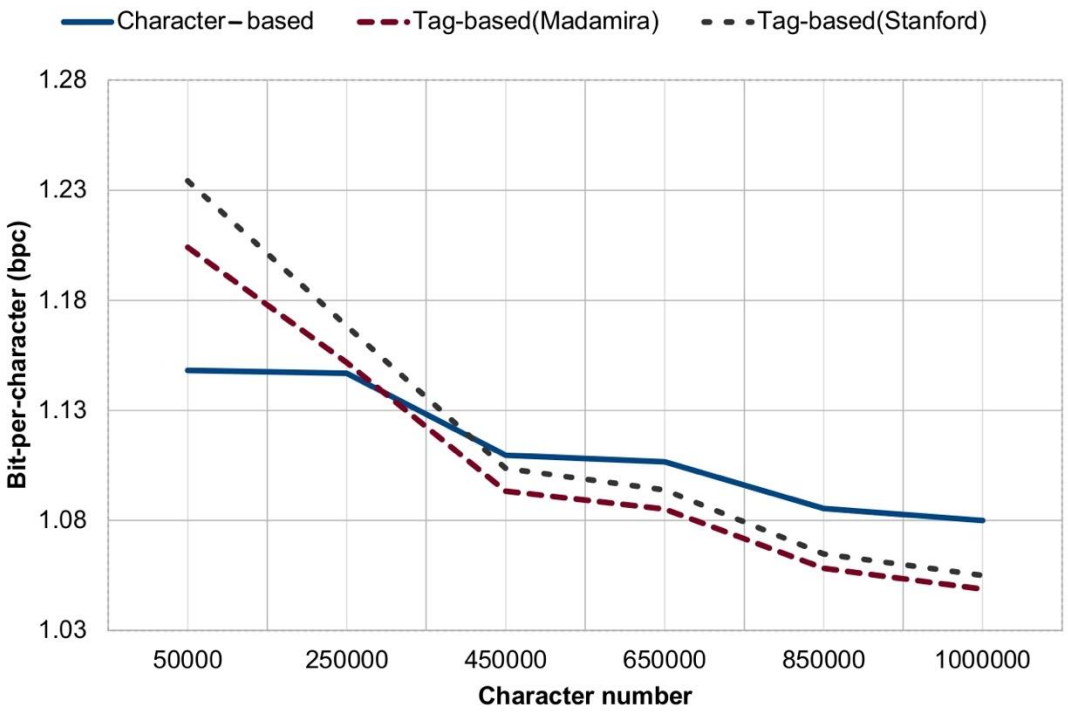


Figure 3.3. Relation between PPM compression and corpus size.

Characters Number	Corpus Size	Char encode size	Madamira Tag encode size	Stanford Tag encode size
50,000	92,562	13,284	13,931	14,280
250,000	469,968	67,373	67,657	68,636
450,000	829,449	115,045	113,369	114,426
650,000	1,197,432	165,646	162,437	163,721
850,000	1,566,402	212,522	207,201	208,484
1,000,000	1,842,044	248,670	241,493	242,929

Table 3.17. The compression output sizes for different PPM models and different corpus size.

Characters Number	Characters-based bpc	Tag-based bpc (M)	Tag-based bpc (S)	Madamira Improve. (%)	Stanford Improve. (%)
50,000	**1.15**	1.20	1.23	-4.64%	-6.97%
250,000	**1.15**	1.15	1.17	-0.42%	-1.84%
450,000	1.11	**1.09**	1.10	1.48%	0.54%
650,000	1.11	**1.09**	1.09	1.98%	1.18%
850,000	1.09	**1.06**	1.06	2.57%	1.94%
1,000,000	1.08	**1.05**	1.06	2.97%	2.36%

Table 3.18. The compression ratios (in bpc) for different PPM models and different corpus size.

Since PPM tag-based model uses two streams, a tag stream and a word stream to build the model, compressing the text using the character-based model will take less time compared to the tag-based model since the latter requires the text to be tagged during preprocessing. For example, Table 3.19 compares the average time, in seconds, required to compress five corpora. First, the file has to be tagged using the Madamira or Stanford Arabic tagger. Second, the tagged file is processed and the tags are extracted. Finally, the formatted file is compressed using the Tawa toolkit [192]. In contrast, to compress an Arabic corpus using a character-based model, first, the Arabic letters are converted to unsigned integers. Then, the resulting file is passed to PPM for compression.

3.4 Summary and Discussion

Tag-based compression of Arabic text based on the Prediction-by-Partial Matching (PPM) text compression scheme was investigated and compared with character-based and word-based methods. The tag-based method requires first tagging the text being compressed, and then transmitting both the words in the text along with their tags. The results of compressing tagged and untagged texts show that using tag-based compression significantly outperforms both the word-based and character-based models, and the added extra-tag information improves overall compression compared to the untagged compressed text.

Corpus name	Character-based Compression (Seconds)	Tag-based Compression Time (Seconds)
Books	7.7	49.9
Business	22.6	131.5
Conferences	18.6	107.2
Crimes	8.7	51.5
Issues	8.1	56.5

Table 3.19. Corpus A compression time when using the PPM character-based and tag-based compression.

Two taggers for tagging the text were investigated – Madamira and the Stanford tagger. Using segmented text which was tagged by Madamira, the compression was improved by 7.6% as opposed to an 7.4% improvement when the Stanford tagger was used when compared to the state-of-the-art PPM character-based model. Future improvements can be made by improving the quality of the tagging process. The results also indicate that there is a difference in quality between tagging Classical Arabic and Modern Standard Arabic. One way of addressing this is to investigate whether it is possible to distinguish the two types of the Arabic language (MSA and CA). The PPM tag-based compression technique also provides an interesting way for evaluating the performance of different Arabic taggers and for helping to investigate the linguistic validity of the tagsets.

The next chapter will investigate the use of minimum cross-entropy as a text classifier to classify and segment the two types of Arabic text to overcome the problem of code-switching in Arabic text and improve the tag-based compression of the Arabic text, and help improve for other Arabic NLP tasks that are designed for specific types of Arabic text such as Arabic POS tagging.

CHAPTER 4

Classifying and Segmenting Classical and Modern Standard Arabic using Minimum Cross-Entropy

Contents

4.1 Introduction

The previous chapter investigated the use of the tag-based compression of various Arabic resources to re-evaluate the performance of tag-based compression. The results from the last chapter indicate that some Arabic corpora, such as the Bangor Arabic Compression Corpus (BACC), is a mixture of both CA and MSA text. An example is the BACC sub-corpus '*Arabic book1'*, which includes both recent novels with ancient Arabic poems. (See Figure 2.9 for one example). The results of utilising such a corpus in order to perform various NLP tasks will vary and will not be consistent and reliable, as demonstrated in the previous chapter. Consequently, NLP applications should treat these texts separately and use different training data for each or process them differently, and therefore this provides the main motivation for this chapter. This chapter investigates whether it is possible to distinguish the two types of the Arabic language (MSA and CA) using PPM (as per research question 4).

The work in this chapter uses an approach based on the Prediction-by-Partial Matching (PPM) compression scheme (order 5 PPMD in particular), as the basis of both text classification and segmentation. This Markov-based approach effectively uses character-based language models and has been employed in many NLP tasks in the past often with state-of-the-art results or results competitive with traditional schemes [196], [193], [195], [197], [15], [194].

The chapter reports the experiments that were performed as part of the evaluation of the PPM classifier and segmenter when applied to Arabic text. Four experiments were conducted: (A) initial classification experiments in section 4.2; (B) classification of published Arabic MSA and CA corpora in section 4.3; (C) segmentation of the same Arabic corpora in section 4.4; and (D) tagged-based compression experiments of Arabic text in section 4.5. A summary and discussion are presented in section 4.6.

The first experiment uses 200 files for the initial evaluation process. The second experiment examines the result of classifying a number of published Arabic MSA and CA using minimum cross-entropy as described in section 2.2.3. The third experiment conducts classification of each separate line for the same Arabic corpora used in section 4.3.1 to find out whether different Arabic corpora have a mixture of CA and MSA text. The fourth experiment performs text segmentation on a text file with a mixture of CA and MSA sentences that were gathered

randomly from the testing files used in section 4.2. The fifth experiment conducts text segmentation to investigate whether different Arabic corpora have a mixture of CA and MSA text by examining the results of segmenting the same Arabic corpora used in section 4.3.1. All experiments in section 4.4 use a Viterbi-based algorithm that finds the most probable sequence of segmented characters. Lastly, section 4.5 utilises the results of tag-based compression obtained in Chapter 3 to examine the correlation between the quality of the compression with the classification results from the previous section.

The published Arabic MSA and CA corpora which were used in this chapter are the Bangor Arabic Compression Corpus (BACC) [26], the Universal Dependencies (UD) project corpus [1], the Arabic in Business and Management corpus (ABMC) [88] and the Arabic Learner Corpus [22]. The Universal Dependencies corpus (UD), which according to the authors, is an MSA corpus containing mainly newswire. The corpus is based on other Arabic sources such as the Prague Arabic Dependency Treebank (PADT) [78] and the Penn Arabic Treebank (PATB) [79]. The second corpus is the ABMC corpus. According to the El-Haj [88], the Arabic in Business and Management Corpora is obtained from WikiNews, newspapers and summaries of the articles, and is mostly written in MSA text. The Arabic Learner Corpus is a small corpus written mostly in CA text as stated before. The final corpus utilised in this chapter is the BACC corpus. According to Alhawiti [26], BACC corpus comprises 14 genres that contain CA text such as *Arabic book1*, *Arabic history* and *Arabic literature*, and MSA text such as *Education*, *Political* and *Press*.

Segmenting Arabic text increases the performance of some NLP applications such as parts-of-speech tagging. As stated before, most Arabic NLP tasks are trained and built for MSA. The performance of such a task drops when applied to Classical text [42], [38]. The motive of this chapter is to classify and segment CA and MSA using the PPM character-based compression algorithm to overcome the problem of code-switching in Arabic text and improve the performance of NLP tasks that are designed for specific type of Arabic text. The experiments in this chapter used two language models, one for CA and another for MSA. Published Arabic corpora that contain mostly the required type of Arabic text were used to train the two static models. The MSA model was trained using Corpus A [31]. The second model used in this chapter was trained using CA text from parts of the King Saud University Corpus of Classical Arabic (KSUCCA) [40]. As stated, the corpus is relatively large and it contains over 50 million

words, split into six genres such as Literature, Linguistics and Science. To generate a relatively similar size set of training text as the first model (as this helps improve classification accuracy), the sub-genre Religion was not included in the training process. To obtain a more robust evaluation and ensure the training text used for the models was separate from the testing text, a tenfold cross-validation technique was used for the classification experiments.

Both the PPM language modelling and the segmentation were performed using the Tawa toolkit [192]. This toolkit allows *static* models to be created from training text. That is, once the models have been created, they can be used to prime the model(s) used by the application and are subsequently not altered during the compression, classification or segmentation processes.

A portion of this chapter has been published in a journal paper (Alkhazi, I. S., & Teahan, W. J. (2017). Classifying and Segmenting Classical and Modern Standard Arabic using Minimum Cross-Entropy. *International Journal of Advanced Computer Science and Applications*, 8(4), 421-430.)

4.2 Initial Classification Experiments

This initial experiment was conducted to evaluate the PPM classifier in order to perform an initial experiment with some sample test files to find out how well a PPM classifier would perform at distinguishing between MSA and CA text. The testing files were divided into two groups, each with 100 files. The first group comprised 100 files that contained CA text randomly gathered from the Holy Quran, Islamic books such as Ibn Qayyim and Ahmad ibn Hanbal and poems from the famous Arab poet, Al-Mutanabbi. The second group comprised 100 files containing MSA text randomly collected from popular Arabic news websites such Aljazeera.net [28], BBC Arabic [51] and skynewsarabia [184] and recently published novels.

Four evaluation criteria (Accuracy, Recall, Precision and F-measure) were used to evaluate the classification results using the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.4}$$

where TP is the true positives which are the number of cases where the prediction matches the type of Arabic text and TN is the true negatives which represent the number cases where the prediction does not match the type of Arabic text, and FP and FN are the false positives and false negatives respectively, as shown for the confusion matrix in Table 4.1.

	Predicted CA	**Predicted MSA**
Actual CA	*TN*	*FP*
Actual MSA	*FN*	*TP*

Table 4.1. How true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are defined for a confusion matrix.

Classifying the Classical and MSA text using the PPM compression algorithm obtained an accuracy of 95.5%, an average precision of 0.958, an average recall of 0.955 and an average F-measure of 0.954. The results in Table 4.2 show that the PPM classifier predicted all the 100 files that contain CA text and 91 out of 100 files which have MSA text.

	Predicted CA	**Predicted MSA**
Actual CA	100	0
Actual MSA	9	91

Table 4.2. PPM classification results.

4.3 Classifying Arabic Corpora

The previous section showed that on the sample of 200 files, the PPM classifier performed well at distinguishing between MSA and CA text. Section 4.3.1 performs a document level classification of published Arabic MSA and CA corpora to find out how well a PPM classifier would perform at differentiating between MSA and CA corpora. Section 4.3.2 performs classification of each separate line in the corpora using PPM, to find out whether different Arabic corpora have a mixture of CA and MSA text.

4.3.1 Document Level Text Classification

The experiment described in this section examined how well a PPM classifier would perform at differentiating between MSA and CA corpora on a document level. Table 4.3 displays the results of this experiment for the UD corpus, Table 4.4 for the ABMC, Table 4.5 for the Arabic Learner Corpus and Table 4.6 for the BACC corpus. The tables list the size of the text files, the size of the compressed output files (in bytes), the compression ratios (in bits per character or 'bpc') and the type (CA or MSA) predicted from the model with the best compression (as per the classification procedure in section 2.2.3 using equation 2.13).

The steps of the experiment are as follows:

- Using the two static models created earlier for priming, two compressed files are generated by compressing the Arabic texts using an order 5 PPMD character-based compression scheme.

- Then, the cross-entropy i.e. the size of the two compressed files, are compared and the class label of the text, in this case CA or MSA, is chosen from the file with the smallest compressed size.

Genre	Corpus Size	Classical model Compression (bytes)	Modern model Compression (bytes)	Classical bpc	Modern bpc	Predicted Type
AFP	138,223	35,788	33,149	2.07	**1.92**	MSA
UMH	426,811	106,478	97,517	2.00	**1.83**	MSA
XIN	158,997	40,660	36,709	2.05	**1.85**	MSA
ALH	108,599	27,419	25,536	2.02	**1.88**	MSA
ANN	130,068	32,847	31,227	2.02	**1.92**	MSA
XIA	293,104	74,650	67,550	2.04	**1.84**	MSA

Table 4.3. Classification results of the UD corpus.

Genre	Corpus Size	Classical model Compression (bytes)	Modern model Compression (bytes)	Classical bpc	Modern bpc	Predicted Type
Economic News	2,201,462	544,181	496,183	1.98	**1.80**	MSA
Management	1,358,576	317,477	275,826	1.87	**1.62**	MSA
Stock News	890,493	224,493	199,571	2.02	**1.79**	MSA

Table 4.4. Classification results of the ABMC corpus.

Genre	Corpus Size	Classical model Compression (bytes)	Modern model Compression (bytes)	Classical bpc	Modern bpc	Predicted Type
Arabic Learner Corpus	2,806,467	620,563	630,306	**1.77**	1.80	CA

Table 4.5. Classification results of the Arabic Learner Corpus.

Genre	Corpus Size	Classical model Compression (bytes)	Modern model Compression (bytes)	Classical bpc	Modern bpc	Predicted Type
Arabic book1	829,036	187,362	192,804	**1.81**	1.86	CA
Arabic book2	884,273	202,343	206,271	**1.83**	1.87	CA
Arabic book3	977,286	223,451	229,887	**1.83**	1.88	CA
Arabic history	30,251,137	5,750,445	7,838,286	**1.52**	2.07	CA
Arabic literature	18,594,383	3,846,029	4,877,075	**1.65**	2.10	CA
Arabic poems	46,929	11,701	12,313	**1.99**	2.10	CA
Art and music	41,770	9,665	9,137	1.85	**1.75**	MSA
articles	101,641	22,982	21,630	1.81	**1.70**	MSA
Book collection	197,935,882	40,631,602	48,551,255	**1.64**	1.96	CA
culture	34,188	7,867	7,363	1.84	**1.72**	MSA
Economic	15,352	3,583	3,066	1.87	**1.60**	MSA
Education	26,418	6,078	5,504	1.84	**1.67**	MSA
Political	46,884	10,995	9,785	1.88	**1.67**	MSA
Press	536,692	122,961	111,260	1.83	**1.66**	MSA
Sports	31,059	7,225	6,659	1.86	**1.72**	MSA
Stories	1,022,476	242,699	237,372	1.90	**1.86**	MSA

Table 4.6. Classification results of the BACC.

The classification results in tables 4.3, 4.4, 4.5 and 4.6 show that the PPM classifier predicted the correct type of text for all corpora. The classification results in this experiment reflect the dominant type of text in the corpus, and the small difference in compression sizes between the CA and MSA models suggests that the corpus may contain a mixed text of both CA and MSA text, such as the Arabic Learner Corpus in Table 4.5.

4.3.2 Line Level Classification

Classifying corpora of unknown origins, or for which it may be suspected of having a mixture of CA and MSA text, will help Arabic NLP researchers to confirm their content. The previous section showed that PPM classifier performed well at distinguishing between MSA and CA corpora. The results also reveal that some corpora, such as the Arabic Learner Corpus, may contain a mixed text of both CA and MSA. This section will investigate whether performing a classification of each separate line in the same Arabic corpora used in section 4.3.1 detects the code-switching found in some Arabic corpora. Table 4.7 presents the results of this experiment for the UD corpus, Table 4.8 for the ABMC corpus, Table 4.9 for the Arabic Learner Corpus and Table 4.10 for the BACC corpus. In the first column of the tables, the total number of lines in the corpus is listed, the total number of predicted CA and MSA lines and the percentage of each class.

Data set	Number of lines	Classical lines	Modern lines	Classical %	Modern %
AFP	11,375	4,693	6,682	41.26%	58.74%
UMH	35,056	14,036	21,020	40.04%	59.96%
XIN	12,524	4,844	7,680	38.68%	61.32%
ALH	9,134	3,653	5,481	39.99%	60.01%
ANN	11,377	4,832	6,545	42.47%	57.53%
XIA	23,983	9,452	14,531	39.41%	60.59%

Table 4.7. Line level classification results of UD corpus.

Genre	Number of lines	Classical lines	Modern lines	Classical %	Modern %
Economic News	176,399	60,026	116,373	34.03%	65.97%
Management	5,442	456	4,986	8.38%	91.62%
Stock News	403	24	379	5.96%	94.04%

Table 4.8. Line level classification results of the ABMC corpus.

Genre	Number of lines	Classical lines	Modern lines	Classical %	Modern %
Arabic Learner Corpus	8,959	3,581	5,378	39.97%	60.03%

Table 4.9. Line level classification results of the Arabic Learner Corpus.

Genre	Number of lines	Classical lines	Modern lines	Classical %	Modern %
Arabic book1	1,859	1,342	517	72.19%	27.81%
Arabic book2	2,613	1,685	928	64.49%	35.51%
Arabic book3	997	786	211	78.84%	21.16%
Arabic history	91,719	67,251	24,468	73.32%	26.68%
Arabic literature	69,618	67,635	1,983	97.15%	2.85%
Arabic poems	1,259	934	325	74.19%	25.81%
Art and music	172	18	154	10.47%	89.53%
Articles	196	41	155	20.92%	79.08%
Book collection	540,026	484,221	55,805	89.67%	10.33%
Culture	74	23	51	31.08%	68.92%
Economic	38	1	37	2.63%	97.37%
Education	119	5	114	4.20%	95.80%
Political	130	6	124	4.62%	95.38%
Press	1,090	104	986	9.54%	90.46%
Sports	81	11	70	13.58%	86.42%
Stories	9,306	2,396	6,910	25.75%	74.25%

Table 4.10. Line level classification results of BACC.

The steps of the experiment are as follows:

- Using the two static models created earlier for priming, each line of the corpus is compressed using an order 5 PPMD character-based compression scheme.

- Then, the cross-entropies i.e. the sizes of the compressed line are compared and the class label of the text, in this case CA or MSA, is chosen from the result with the smallest compressed size, then the number of lines for each class is counted.

The classification results from these tables confirm the results of the previous section which indicate that some Arabic corpora contain different types of Arabic text, such as the Arabic Learner Corpus in Table 4.9. Examining the Arabic Learner Corpus reveals that the number of terms in each line is uneven which will reduce the accuracy of classification by not reflecting the true ratio of each type of text in a document, as the classification result of a text line containing one term is equivalent, using this method of classification, to the classification result of a text line with many terms. This raises the need for a method to segment the types of text within the text which will reflect an accurate picture of the textual contents.

4.4 Segmenting Mixed Arabic Corpora

The experiment in this section was conducted to find out how well a PPM classifier would perform at segmenting MSA and CA text within the text. Two experiments were performed. Section 4.4.1 shows the results of the first experiments where PPM segmented a mixed text file. Section 4.4.2 investigates whether PPM can be used to segment different Arabic corpora that have a mixture of CA and MSA text. Both experiments performed a text segmentation using a Viterbi-based algorithm that finds the most probable sequence of characters of each class, category or topic in the text [195] where all possible segmentations (as defined by switching between encoding models) are considered.

4.4.1 Segmenting Mixed Arabic Text

The experiment in this section was conducted to evaluate the PPM segmentation performance. A text file with a mixture of CA and MSA text that was gathered randomly from the testing files used in Section 4.2. The text contained 100 sentences, 50 of which are written in MSA and 50 sentences contained CA text, distributed randomly. The Tawa toolkit [192] was used to segment the text file at the character level to insert labels (tags), either CA or MSA, inside the text. The segmentation in this step was applied using a Viterbi-based algorithm [195]. The output file is then processed and the segmented CA and MSA sentences are then examined.

	Predicted CA	**Predicted MSA**
Actual CA	47	3
Actual MSA	11	39

Table 4.11. PPM segmentation results.

Segmenting the CA and MSA text using the PPM compression algorithm obtained an accuracy of 86%, an average precision of 0.869, an average recall of 0.86 and an average F-measure of 0.859. The results in Table 4.11 reveal that the PPM segmented 47 out 50 CA sentences and 39 out of 50 MSA sentences correctly. A sample of the segmented text is shown in Figure 4.1. The text contains three Arabic sentences in one line; the first sentence is a news feed obtained from Aljazeera website (MSA text), and the other two are from Hadith books (CA text).

Figure 4.1. Segmenting MSA, the first three lines, and CA text, the last seven lines, using PPM.

4.4.1 Investigating Mixed Arabic Corpora

The previous section showed that PPM performed well at segmenting MSA and CA sentences within the text. Therefore, we use this result in this section to find out whether different Arabic corpora have a mixture of CA and MSA text by investigating the results of segmenting the same Arabic corpora used in Section 4.3.1.

This experiment was conducted as follows:

- The Tawa toolkit [192] was used to segment the text file at the character level to insert labels (tags), either CA or MSA, inside the text. The segmentation in this step was applied using a Viterbi-based algorithm [195].

- Then, a post-processing of the resulting file was performed to count all the terms of each label.

Table 4.12 displays the outcomes of this experiment for the UD corpus, Table 4.13 for the ABMC, Table 4.14 for the Arabic Learner Corpus and Table 4.15 for the BACC corpus. The

tables list the numbers of words in the segmented files for both CA and MSA texts and the percentages of each.

Data set	Number of words	Number of Classical words	Number of Modern words	Classical (CA)%	Modern (MSA) %
AFP	11,369	594	10,775	5.22%	94.78%
UMH	34,765	2,053	32,712	5.91%	94.09%
XIN	12,666	554	12,112	4.37%	95.63%
ALH	9,019	1,078	7,941	11.95%	88.05%
ANN	11,152	2,252	8,900	20.19%	79.81%
XIA	23,930	617	23,313	2.58%	97.42%

Table 4.12. Segmentation results of the UD corpus.

Genre	Number of words	Number of Classical words	Number of Modern words	Classical (CA) %	Modern (MSA) %
Economic News	169,374	12,200	157,174	7.20%	92.80%
Management	121,603	7,192	114,411	5.91%	94.09%
Stock News	87,943	53	87,890	0.06%	99.94%

Table 4.13. Segmentation results of the ABMC corpus.

Genre	Number of words	Number of Classical words	Number of Modern words	Classical (CA) %	Modern (MSA) %
Arabic Learner Corpus	287,107	161,897	125,210	56.39%	43.61%

Table 4.14. Segmentation results of the Arabic Learner Corpus.

The results from tables 4.12, 4.13, 4.14 and 4.15 indicate that some Arabic corpora contain mixed CA and MSA text, and the PPM compression models can be used to produce an accurate estimate of the extent of both Arabic text types. The illustration of the segmentation process is shown in Figure 4.2 which shows randomly selected segmented samples from two

of BACC sub-genre, '*Arabic literature'* and '*Arabic book1'*. The sample demonstrates typical output of the segmentation process which produces an accurate picture of the textual contents.

<Classic> به إلا ضربا من الحلم أو الكابوس حدقت به فاضت عيناها واصفرت وجنتاها ارتجفت أصابعها<\Classic>
<Modern> أنت تريد بناء دولة خارج الزمان والمكان وتترك فلسطين لماذا لا تقدم هذا الاقتراح إلى إدارة السجن<\Modern>
<Classic> الحسنة وادفع الشبهات الغليظة النتنة والأفكار الباطلة وأخرج الناس من ظلمات الجهلة إلى نور النيرة <\Classic>
<Modern> كنت مع زملائي سفرت إلى الرياض للدراسة عندما وصلت إلى المطار رأيت أجانب من البلاد المختلفة <\Modern>

Figure 4.2. Random segmented samples from the BACC.

4.5 Tag-based Compression Experiments

Most Arabic language NLP systems are made for processing MSA [84]. Since most popular recognised Arabic POS taggers were trained on MSA text [141], the tagging of mixed corpora text will vary in quality and will not be consistent and reliable. This section utilises the results of tag-based compression obtained in Chapter 3 to examine the correlation between the quality of the compression with the classification results from the previous section.

Table 4.16 lists some the of tag-based compression results obtained in Chapter 3 with the classification results from previous sections. It shows in the second column the percentage improvement in compression for the tag-based compression scheme over the character-based compression scheme, and the type of text (CA or MSA) in the third column that was confirmed in the earlier experiments. A positive percentage improvement indicates the tag-based compression was better, and a negative improvement indicates the character-based compression was better.

The results in Table 4.16 show that utilising the tags to compress the BACC sub-corpus '*Arabic literature'*, which was found to consist of 99.74% Classical Arabic text, decreases the compression by 4.38% (compared with the character-based compression scheme). However, using the same compression model to compress the ABMC sub-corpus '*Economic News'*, which was found to consist of 92.80% MSA text, increases the compression by 6.50% (compared with the character-based compression scheme). The difference in compression quality provides an indication that the quality of tagging for the CA text has dropped, compared to the quality of tagging for the MSA text, because the compression size has increased.

Genre	Number of words	Number of Classical words	Number of Modern words	Classical (CA)%	Modern (MSA) %
Arabic book1	85,441	65,867	19,574	77.09%	22.91%
Arabic book2	89,015	61,645	27,370	69.25%	30.75%
Arabic book3	104,055	83,503	20,552	80.25%	19.75%
Arabic history	3,350,365	3,348,513	1,852	99.94%	0.06%
Arabic literature	1,983,790	1,978,670	5,120	99.74%	0.26%
Arabic poems	4,701	4,151	550	88.30%	11.70%
Art and music	3,985	528	3,457	13.25%	86.75%
Articles	9,624	1,792	7,832	18.62%	81.38%
Book collection	20,725,720	19,836,491	889,229	95.71%	4.29%
Culture	3,107	476	2,631	15.32%	84.68%
Economic	1,376	3	1,373	0.22%	99.78%
Education	2,437	33	2,404	1.35%	98.65%
Political	4,317	62	4,255	1.44%	98.56%
Press	50,977	4,351	46,626	8.54%	91.46%
Sports	2,875	221	2,654	7.69%	92.31%
Stories	111,809	28,664	83,145	25.64%	74.36%

Table 4.15. Segmentation results of the BACC.

Corpus	Tag-based Compression Improvement	Text Type
BACC – Arabic history	-1.4%	CA
BACC – Arabic literature	-4.38%	CA
ABMC – Economic News	6.50%	MSA
ABMC – Stock News	7.48%	MSA

Table 4.16. Tag-based Compression on CA and MSA Text.

4.6 Summary and Discussion

Classification of Classical Arabic (CA) and Modern Standard Arabic (MSA) text was performed on sample texts using a PPM character-based compression scheme achieving an accuracy of 95.5%, an average precision of 0.958, an average recall of 0.955 and an average F-measure of 0.954. Further classification experiments were conducted in this study to analyse mixed Arabic corpora. A line-level classification of Arabic corpora was performed and the results showed that different sub-genres of some Arabic corpora contain a mixture of CA and MSA. The fourth experiment performed text segmentation on a text file with a mixture of CA and MSA sentences that were gathered randomly from the testing files used in section 4.2. Segmenting the CA and MSA text using the PPM compression algorithm obtained an accuracy of 86%, an average precision of 0.869, an average recall of 0.86 and an average F-measure of 0.859. Further segmentation experiments were conducted to investigate whether different Arabic corpora have a mixture of CA and MSA text by examining the results of segmenting different Arabic corpora. The results from the last segmentation experiment confirmed the results obtained in section 4.3.2, which showed that different Arabic corpora have a mixture of CA and MSA text. Lastly, section 4.5 utilised the results of tag-based compression that were reported in Chapter 3 to examine the correlation between the quality of the compression with the classification results from the previous sections. The results in section 4.5 provides an indication that the quality of the tagging is affected when either CA and MSA text is being tagged, as confirmed in [42], [41], [38], therefore showing that NLP applications (such as taggers) should treat these texts separately and use different training data for each or process them differently.

The next chapter will describe the creation of the new Bangor Arabic Annotated Corpus (BAAC) which is a Modern Standard Arabic (MSA) corpus that comprises 50K words manually annotated by parts-of-speech.

CHAPTER 5

BAAC: Bangor Arabic Annotated Corpus

Contents

5.1 Introduction

The previous chapter explored the approach of classifying Arabic text using PPM. This chapter describes the creation of the new Bangor Arabic Annotated Corpus (BAAC) which is a Modern Standard Arabic (MSA) corpus that comprises 50K words manually annotated by parts-of-speech.

POS annotated corpora are essential for the development of many NLP systems, such as part-of-speech tagging [180], statistical modelling [111]. The lack of such resources limits some researchers from progressing further in their efforts. The limited availability of some existing annotated corpora and the cost of acquiring others are one of the main reasons that contribute to resource scarcity. Several efforts have been made to overcome the lack of resources [37], [9], [85].

There exist some Arabic language resources that cannot be utilised by many researchers. Alhawiti [26] stated that availability, and cost issues, were significant issues such as for the Arabic Treebank corpus [141]. Other resources are designed to be used for particular research or annotated using a distinctive tagset produced for an explicit purpose. The Qur'anic Arabic Dependency Treebank is one example where the text is written in CA text and the corpus uses a tagset which is designed to tag CA text using traditional Arabic grammar [85], [30]. This need for annotated corpora, which are necessary for the development of many NLP systems, provided the motivation to create a manually annotated corpus for the Arabic language for this study (as per research question 5).

Another goal is to provide a new resource required by many kinds of research, such as the ongoing tag-based text compression research in chapter 3, where the only annotation required at this stage is POS tags. The tagset used to annotate the new corpus is the same as used by the Madamira Arabic tagger, for reasons that will be discussed in the annotation tagset section (section 5.3). Since the Madamira Arabic POS tagger is trained by the Arabic Treebank corpus [141], and that corpus is written in MSA, the newly annotated corpus must also be written in MSA.

The chapter is organised as follows. Section 5.2 presents the sources used to created the newly created annotated corpus. The next section, section 5.3, interduces the tagset used in the annotation process. Section 5.4 describes the automatic POS tagging of the selected text. The newly developed annotation tool was presented in section 5.5. The following section, section 5.6, describes the data preparation stage of the annotation process. The new annotated corpus is evaluated in section 5.7. Section 5.8 presents the new corpus statistics and section 5.9 is the summary and discussion of this chapter.

A portion of this chapter has been published in a journal paper (Alkhazi, Ibrahim S., and William J. Teahan. "BAAC: Bangor Arabic Annotated Corpus." *International Journal of Advanced Computer Science and Applications*.11 (2018): 131-140.)

5.2 The Data Source

The data source for the new corpus is the Press sub-corpus from the BACC corpus [26]. The BACC corpus was created originally to test the performance of various text compression algorithms on different text files. The results of the text classification performed in the previous chapter reveal that the Press sub-corpus is 91% written in MSA, as shown in Figure 5.1. According to the authors, the sub-corpus is a newswire text consisting of 51K terms, gathered from various news websites between 2010 and 2012 and covers many topics such as political and technology news.

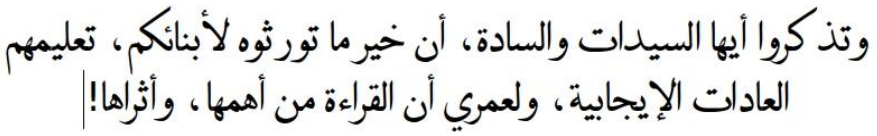

وتذكروا أيها السيدات والسادة، أن خير ما تورثوه لأبنائكم، تعليمهم
العادات الإيجابية، ولعمري أن القراءة من أهمها، وأثراها!

Figure 5.1. A Social News, that promotes reading, from the Press sub-corpus [26] in MSA text.

5.3 The Annotation Tagset

The tagset used in the BAAC corpus is the same as used by the Madamira tagger [161], which was used initially by the MADA tagger [104]. The tagset is the subset of the English tagset which

was presented with the English Penn Treebank and consists of 32 tags and was initially proposed by Diab, Hacioglu and Jurafsky [80]. The experiments conducted in chapter 3 have concluded that the quality of tag-based compression varies from one tagset to another. The different tagsets, some of which are shown in Table 5.1, were used to compress MSA text using POS tags, and tag-based compression using the Madamira tagset outperforms other tagsets such as Stanford [101] and Farasa [6]. Since one of the main goals of creating a gold-standard POS annotated text is to investigate the effect of manual annotation on the tag-based text compression, as described below in the experiments, therefore, the Madamira tagset, which outperformed other tagsets and consists of only 32 tags that are shown in Table 5.2, is used to annotate the BAAC POS tag and to create the ground-truth data which will be used later for training and evaluation purposes.

Term	Madamira Tag	Stanford Tag	Farasa Tag
الادارة	noun	DTNN	NOUN-FS
ترحب	noun_prop	VBP	E/ES/SV
بالتزام	verb	NN	NOUN-MS
الامين	noun	DTNN	NOUN-MS
العام	noun	DTJJ	ADJ-MS
بزيادة	adj	NN	NOUN-FS
عنصر	noun	NN	NOUN-MS
الميزانية	noun	DTNN	NOUN-FS
العادية	noun	DTJJ	ADJ-FS
لمكتب	noun	NN	NOUN-MS
الامم	noun	DTNN	NOUN-MP
المتحدة	noun	DTJJ	ADJ-MP

Table 5.1. Different Arabic Tagsets.

Tag	Agreements	Disagreements	Observed Agreement %
noun	23570	529	97.80
verb	5714	44	99.24
prep	5574	10	99.82
adj	4632	1235	78.95
noun_prop	2272	520	81.38
conj_sub	1534	17	98.90
conj	1148	79	93.56
pron_rel	992	37	96.40
pron_dem	767	11	98.59
noun_quant	574	1	99.83
part_neg	498	2	99.60
pron	367	6	98.39
adv	166	195	45.98
adj_comp	265	15	94.64
noun_num	252	7	97.30
part_verb	221	0	100.00
verb_pseudo	203	0	100.00
adj_num	156	26	85.71
adv_interrog	25	111	18.38
adv_rel	83	3	96.51
abbrev	60	2	96.77
part_restrict	59	16	78.67
part	25	27	48.08
pron_interrog	19	30	38.78
part_focus	14	9	60.87
part_interrog	22	0	100.00
part_fut	12	0	100.00
part_voc	10	0	100.00
part_det	8	2	80.00
interj	2	0	100.00
Total	**49244**	**2934**	**94.38%**

Table 5.2. The number of agreements and disagreements of different tags between the two annotators in reverse frequency order.

5.4 Automatic POS Tagging

Madamira [161] was utilised to automatically tag the corpus by POS. The manual annotation process of the BAAC corpus followed annotation guidelines proposed by Maamouri [142] for annotating POS tags. All the previous corrections that are made to a tag are shown to the annotators during the process of annotation, as illustrated in section 5.6, and the Madamira tagset used to annotate this corpus applies the criteria proposed by Maamouri.

5.5 The Annotation Tool

Most existing tools, such as the TrEd tool [159], [165] which was used in the annotation of The Prague Dependency Treebank, are developed to annotate Treebank types of corpora, such as dependency trees corpora, that contain other information about each term, such as the gloss or a comment from an annotator, as shown in Figure 2.11. As mentioned earlier, the first stage of the BAAC annotation process will only add the POS tags to the corpus. Other linguistic information, such as the structural annotation, will be adapted in future work, therefore, the tool which will be used to manually annotate this corpus will only annotate POS tags.

During the preparation for the annotation process, many constraints arose and defined four requirements that had to be met by the annotation tool. First, as the annotators are native Arabic speakers, a detailed Arabic translation of the tagset was provided with examples during the annotation process. Second, the software used for the annotation had to comply with the hardware and software requirements of the computer used to perform the annotation. Thirdly, the annotation tool, as shown in Figure 5.2, had to be executed on different operating systems, therefore, the tool was designed to be portable. Finally, online backing up procedures with the ID of the annotators was done to ensure the safety of the data.

The previous requirements were met by developing a new annotation tool. First, a detailed Arabic translation of the tagset, which was obtained from Alrabiah [42] and then examined by Arabic specialists, was coded in the annotation tool as shown in figure 5.2. The annotation tool also offers examples of the tag if required by the annotator as will be explained in the following paragraph. To comply with the hardware requirements and reduce memory dependency, the tool loads only one sentence to be modified at a time. To follow the Maamouri [142] annotation

guidelines, the tool also displays the history of annotation by showing two types of modifications, the original tag assigned by the Madamira tagger and any tag chosen by previous annotators, if they exist. A current status of the annotation process is also displayed to the annotator, such as the number of annotated tags in the current session and the number of modified tags in the total document, as explained in the following paragraph. The Java programming language was used to develop the annotation tool, and therefore, the tool can be executed on different operating systems. The tool also provided online backing up procedures each time the annotator modified a tag to eliminate any data loss.

The first information given to the annotator is the number of the current sentence out of the total, as labelled in Figure 5.2 by number 1. Clicking on the button labelled as 2 in the figure opens a file dialogue that enables the annotator to edit an external file and not the default annotated text file saved in the home directory. Clicking on the button marked as 3 opens a file dialogue that saves a backup file. The text area identified as 5 in the figure displays the current sentence which the annotator is currently editing. The term, which is coloured in black, is followed by a tag, which is between the brackets. Every tag is displayed with a distinctive colour, for example, all the verb tags, or "فعل", in the figure are displayed in red, and all nouns, or "اسم", is coloured in blue. The font size of the text area identified as 5 can be changed by clicking on the (+) and (-) buttons labelled as 4. At the bottom of the text area, where label 6 is, a log of all the changes made by the annotator is displayed. The log shows the term, the original tag and the updated one. The annotator displays the previous sentence by click on the button labelled as 7, and the button labelled as 8 displays the next sentence. The progress bar labelled as 9 displays the amount of progress made by the annotator.

The text area labelled as 10 shows more statistics about the work, such as the number of terms in the current sentence with the number of modified tags are shown in the first line, the total number of terms in the entire annotated file with the number of sentences presented in the second line and location of the annotated file shown in the line before the last. If the annotator wants to modify a tag, the checkbox labelled as 14 needs to be clicked first. Then, by clicking on the dropdown menu labelled as 11, all the terms with their current tags will be displayed to the annotator. After selecting the term for modification, the annotator will select the new tag from the second dropdown menu which will display a list of all the tags with their translation and an example tag will be shown in a message if the annotator clicks on the text labelled as 15. Saving

the changes made to the sentence is done by clicking on the button at the bottom left of the figure, where the label 13 is.

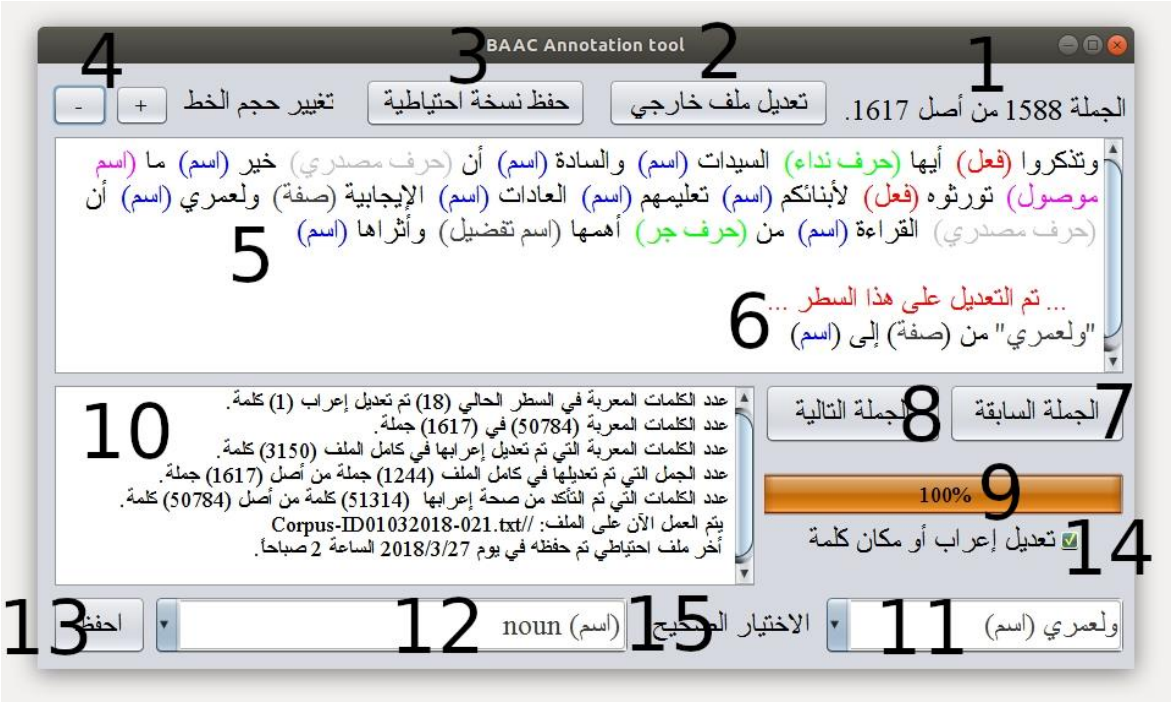


Figure 5.2. The Annotation tool.

5.6 Data Preparation

After using Madamira [161] to automatically POS tag the BAAC corpus, a copy of the tagged corpus was given to each annotator. Each copy was split into batches of documents that have 10-20 sentences and the ID of the annotator was coded with each batch to be used later in the evaluation section. The two annotators, who are native Arabic speakers and postgraduate students in Arabic Studies, started working to manually annotate the corpus on a full-time basis in two stages.

In the first stage of the annotation process, the annotators were required to work on-site to resolve any issues with the annotation tool and the annotation of the corpus was completed using the facilities provided by Tabuk Public Library. When the annotation process was finished, the two versions were evaluated and the Inter Annotator Agreement was calculated using two metrics, as will be discussed below in the BAAC evaluation section (section 5.7). The differences between the two versions were examined and adjusted off-site by a third annotator,

who is a native Arabic speaker and PhD candidate student in Arabic Studies, to produce a final version of the corpus. The total time needed to annotate the corpus was two months – three weeks for the first stage and the rest for the final stage.

5.7 BAAC Evaluation

The quality of the annotated corpus affects the quality of the NLP application that utilises it. For instance, Reidsma and Carletta [168] has illustrated that the errors produced by machine learning tools are the same errors made by the annotators of the corpus that was used for training those tools. Two metrics were used to evaluate the quality of the BAAC, the Kappa coefficient [73] to calculate the inter-annotator agreement (IAA) among the two annotators and a direct percent agreement for each tag [145]. Using the data in tables 5.3 and 5.4, the obtained Kappa value is 0.956, which is recognised as perfect according to Landis and Koch [133]. The total observed agreement from Table 5.2, which displays the number of agreements and disagreements of different tags between the two annotators in a reverse frequency order, is 94.38%. Taking the number of tag occurrences into consideration, Table 5.2 shows that the tag verb or 'فعل' has the highest agreement between the annotators with 99.24% agreement. It also shows that the annotators agreed only 25 times out of 136 (18%) on the tag 'adv_interrog' or 'حال'. Also, the annotators agreed 45.98% of the time for the tag 'adv', and 38.78% of the time for the tag 'pron_interrog'.

The reasons for such variation between the annotators were:

- The different understanding of the tag and, in some cases, its subset of tags by the annotators. For example, tables 5.3 and 5.4 show that the two annotators disagreed concerning the tag 'noun' and the tag 'adj' in many instances. The different understanding of the tag 'adv_interrog' and the tag 'adj' has also caused a noticeable number of disagreements between the two annotators.

	abbrev	adj	adj_comp	adj_num	adv	adv_interrog	adv_rel	conj	conj_sub	interj	noun	noun_num	noun_prop	noun_quant	part
abbrev	59								1						
adj		4363	1	1							247		22		
adj_comp		7	260	1							4		2		
adj_num				142								12	2		
adv		92		12	108		1	6	67		63		3	1	
adv_interrog		106		6		9		1	3		7				
adv_rel		8					74								
conj								1148							
conj_sub								52	1455						
interj										2					
noun	1	1151		4	42		4	5	18		22762	2	98		1
noun_num		3		15							5	235	1		
noun_prop		166	4	1	1			1	1		450	3	2121		
noun_quant		1									2			573	
part					8			2	1		1		1		23
part_det		1									1				
part_focus								9							
part_fut															
part_interrog															
part_neg		1						1							
part_restrict															
part_verb															
part_voc											1				
prep		18						1			6		1		
pron		4									1				1
pron_dem		1			7			1					1		
pron_interrog						16	7		1						
pron_rel													1		
verb		18							4		20		19		
verb_pseudo															
Total	60	5940	265	182	166	25	86	1227	1551	2	23570	252	2272	574	25

Table 5.3. The BACC Agreement Table Part 1.

	part_det	part_focus	part_fut	part_interrog	part_neg	part_restrict	part_verb	part_voc	prep	pron	pron_dem	pron_interrog	pron_rel	verb	verb_pseudo	Total
abbrev									2							62
adj									1					4		4639
adj_comp														6		280
adj_num																156
adv									8							361
adv_interrog														4		136
adv_rel												1				83
conj																1148
conj_sub													32			1539
interj																2
noun								1					4	41		24134
noun_num																259
noun_prop									11					36		2795
noun_quant																576
part						16										52
part_det	8															10
part_focus		14														23
part_fut			12													12
part_interrog				22												22
part_neg					498											500
part_restrict						58			1							59
part_verb							221									221
part_voc								9								10
prep						1			5556	1				2		5586
pron										366				1		373
pron_dem											767			1		778
pron_interrog									2			18	5			49
pron_rel									3				988			992
verb														5663		5724
verb_pseudo															203	203
Total	8	14	12	22	498	75	221	10	5584	367	767	19	1029	5758	203	

Table 5.4. The BACC Agreement Table Part 2.

- Human error in the annotation process contributed to some of the errors in the annotated corpus. This was confirmed by random samples taken to be re-annotated by the same annotator.

The previous reasons were taken into consideration, and all the disagreements were highlighted, which was then given to the third annotator who went through all the disagreements and modified them based on his judgment. Finally, a final version of the corpus, which contains the agreements from the first two annotators and the agreements of the third one, was produced and used for further applications, as illustrated in the experiments section.

Tag	Frequency	%
noun	24099	47.52
verb	5714	11.27
prep	5574	10.99
adj	4632	9.13
noun_prop	2792	5.51
conj_sub	1534	3.02
conj	1148	2.26
pron_rel	992	1.96
pron_dem	778	1.53
noun_quant	575	1.13

Table 5.5. The ten most frequent tags by the first annotator.

5.8 Corpus Statistics

As stated, the text of the BAAC corpus was obtained from the sub-corpus `Press` of the BACC. The first annotator made 3150 changes to the originally tagged corpus and the second made 2959 modifications. Table 5.5 and Table 5.6 list the first ten most frequent tags for the annotators. The most frequent tag is 'noun' representing 47.52% for the first annotator and 46.48% for the second. The least used tag is 'noun_quant' being 1.13% of the tags for both annotators. A noticeable difference between the two annotators is the use of the tag 'adj' which

occurred 4632 times (9.13%) for the first annotator and occurring 1235 more times for the second annotator (11.57%).

Tag	Frequency	%
noun	23570	46.48
adj	5867	11.57
verb	5758	11.35
prep	5584	11.01
noun_prop	2272	4.48
conj_sub	1551	3.06
conj	1227	2.42
pron_rel	1029	2.03
pron_dem	767	1.51
noun_quant	574	1.13

Table 5.6. The ten most frequent tags by the second annotator.

Tables 5.7, 5.8 and 5.9 show the ten most frequently used terms in the BAAC. The first and second most frequent words in the BAAC are 'في' which is a 'prep', which translates as 'in', and 'من', which is also a 'prep', which translates as 'from' representing 2.83% and 2.65% of the text respectively, as shown in Table 5.7. Table 5.8 shows that the most commonly used bigram is 'من خلال', which translates as 'through' occurring 37 times in the corpus. Since the `Press` sub-corpus, which is the source of the BAAC, was gathered between 2010 and 2012 from several Arabic news websites, the most commonly used trigrams in the BAAC, as shown in Table 5.9, are 'في ميدان التحرير' which translates as 'In Tahrir Square', and 'الأعلى للقوات المسلحة' which translates as 'Higher Council of the Armed Forces', which were mentioned 12 times, and both trigrams relate to the events that happened in Egypt during the same period.

Rank	Word	Freq	%
1	في	1437	2.83
2	من	1345	2.65
3	و	735	1.45
4	أن	698	1.38
5	على	615	1.21
6	إلى	401	0.79
7	التي	352	0.69
8	عن	351	0.69
9	أو	275	0.54
10	لا	245	0.48

Table 5.7. Word unigrams statistics from the BAAC.

Rank	Bigram	Freq	%
1	من خلال	37	0.07
2	إلى أن	37	0.07
3	الولايات المتحدة	34	0.07
4	ميدان التحرير	30	0.06
5	في مصر	28	0.05
6	عدد من	28	0.05
7	من قبل	26	0.05
8	ثورة يناير	26	0.05
9	بعد أن	26	0.05
10	أن يكون	25	0.05

Table 5.8. Word bigrams statistics from the BAAC.

Figure 5.3 plots using log scales the ranked tag, bi-tag and tri-tag sequences versus their frequencies in the BAAC. There are 32 unique tags used in the annotated corpus, as mentioned earlier. The corpus also has 433 unique bi-tags where the sequence 'noun noun' dominates most of the bi-tags sequences. Finally, there are 2,113 distinct tri-tags used in the BAAC. The figure shows a Zipf's Law-like behaviour which mirrors the behaviour of a similar plot for the

English language [190]. More details about the BAAC n-tag sequences are found in Table 5.10, Table 5.11 and Table 5.12, and will be discussed below.

Rank	Trigram	Freq	%
1	في ميدان التحرير	12	0.02
2	الأعلى للقوات المسلحة	12	0.02
3	المجلس الأعلى للقوات	11	0.02
4	القانون رقم لسنة	10	0.02
5	غفر الله له	9	0.02
6	قال أبو عبدالله	8	0.02
7	عبدالله غفر الله	8	0.02
8	اللجنة الوطنية للاستقدام	8	0.02
9	الكسب غير المشروع	8	0.02
10	أبو عبدالله غفر	8	0.02

Table 5.9. Word trigrams statistics from the BAAC.

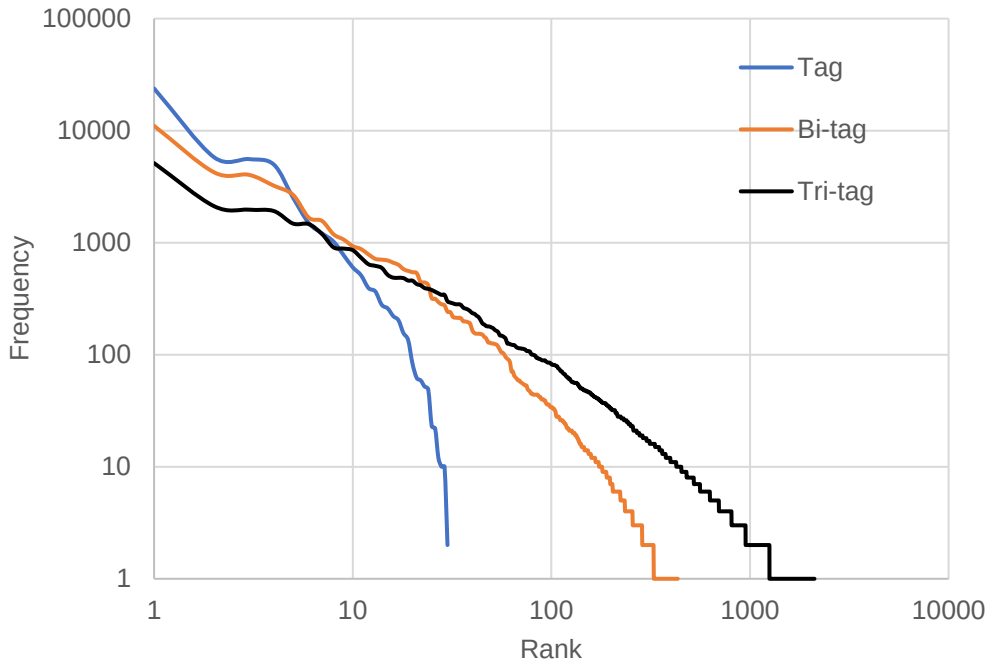


Figure 5.3. Rank versus Tag, Bi-tag and Tri-tag Frequencies for the BAAC.

Rank	Tag	Freq
1	noun	23782
2	verb	5801
3	prep	5574
4	adj	4995
5	noun_prop	2532
6	conj_sub	1501
7	conj	1212
8	pron_rel	1025
9	pron_dem	774
10	noun_quant	573

Table 5.10. Most frequent Tags from the BAAC.

Rank	Bi-tag	Freq	%
1	noun noun	11035	21.8
2	prep noun	4255	8.39
3	noun adj	4037	7.96
4	verb noun	3229	6.37
5	noun prep	2679	5.28
6	adj noun	1676	3.31
7	noun verb	1566	3.09
8	verb prep	1190	2.35
9	noun noun_prop	1066	2.10
10	noun_prop noun_prop	932	1.84

Table 5.11. Most frequent Bi-tag sequences from the BAAC.

Table 5.10, Table 5.11 and Table 5.12 illustrate the ten most frequently used tag, bi-tag and tri-tag sequences in the BAAC. The tag 'noun' was utilised 23,782 times (46.9%) followed by the tag 'verb' that appeared 5,801 times (11.44%) in the text, as shown in Table 5.10. The sequence of two nouns, the bi-tag 'noun noun', appeared on 11,035 occasions (21.76%), followed by the bi-tag 'prep noun' which was used 4,255 (8.39%) times in the BAAC, as shown in Table 5.11. The sequence of three nouns came 5,133 times in the text, which represents 10.12% of the

text, followed by the tri-tag 'noun prep noun' which came in 4.18% of the BAAC, as shown in Table 5.12.

Rank	Tri-tag	Freq	%
1	noun noun noun	5133	10.1
2	noun prep noun	2121	4.18
3	prep noun noun	1970	3.88
4	noun noun adj	1918	3.78
5	noun adj noun	1482	2.92
6	verb noun noun	1467	2.89
7	noun noun prep	1195	2.36
8	noun verb noun	909	1.79
9	verb prep noun	886	1.75
10	adj noun noun	858	1.69

Table 5.12. Most frequent Tri-tag sequences from the BAAC.

Rank	Tag	Freq	%
1	noun	485250	50.2
2	adj	120187	12.4
3	prep	104158	10.8
4	verb	91064	9.41
5	noun_prop	51985	5.37

Table 5.13. Most frequent Tag of the Khaleej sub-corpus 'News'.

To further analyse the n-tag results of the BAAC, Table 5.13, Table 5.14 and Table 5.15, show the tag, bi-tag and tri-tag statistics of the News sub-corpus from a different corpus, the Khaleej corpus [2], which also was tagged using the Madamira tagger. The sub-corpus contains 967K terms gathered from news websites. The tables shows that both corpora, News and the BAAC, share the same most frequent tag, bi-tag and tri-tag sequence, where the tag 'noun' in the sub-corpus News represents 50.2% of the text, as shown in Table 5.13, the bi-tag 'noun noun' was used 243,525 times (25.2%), as presented in Table 5.14, and the tri-tag 'noun noun noun'

appeared 122,386 times (0.13%) of the text, as shown in Table 5.15. These results confirm that the tag statistics are comparable between the different corpora.

Rank	Bi-tag	Freq	%
1	noun noun	243525	25.2
2	noun adj	91607	9.47
3	prep noun	81537	8.43
4	verb noun	52016	5.38
5	noun prep	48968	5.06

Table 5.14. Most frequent Bi-tag sequence of the Khaleej sub-corpus 'News'.

Rank	Tri-tag	Freq	%
1	noun noun noun	122386	0.13
2	noun noun adj	49187	0.05
3	prep noun noun	43107	0.04
4	noun prep noun	39116	0.04
5	noun adj noun	35544	0.04

Table 5.15. Most frequent Tri-tag sequence of the Khaleej sub-corpus 'News'.

5.9 BAAC Applications

The BAAC corpus was utilised in two applications, to evaluate the performance of the Madamira tagger, and to further investigate the tag-based text compression models as applied in Chapter 3. Using the BAAC corpus, the Madamira tagger achieved an accuracy of 93%. To evaluate the effect of manual annotation on the tag-based text compression, the two versions of the BAAC, which were obtained from the two annotators, were compressed using tag-based text compression models. The results of the compression were then compared to the compressed results of the original Madamira auto-tagged corpus. Table 5.16 illustrates the compression size (in bytes) and ratio (in bits per charactar) of all three files, the two versions of the BAAC which were obtained from the two annotators and the original Madamira auto-tagged version, and the results confirm that (1) manual annotation of the text reduces the quality of tag-based

compression, as reported in Chapter 3 and in [196], [198], [194], [66], [195], and (2) compressing the text using word-based and character-based text compression algorithms outperforms the tag-based text compression when compressing small text files, such as the BAAC corpus, as mentioned by Alhawiti and others [196], [26].

Annotator	File size	Compressed size (bytes)	Compression ratio (bpc)
1	824,151	111,009	1.0776
2	819,482	110,954	1.0832
Original File	818,508	110,874	1.0837

Table 5.16. Tag-based Compression Results.

Further investigation is required to study the effect of using POS tagging systems, such as the OpenNLP project [154], trained using the BAAC on the tag-based text compression. Future work will add more annotated MSA text and will expand to cover CA text. More linguistic information, such as the structural annotation, will also be added to the BAAC to increase the possible NLP applications of the corpus.

5.9 Summary and Discussion

A new corpus, BAAC, was presented in this chapter. It is an MSA corpus that contains 50K words manually annotated by part-of-speech tags. The annotated corpus used the same tagset utilised by the Madamira tagger and followed annotation guidelines proposed by Maamouri for annotating the POS tags. Also, a new annotation tool was developed and employed for the annotation process of BAAC which obtained a Kappa value of 0.956, and an average observed agreement of 94.25%. The BAAC was used to evaluate the Madamira tagger and to study the effect of the manual annotation on the performance of the tag-based Arabic text compression.

The next chapter will utilise the BAAC corpus and the results obtained in chapter 3 and 4 to develop novel compression-based criteria for evaluating Arabic part-of-speech taggers.

CHAPTER 6

Compression-based Tag Models for Evaluating Arabic Parts-of-Speech Taggers

Contents:

6.1 Introduction

The previous chapter described the creation of the new Bangor Arabic Annotated Corpus (BAAC). This chapter will utilise the BAAC corpus and the results obtained in chapter 3 and 4 to investigate the method of employing the compression results of the Arabic text that utilises both the POS (tags) and the text to evaluate the performance and the quality of two of the most commonly recognised Arabic POS taggers, the Madamira [161] and Stanford Arabic taggers [101].

The results in chapter 3 show that the precision and quality of the tagging process determines the quality of the tag-based compression of the Arabic text. It also concluded that compressing Arabic text using the tag-based compression models produced better results than the other two word-based and character-based methods. Since the main objective of this research is to develop and train a POS tagger for the Arabic language, this chapter will explore the use of compression results as a method of assessing the performance of a POS tagger when used to tag different types of text (as per research question 3). This is accomplished by illustrating the correlation between the quality of the tagging and the results of tag-based compression when used to compress the CA and MSA text that is tagged by two Arabic taggers. This chapter will also investigate the use of the tag-based compression output as a means of comparing the performance of two POS taggers. This is achieved by calculating the accuracy of two taggers using a gold-standard corpus, then comparing this with the tag-based compression results.

The chapter is organised as follows. Section 6.1 interduces the chapter. Section 6.2 describes the tag-based compression experiments on CA and MSA text. The results of the experiments are discussed in section 6.3. The performance of two taggers, the Madamira and Stanford Arabic taggers, are compared in section 6.4. Finally, section 6.5 presents a summary and discussion.

A portion of this chapter has been published in a conference paper (Alkhazi, I. S., & Teahan, W. (2019). Compression-based Tag Models for Evaluating Arabic Parts-of-speech Taggers. 2019 *IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology* (JEEIT) (JEEIT 2019). Amman, Jordan.)

6.2 CA and MSA Tag-based Compression Experiments

These experiments have used three corpora that have either CA or MSA text. The first corpus is Corpus A [30] which is an MSA corpus. The second corpus is King Saud University Corpus of Classical Arabic (KSUCCA) which is a CA corpus. The final corpus is the BACC corpus [26] which is a mixture of CA and MSA text. As stated, this corpus was originally created to investigate the performance of the character-based text compression on various Arabic text files. These corpora were tagged by two of the most popular Arabic POS taggers, The Madamira tagger [161] and Stanford tagger [101]. Finally, the Tawa toolkit [192] was used to perform the tag-based compression.

The experiments were conducted as follows:

- First, the three corpora that have either CA or MSA text were selected.

- Then, the tool described in Chapter 4 was utilised to classify the type of Arabic text used in each corpus.

- Thirdly, the Madamira and the Stanford taggers were used to tag all corpora.

- Lastly, tag-based compression was performed using the Tawa toolkit [192]. The compression results are shown in three tables, Table 6.1, Table 6.2 and Table 6.3, for each tagger.

The outcomes of tagging then compressing some of Corpus A's sub-texts are presented in Table 6.1, and for some of the BACC's sub-corpora are presented in Table 6.2. Table 6.3 shows the percentage of tag-based improvement using the two taggers to tag then compress some of the KSUCCA's sub-corpora. The following two sections discuss the findings of these experiments.

Sub-texts	Text Type	Char. encode size	Madamira Tag encode size	Stanford Tag encode size	Madamira Improv. (%)	Stanford Improv. (%)
Cinema	MSA	7,693,241	**7,318,652**	7,402,514	**5.12**	3.93
Crimes		1,341,031	**1,268,526**	1,272,853	**5.72**	5.36
Stories		4,213,806	**3,981,845**	4,004,927	**5.83**	5.22

Table 6.1. Tag-based compression improvement for various sub-texts in Corpus A.

Sub-texts	Text Type	Char. encode size	Madamira Tag encode size	Stanford Tag encode size	Madamira Improv. (%)	Stanford Improv. (%)
Arabic History	CA	**4,206,076**	4,267,257	4,286,946	-1.43	-1.89
Arabic Literature		**3,029,433**	3,045,281	3,059,687	-0.52	-0.99
Arabic Book 1		**164,445**	170,881	173,793	-3.77	-5.38

Table 6.2. The BACC Text Type and Tag-based compression improvement.

6.3 Different Texts Tagging Assessment

The goal of this section is to investigate the use of compression results as a means of evaluating the performance of a POS tagger when utilised to tag various types of text. Table 6.1, Table 6.2 and Table 6.3 show different corpora, that have either CA or MSA text, which were tagged by both the Madamira and the Stanford taggers and compressed using the Tawa toolkit.

As stated, most of the Arabic NLP systems, such as POS tagging, are developed for processing MSA text [84], as most of the available Arabic resources used for training, such as the Arabic Treebank (ATB) corpus [141], [211], are written in MSA. For example, both POS taggers used in the experiments section are trained using the ATB corpus [141]. As a consequence, many researchers, such as Alosimay and Alrabiah [42], [37], [38] have reported a drop in accuracy by

10 to 15% when one of the previous taggers is used for tagging CA text. The results in all three tables confirm that the quality of the tag-based text compression also drops when compressing a CA text that is tagged by one of the previous taggers. Therefore, a clear correlation between the drop in CA tagging quality reported by Alosimay and Alrabiah [42], [37], [38] and the decrease in the tag-based compression quality has been demonstrated.

6.4 Comparing the Performance of Two Taggers

In this section, the accuracy of two of the most popular known Arabic POS taggers, the Madamira and the Stanford taggers, were calculated using a gold-standard corpus for each tagger. For the Madamira tagger [161], the BAAC corpus was used to calculate the accuracy. As for the Stanford tagger [101], a version of the BAAC corpus was manually annotated as follows:

- A version of the BAAC corpus with only 5K terms was selected, then the tags were removed.

- The tagset used by the Stanford tagger was translated using the suggested translation by Alrabiah [42], [40], [137], then it was coded in the annotation tool.

- The same steps described in Chapter 5 were followed to manually annotate the gold-standard corpus for the Stanford tagger with the annotation process lasting two weeks.

Sub-texts	Text Type	Char. encode size	Madamira Tag encode size	Stanford Tag encode size	Madamira Improv. (%)	Stanford Improv. (%)
Religion	CA	**19,057,454**	19,189,175	19,209,861	-0.69	-0.79
Biography		**3,881,458**	3,920,285	3,937,428	-0.99	-1.42
Sociology		**3,713,723**	3,739,541	3,753,521	-0.69	-1.06

Table 6.3. KSUCCA Text Type and Tag-based compression improvement.

This corpus was used to evaluate the Stanford tagger. It is a 5K corpus written in MSA and manually annotated with POS tags. The annotator has corrected 663 of the incorrectly assigned tags, where the top 10 most frequently corrected tags are shown in Table 6.4, and a sample of corrected tags are shown in Table 6.5. The corpus used the Stanford tagset which consists of 24 tags, which was originally obtained by a manual reduction of the 135 tags taken from the ATB tagset.

The Tag	Modified To	Frequency
DTJJ	JJ	119
NNP	NN	73
VBP	VBD	48
NN	RP	38
JJ	NN	30
NN	VBD	26
DTNN	DTNNS	22
NN	RB	21
NN	NNP	17
NN	IN	16

Table 6.4. The most frequent corrected tags.

The idea is to use the two gold-standard corpora that contain the same text but tagged differently, to calculate the accuracy of the tagging process. The Madamira tagger achieved an accuracy of 93%, whereas the Stanford tagger achieved an accuracy of 86.4%. More information about the text used in this step is descibed in chapter 5. Also, all three tables show that the improvement in the tag-based compression quality of the text which is tagged by the Madamira tagger is slightly higher than that of the text which is tagged by the Stanford tagger. Therefore, an association between the high accuracy of the tagging and the high quality of compression of the two taggers can be derived, as tagging the text correctly leads to a better forecast of the forthcoming terms and, therefore, a better compression of the text.

6.5 Summary and Discussion

This chapter examined the feasibility of using the tag-based text compression results for Arabic text as a way of assessing the performance and quality of the Arabic POS taggers. First, the compression results were used to assess the performance of the Madamira tagger and the Stanford tagger when used on the two types of Arabic text, CA and MSA. Second, a correlation between the quality of the tagging process and the accuracy of the tagger illustrated by measuring the accuracy of two taggers, the Madamira and Stanford tagger, using a gold-standard corpus, then comparing the tag-based compression results on different corpora that were tagged using the previous two taggers.

The original tags	The corrected tags
NN/أن NN/قبل NNP/بدستور DTNN/الجمل NN/وأشاد WP/التي DTNNS/التعديلات NNP/عليه NN/تدخل NN/مدد NN/إطلاق NN/تعديل IN/من NN/بدءا NN/شوهته CC/و NNS/تعديلات CC/ثم DTNN/الجمهورية NN/رئيس VBD/وأن JJ/دستورية NN/خطايا VBD/اعتبرها WP/التي NNP/مصر VBD/أهانوا NN/صاغوها IN/من	RP/أن RB/قبل NN/بدستور NNP/الجمل VBD/وأشاد WP/التي DTNNS/التعديلات IN/عليه NN/تدخل NN/إطلاق NN/تعديل WP/من NN/بدءا VBD/شوهته NNS/تعديلات CC/ثم DTNN/الجمهورية NN/رئيس NN/مدد JJ/دستورية NN/خطايا VBD/اعتبرها WP/التي CC/و NNP/مصر VBD/أهانوا VBD/صاغوها IN/من RP/وأن
DTJJ/الفنية DTNNP/الأمانة NNP/بأن NNP/ونوه VBD/وجهت RP/قد DTJJ/القومي DTNN/الوفاق NN/بلجنة DTNN/المجتمع NN/أطياف NN/لكافة NNS/دعوات DTNN/المؤتمر IN/في NN/للمشاركة	NN/بلجنة JJ/الفنية DTNNP/الأمانة RP/بأن VBD/ونوه NNS/دعوات VBD/وجهت RP/قد JJ/القومي DTNN/الوفاق IN/في NN/للمشاركة DTNN/المجتمع NN/أطياف NN/لكافة DTNN/المؤتمر
NN/بين JJ/كاملا NN/تعاونا RB/هناك VBD/أن NN/وأكد DTJJ/المسلحة NNS/للقوات DTJJ/الأعلى DTNN/المجلس NNS/مجلسي NNS/انتخابات NN/وأن NN/والحكومة NNS/والانتخابات NN/والشورى DTNN/الشعب NN/موعدها IN/فى NNP/ستجرى DTJJ/الرئاسية DTNN/المقرر	RB/بين JJ/كاملا NN/تعاونا RB/هناك RP/أن VBD/وأكد DTJJ/المسلحة NNS/للقوات DTJJ/الأعلى DTNN/المجلس NNS/مجلسي NNS/انتخابات RP/وأن NN/والحكومة JJ/الرئاسية NNS/والانتخابات NN/والشورى DTNN/الشعب JJ/المقرر NN/موعدها IN/فى VBN/ستجرى

Table 6.5. A Sample of corrected tags.

Further study is required to examine the outcome of using the tag-based results to evaluate the ongoing effort which is currently been made to improve the performance of many NLP Arabic tasks which are designed for CA text, such as the making of manually annotated CA corpus by Alosaimy and Atwell [37], [38]. Also, the effect of using different Arabic resources, such as the BAAC corpus, to develop and train new Arabic POS taggers can be assessed by utilising tag-based compression results. Finally, the tag-based compression results can be used to compare

and confirm the tagging quality of different POS taggers, especially those which have different tagsets.

The next chapter will utilise the BAAC corpus to develop and train a compression-based Arabic part-of-speech tagger and will also apply the previous method to evaluate the newly developed tagger.

CHAPTER 7

Compression-based Parts-of-Speech Tagger for the Arabic Language

Contents:

7.1 Introduction

The previous chapter investigated a method of employing the compression results of the Arabic text that utilises both the POS (tags) and the text to evaluate the performance and the quality of two of the most commonly recognised Arabic POS taggers, the Madamira [161] and Stanford Arabic taggers [101]. This chapter investigates the development and training of a previously unpublished compression-based Arabic part-of-speech tagger. The new tagger utilises the Prediction by Partial Matching text compression scheme (PPM), which uses an adaptive statistical language model to make predictions about upcoming text and has been successfully applied to several Arabic NLP tasks, such as authorship attribution [46], [45], cryptology [15], text correction [19] and text compression [26], [29], but it has yet to have been applied to POS tagging. The adoption of the algorithm for Arabic POS tagging may increase the efficiency and reduce the Arabic language ambiguity problem (as per research questions 5 and 6).

This chapter will first discuss the sources used in the experiments in section 7.2. Then, it will discuss the two parts of the experiment, where silver-standard data is used in the first section to train the Tawa Arabic POS Tagger (TAPT), in section 7.3, and a gold-standard data, the BAAC corpus, is used in the second section as training data in section 7.4. The BAAC will be used to evaluate the tagger and limitations of those experiments are discussed in detail. In both sections, the effectiveness of using silver and gold-standard models will be examined by utilising the tag-based models to compress CA and MSA corpora tagged by the TAPT tagger. Finally, the summary and discussion are presented in section 7.5.

A portion of this chapter has been published in a journal paper (Alkhazi, I. S., & Teahan, W. (2019). Compression-based Parts-of-speech tagger for the Arabic Language. International Journal of Computational Linguistics, 10(1).)

7.2 Tawa tag encode models

According to Teahan [196], the best two models for encoding the tags are the TTT and TTWT models as shown in Table 7.1. First, the "TTT" model predicts the current tag using the prior two tags, and if no prediction was made, the model then escapes and employs just the prior tag. If using only the previous tag fails at predicting the current tag, the model escapes again

and predicts without any context. Compared to HMM taggers, such as the TnT tagger [59], if the current tag has not been recorded in this model and it's been seen for the first time, then the model will perform the last escape where a character-based model is utilised and each tag will have an equal probability. The second model is the "TTWT" model, which first attempts to predict the current tag utilising the previous tag, the previous word and the tag preceding that. If the attempt fails, the model employs an escape hierarchy similar to the "TTT" model. For efficiency reasons, the Tawa toolkit implements the TTT model rather than the slightly more effective TTWT model (in terms of compression).

WTW model	TTWT model	TTT model
$p(w_i \mid t_i w_{i-1})$ ↳ $p(w_i \mid t_i)$ ↳ $character\ model$	$p(t_i \mid t_{i-1}\, w_{i-1}\, t_{i-2})$ ↳ $p(t_i \mid t_{i-1}\, w_{i-1})$ ↳ $p(t_i \mid t_{i-1})$ ↳ $p(t_i \mid)$ ↳ $p_{eq}(t_i \mid)$	$p(t_i \mid t_{i-1}\, t_{i-2})$ ↳ $p(t_i \mid t_{i-1})$ ↳ $p(t_i \mid)$ ↳ $p_{eq}(t_i \mid)$

Table 7.1. Models for tag-based compression.

The WTW model is an n-gram model that first utilises the current tag with the previous word to predict the current word. If the word prediction fails, then the model escapes and uses only the current tag to predict the current word. If that also fails and no prediction was made, the model then backs-off or escapes to an order 4 character-model where every character in the term, which includes the space character to indicate the end of the term, is encoded separately. The models at this stage can be regarded as the "vocabulary" of the text since every word encoded is either unique, when utilising the word-based models or has been tagged uniquely, therefore in a sense, they can be regarded as the "vocabulary" of the text.

Tawa implements separate character models for each tag as this was found to lead to better compression. When the tag-based model has to back-off to the character model for an unknown word, the tag for that word will be known, therefore it can make use of a character model specifically trained on characters from previous words tagged in the same way. In essence, the PPM character model effectively learns the typical spelling characteristics for each tag in order to ensure better compression performance e.g. for the tag VBG, new words will invariably end

in the character sequence ‘ing’; for prepositions, new (rare) propositions will often contain texts from the more common prepositions that were encountered at the beginning of the text; and so on.

For predicting the word, either the TTT model or TTWT model is combined with the WTW model to create an n-pos model. The combined TTWT model and WTW model, for instance, is described by the next formula:

$$p(s) = \prod_{i=1}^{m} p'(t_i | t_{i-1}, w_{i-1}, t_{i-2})\ p''(w_i \mid t_i, w_{i-1}) \qquad (7.1)$$

where p' provides the probabilities passed by the TTWT model and p'' provides the probabilities passed by the WTW model.

To explain tag-based encoding in more detail, tables 7.2 and 7.3 present how the toolkit models a given string using the WTW and TTT models. The example in this case is how the PPMD prediction method models the string "to be or not to be to be or not to be that is the question" that has the following tag sequence "TO VB CC RB TO VB TO VB CC RB TO VB DT VBZ DT NN". In Table 7.3, WTW modelling is applied to the string. Table 7.2 presents how TTT models the previous string which is essentially the same as using an order 2 PPMD model. For simplification purposes, the tag sequence is translated into the following equivalent character sequence "tvcrtvtvcrtvdzdn", where "t" stands for "TO", "v" for "VB" etc. For both WTW and TTT models, calculating the probability for this example is similar to the character-based method explained in section 2.2.2.3 with the exception that the escape count equals 1 plus the number of symbols which have a count of 1 (these are called “singletons”). (This method for calculating the escape count for word-based models was found by Teahan [196] to yield better results in a range of compression experiments). The WTW model defaults to the character model as for Table 2.2.

7.3 Data Source

In the first part of the experiments, two sub-corpora of Corpus A [32] were used to train the TAPT tagger. As stated, Corpus A is an MSA corpus that includes various topics such as politics, opinions, legal issues, economics, conferences, business, cinema and books. The text

in Corpus A was gathered from the Al-Hayat website, a bilingual newspaper, and from the open-source online corpus, OPUS [30]. The second section of the experiments has utilised the BAAC corpus to train and evaluate the TAPT tagger. The Bangor Arabic Annotated Corpus (BAAC) is an MSA corpus that comprises 50K words manually annotated by parts-of-speech that was described in Chapter 5. The data source for the new corpus is the Press sub-corpus from the BACC corpus [26], which was created originally to test the performance of various text compression algorithms on different text files. The results of the text classification and segmentation in Chapter 4 revealed that the Press sub-corpus is mostly written in MSA, as shown in tables 4.7, 4.11 and 4.15. According to Alhawiti [26], the sub-corpus is a newswire text consisting of 50K terms, gathered from various news websites between 2010 and 2012 and covers many topics such as political and technology news.

Order 2			**Order 1**			**Order 0**			**Order -1**		
Prediction	***c***	***p***	**Prediction**	***c***	***p***	**Prediction**	***c***	***p***	**Prediction**	***c***	***p***
"tv" → "c"	2	3/8	"t" → "v"	4	7/8	→ "v"	3	5/22	→ *A*	1	1/|*A*|
→ "t"	1	1/8	→ esc	1	1/8	→ "c"	2	3/22			
→ "d"	1	1/8	"v" → "c"	2	3/8	→ "r"	2	3/22			
→ esc	3	3/8	→ "t"	1	1/8	→ "t"	4	7/22			
"vc" → "r"	2	3/4	→ "d"	1	1/8	→ "d"	2	3/22			
→ esc	1	1/4	→ esc	3	3/8	→ "z"	1	1/22			
"cr" → "t"	2	3/4	"c" → "r"	2	3/4	→ "n"	1	1/22			
→ esc	1	1/4	→ esc	1	1/4	→ esc	7	7/22			
"rt" → "v"	2	3/4	"r" → "t"	2	3/4						
→ esc	1	1/4	→ esc	1	1/4						
"vt" → "v"	1	1/2	"d" → "z"	1	1/4						
→ esc	1	1/2	→ "n"	1	1/4						
"vd" → "z"	1	1/2	→ esc	2	2/4						
→ esc	1	1/2	"z" → "d"	1	1/2						
"dz" → "d"	1	1/2	→ esc	1	1/2						
→ esc	1	1/2									
"zd" → "n"	1	1/2									
→ esc	1	1/2									

Table 7.2. The TTT processing of the tag sequence "TO VB CC RB TO VB TO VB CC RB TO VB DT VBZ DT NN" which is converted into "tvcrtvtvcrtvdzdn" for illustration purposes and *A* is the number of tags.

$p(w_i \mid t_i\ w_{i-1})$				$p(w_i \mid t_i)$				$p(w_i\|)$		
Prediction		***c***	***p***	**Prediction**		***c***	***p***	**Prediction**	***c***	***p***
"VB to"	→ "be"	4	4/5	"TO"	→ "be"	4	4/5	→ "be"	4	4/20
	→ esc	1	1/5		→ esc	1	1/5	→ "or"	2	2/20
"CC be"	→ "or"	2	2/3	"VB"	→ "or"	2	2/7	→ "to"	3	3/20
	→ esc	1	1/3		→ "to"	1	1/7	→ "that"	1	1/20
"RB or"	→ "not"	2	2/3		→ "that"	1	1/7	→ "not"	2	2/20
	→ esc	1	1/3		→ esc	3	3/7	→ "is"	1	1/20
"TO not"	→ "to"	2	2/3	"CC"	→ "not"	2	2/3	→ "question"	1	1/20
	→ esc	1	1/3		→ esc	1	1/3	→ "the"	1	1/20
"TO be"	→ "to"	1	1/3	"RB"	→ "to"	2	2/3	→ esc	5	5/20
	→ esc	2	1/3		→ esc	1	1/3			
"DT be"	→ "that"	1	1/3	"DT"	→ "is"	1	1/5			
	→ esc	2	1/3		→ "question"	1	1/5			
"VBZ that"	→ "is"	1	1/3		→ esc	3	3/5			
	→ esc	2	1/3	"VBZ"	→ "the"	1	1/3			
"DT is"	→ "the"	1	1/3		→ esc	2	1/3			
	→ esc	2	1/3							
"NN the"	→ "question"	1	1/3							
	→ esc	2	1/3							

Table 7.3. The WTW processing of the string "to be or not to be to be or not to be that is the question".

A new one-to-one transliteration tool was developed and then used in both experiments to transliterate Arabic characters to Latin characters. The new tool is based on the Buckwalter Arabic transliteration tool [65], [138] developed by Tim Buckwalter. The new mapping, as shown in Table 7.4, adds Arabic numbers and some Quranic symbols that were found in CA corpora used in the experiments. The tool was utilised to transliterate training and input text for the TAPT tagger to Latin characters and the output tagged text to Arabic characters. Figure 7.1 shows a sample Arabic transliterated text using the developed transliteration tool.

Arabic Character	Latin Character	Arabic Character	Latin Character	Arabic Character	Latin Character
\u0621	q	\u0634	z	\u064C	D
\u0622	w	\u0635	x	\u064D	F
\u0623	e	\u0636	c	\u064E	R
\u0624	r	\u0637	v	\u064F	W
\u0625	t	\u0638	b	\u0650	U
\u0626	y	\u0639	n	\u0651	S
\u0627	u	\u063A	m	\u0652	E

Table 7.4. A sample of the new character mapping.

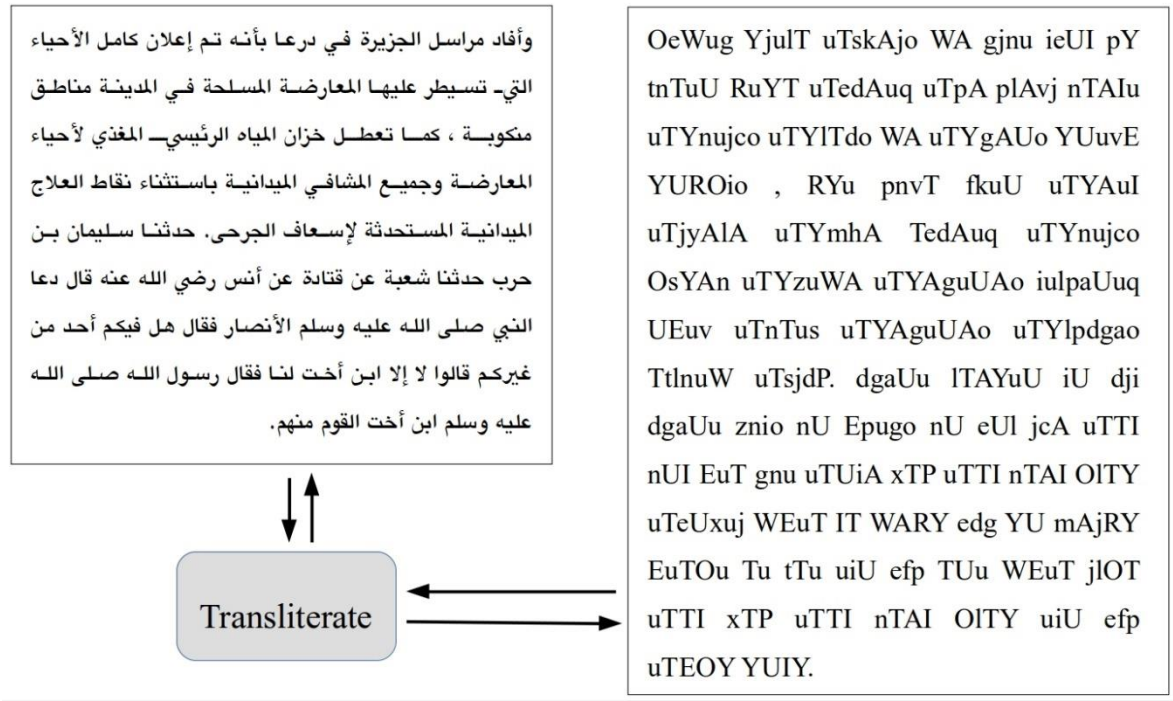


Figure 7.1. Sample Arabic transliterated text.

7.4 Silver-standard Data Experiment

This section illustrates the use of silver-standard data, which was tagged using both the Madamira and the Stanford taggers, to train and then evaluate the TAPT tagger. The experiment was conducted as follows:

- Corpus A was first tagged using Madamira and the Stanford taggers.
- Then, the text was preprocessed and input into the Tawa toolkit [192] then transliterated to Latin characters.
- Next, two PPM tagging models were created, the first model was trained using Madamira tagged text and the second model was trained using Stanford tagged text.
- Finally, a smaller version of the BAAC corpus, that has only 5K terms, was selected then tagged using the two models from the previous step.

Tables 7.5 and 7.6 show the top 10 most incorrectly assigned tags for the TAPT tagger trained on silver-standard Madamira and Stanford models. To calculate the accuracy of using silver-standard data to train the TAPT tagger, the Madamira and Stanford gold-standard data described in Chapter 6 was used to establish the number of incorrectly assigned tags. The tagger achieved an accuracy of 84%, with 794 incorrectly assigned tags, using the Madamira silver-standard model, and 81% using the Stanford silver-standard model with 927 incorrectly assigned tags. Table 7.7 demonstrates the most incorrectly assigned tags for the TAPT tagger which was trained by silver-standard text tagged by Madamira POS tagger. Table 7.6 displays the most incorrectly assigned tags for the TAPT tagger which was trained by silver-standard text tagged by Stanford POS tagger.

The results in Table 7.7 show that almost 25.56% of the incorrectly assigned tags by the TAPT tagger that used the Madamira model were in fact verbs and 8.18% were nouns, which includes noun_prop and noun. Compared to the Stanford model, as shown in in Table 7.6, only 5.17% of the inaccurately assigned tags by the TAPT tagger that used the Stanford model were in fact verbs whereas 29.34% of the inaccurately assigned tags were nouns, that includes NNP, NN and DTNN. The previous results confirm the results in Chapter 6 which suggest that there is an issue in the process of assigning the verb tag by the Madamira tagger and the noun tag by the Stanford tagger.

Frequency	Madamira Assigned Tag	BAAC Tag
165	noun	verb
51	noun	adj
46	conj_sub	verb_pseudo
34	noun	abbrev
27	adj	noun
21	noun_prop	noun
20	prep	verb_pseudo
17	verb	abbrev
17	noun	noun_prop
16	prep	part_neg

Table 7.5. Top 10 most incorrectly assigned tags for TAPT trained on silver-standard Madamira model.

Frequency	Stanford Assigned Tag	BAAC Tag
118	JJ	DTJJ
64	NN	NNP
48	VBD	VBP
45	VBD	NN
44	RP	NN
37	NNP	NN
37	NN	JJ
36	NNP	DTNN
24	DTNNS	DTNN
22	RB	NN

Table 7.6. Top 10 most incorrectly assigned tags for TAPT trained on silver-standard Stanford model.

To evaluate the performance of the TAPT tagger that was trained on Madamira silver-standard text, the BACC corpus was tagged then compressed using tag-based compression models. The BACC corpus as stated in Chapter 3, is a mixture of MSA and CA text. Table 7.7 and Table 7.8 represent the results of compressing the BACC sub-corpora *'Arabic History'*, *'Arabic Literature'*, *'Art and Music'* and *'Sports'*. The two tables show that the tag-based compression performance on the text that was tagged by TAPT, that was trained on silver-standard text, has decreased compared to the performance of the Madamira tag-based compression.

Sub-text	**Text Type**	**Corpus Size**	**Character-based Compression size**	**Madamira Tag-based Compression size**	**TAPT Tag-based Compression size**
Arabic History	CA	30251137	4206076	4267257	4290052
Arabic Literature		18594383	3029433	3045281	3067010
Art and Music	MSA	41770	9510	10583	10604
Sports		31059	6497	7124	7149

Table 7.7. The character-based and the tag-based compression results of the Madamira and TAPT trained on silver-standard corpus.

Sub-text	**Text Type**	**Character-based bpc**	**Madamira bpc**	**TAPT bpc**	**TAPT Performance Decrease**
Arabic History	CA	1.11	1.13	1.13	-0.52%
Arabic Literature		1.30	1.31	1.32	-0.70%
Art and Music	MSA	1.82	2.03	2.03	-0.18%
Sports		1.67	1.83	1.84	-0.32%

Table 7.8. The decrease in the tag-based compression performance of TAPT trained on silver-standard text compared to the Madamira tagger.

7.5 Gold-standard Data Experiment

This section represents the use of a gold-standard annotated text, the BAAC corpus, to train and then evaluate TAPT. Using a tenfold cross validation method, TAPT achieved an accuracy of 93% when trained using the BAAC corpus. Table 7.9 shows the most frequently assigned tags by TAPT and Table 7.10 displays the most incorrectly assigned tags compared to the tag at the BAAC corpus.

Frequency	Tag
24787	noun
5693	prep
5584	verb
4431	adj
2519	noun_prop
1656	conj_sub
1148	conj
985	pron_rel
765	pron_dem
599	noun_quant
500	part_neg
355	pron
329	adv
251	noun_num

Table 7.9. The most frequently assigned tags by TAPT trained on gold-standard text.

To evaluate the performance of the TAPT tagger when trained on gold-standard text, four BACC sub-corpora were first tagged by the TAPT tagger and then the text was compressed using tag-based compression models. Table 7.11 compares the results of compressing the BACC sub-corpora *'Arabic History', 'Arabic Literature', 'Art and Music'* and *'Sports'* using the character-based and the tag-based model. Both *'Arabic History' and 'Arabic Literature'* are 99% written in CA text, whereas *'Art and Music' and 'Sports'* are 91% and 95% consecutively, written in MSA text. Table 7.12 shows the tag-based compression ratio (in bits per character) of the

four BACC sub-corpora which were tagged by the TAPT tagger and the Madamira tagger. It is noticeable that the quality of compression of the *'Art and Music' and 'Sports' sub-corpora* has increased by 4.98% and 4.25% respectively, whereas the compression quality of the sub-corpora, *'Arabic History' and 'Arabic Literature'*, has decreased by 2.69% and 1.56% respectively, compared to the tag-based compression results of the Madamira tagger.

Frequency	PPM Assigned Tag	BAAC Tag
73	noun	adj
45	adj	noun
41	verb	noun
19	noun	verb
12	noun_prop	noun
12	noun	conj
11	noun	noun_prop
10	conj_sub	verb_pseudo
5	noun_prop	verb
5	adv	adv_interrog

Table 7.10. Top 10 most incorrectly assigned tags for TAPT trained on gold-standard corpus.

The results in Table 7.11 and 7.12 indicate that tagging MSA text using the TAPT tagger increases the quality of the tag-based compression compared to the Madamira tagged text. The results also show that the quality of the tag-based compression of CA text that was tagged by the TAPT tagger has decreased. A possible cause of improvement in compressing the MSA corpora is the fact that the TAPT tagger is trained using the BAAC corpus which is mostly written in MSA as concluded in Chapter 5.

Sub-text	Text Type	Corpus Size	Character-based Compression size	Madamira Tag-based Compression size	TAPT Tag-based Compression size
Arabic History	CA	30251137	4206076	4267257	4387191
Arabic Literature		18594383	3029433	3045281	3093824
Art and Music	MSA	41770	9510	10583	10027
Sports		31059	6497	7124	6807

Table 7.11. The character-based and the tag-based compression results of the Madamira and TAPT trained on gold-standard corpus.

Sub-text	Text Type	Character-based bpc	Madamira bpc	TAPT bpc	TAPT Improvement
Arabic History	CA	1.11	1.13	1.16	-2.69%
Arabic Literature		1.30	1.31	1.33	-1.56%
Art and Music	MSA	1.82	2.03	1.92	4.98%
Sports		1.67	1.83	1.75	4.25%

Table 7.12. The tag-based compression improvement of TAPT trained on gold-standard corpus compared to the Madamira tagger.

7.6 Summary and Discussion

This chapter presented a newly developed compression-based POS tagger for the Arabic language which is based on a Prediction-by-Partial Matching (PPM) compression system. The results of the tagger were presented in two experiments. The first used models which were trained using silver-standard data from two different POS Arabic taggers, the Stanford and the

Madamira taggers [161], [74]. The results of the previous experiment show that using silver-standard data to train the TAPT tagger decreases the quality of the tag-based compression of both the CA and MSA text compared to the Madamira tagger. The second experiment trained a model using the BAAC corpus, which is a 50K term manually annotated MSA corpus, where the TAPT tagger achieved an accuracy of 93%. The tag-based compression results of the second experiment show that the use of the gold-standard model increases the quality of the tag-based compression when the TAPT tagger is used to tag MSA text.

Future enhancements to the tagger can be made by utilising more Arabic resources, such as the ‘Sunnah Arabic Corpus' [38] which is a set of CA text that is popularly cited in Islamic books and the ATB corpus [111]. Including such resources might increase the accuracy of the TAPT tagger.

CHAPTER 8

Conclusion

Contents:

8.1 Summary

This chapter examines the achievements of this study. First, it presents a summary in section 8.1. Then, it reviews the aim and objectives of this research in section 8.2 and examines the research questions in section 8.3. Section 8.4 presents the limitations of this study. Lastly, a number of suggestions are offered in section 8.5.

The Arabic language is a morphologically complex language that causes various difficulties for various NLP systems, such as POS tagging. The statistical method of tagging the Arabic text is broadly utilised to solve the POS uncertainty of the Arabic text [180]. Chapter 7 investigated the development and training of a compression-based Arabic POS tagger using the PPM algorithm. The new tagger (TAPT) was trained using silver-standard data and gold-standard. The results show that using silver-standard data to train the TAPT tagger decreases the quality of the tag-based compression of both the CA and MSA text compared to the Madamira tagger. The second experiment trained a model using the BAAC corpus, which is a 50K term manually annotated MSA corpus, where TAPT achieved an accuracy of 93%. The tag-based compression results of the second experiment show that the use of the gold-standard model increases the quality of the tag-based compression when TAPT is used to tag MSA text.

Previous studies were conducted to examine the performance of the tag-based compression of the Arabic text [26], where the only resource used was the Arabic Treebank Corpus (ATC) [26]. As the best text compression algorithms can be applied to natural language processing tasks often with state-of-the-art results [196], [193], [195], [197], [15], and the improved tag-based compression has applications beyond the specific compression application, Chapter 3 examined the use of tag-based compression of larger Arabic resources to re-evaluate the performance of tag-based compression. The results of the experiments in this Chapter 6 showed that the tag-based compression of the text can effectively be used for assessing the performance of Arabic POS taggers when used to tag different types of the Arabic text, and also as a means of comparing the performance of two Arabic POS taggers on the same text.

Some Arabic corpora, such as the Bangor Arabic Compression Corpus (BACC), is a mixture of both CA and MSA text. The results of using such a corpus in order to perform various NLP tasks will vary and will not be consistent and reliable. Studies that address the problems of

classification and segmentation of the Arabic language are limited compared to other languages, most of which implement word-based and feature extraction algorithms. Chapter 4 adopted a PPM character-based compression scheme to classify and segment Classical Arabic (CA) and Modern Standard Arabic (MSA) texts. An initial experiment using the PPM classification method on samples of text resulted in an accuracy of 95.5%, an average precision of 0.958, an average recall of 0.955 and an average F-measure of 0.954, using the concept of minimum cross-entropy. Segmenting the CA and MSA text using the PPM compression algorithm obtained an accuracy of 86%, an average precision of 0.869, an average recall of 0.86 and an average F-measure of 0.859.

POS annotated corpora are essential for the development of many NLP systems, such as part-of-speech tagging [180]. The lack of such resources limits some researchers from progressing further in their efforts. The limited availability of some existing annotated corpora and the cost of acquiring others are one of the main reasons that contribute to resource scarcity. Chapter 5 described the creation of the new Bangor Arabic Annotated Corpus (BAAC) which is a Modern Standard Arabic (MSA) corpus that comprises 50K words manually annotated by parts-of-speech. For evaluating the quality of the corpus, the Kappa coefficient and a direct percent agreement for each tag were calculated for the new corpus and a Kappa value of 0.956 was obtained, with an average observed agreement of 94.25%. The corpus was used to evaluate the widely used Madamira Arabic POS tagger and to further investigate compression models for text compressed using POS tags. Also, a new annotation tool was developed and employed for the annotation process of the BAAC.

8.2 Review of Aim & Objectives

The aim and objectives of this thesis which have been proposed in Section 1.2 have all been successfully achieved. A novel compression-based Arabic part-of-speech tagger based on PPM was developed and the new tagger was evaluated using a novel compression-based criterion. The new tagger utilised the newly created POS annotated corpus. Also, MSA and CA text were classified and segmented using a PPM character-based text compression scheme.

Therefore, the particular objectives as described in section 1.2 were accomplished as follows:

- *Investigate the most efficient PPM compression method of the Arabic text.*

  Chapter 3 examined the use of tag-based compression of larger Arabic resources to re-evaluate the performance of tag-based compression. The results of compressing tagged and untagged texts show that using tag-based compression significantly outperforms both the word-based and character-based models, and the added extra-tag information improves overall compression compared to the untagged compressed text.

- *Investigate the applications of PPM tag-based compression to several Arabic NLP tasks*.

  The novel PPM compression-based criterion was utilised in Chapter 4 to confirm the classification and segmentation results and as a means of comparing the performance of two POS taggers in Chapter 6.

- *Develop novel methods for classification and segmentation of Classical Arabic and Modern Standard Arabic text using PPM*.

  Classification of Classical Arabic (CA) and Modern Standard Arabic (MSA) text was performed in Chapter 4 on sample texts using a PPM character-based compression scheme achieving an accuracy of 95.5%, an average precision of 0.958, an average recall of 0.955 and an average F-measure of 0.954. Segmenting the CA and MSA text using the PPM compression algorithm obtained an accuracy of 86%, an average precision of 0.869, an average recall of 0.86 and an average F-measure of 0.859. Further classification and segmentation experiments were conducted in Chapter 4 to analyse mixed Arabic corpora and the results showed that different Arabic corpora have a mixture of CA and MSA text.

- *Create and evaluate a new POS manually annotated Arabic Corpus*.

  A new corpus, BAAC, was presented in Chapter 5. It is an MSA corpus that contains 50K words manually annotated by part-of-speech tags. The annotated corpus obtained a Kappa value of 0.956, and an average observed agreement of 94.25%. The BAAC was used to evaluate the Madamira tagger and to study the effect of the manual annotation on the performance of the tag-based Arabic text compression.

- *Develop and train a novel compression-based Arabic part-of-speech tagger based on PPM*.

  The previous chapter presented a newly developed compression-based POS tagger for the Arabic language (TAPT) which is based on a Prediction-by-Partial Matching (PPM) compression system. The new tagger was trained using the BAAC corpus, which is a 50K term manually annotated MSA corpus, and achieved an accuracy of 93%. The tag-based compression results show that the use of the gold-standard model increases the quality of the tag-based compression when the TAPT tagger is used to tag MSA text.

- *Develop novel compression-based criteria for evaluating Arabic part-of-speech.*
  Chapter 6 examined the feasibility of using the tag-based text compression results for Arabic text as a way of assessing the performance and quality of the Arabic POS taggers. First, the compression results were used to assess the performance of two taggers when used on the two types of Arabic text, CA and MSA. Second, a correlation was found between the quality of the tagging process and the accuracy of the tagger illustrated by measuring the accuracy of two taggers, the Madamira and Stanford tagger, using a gold-standard corpus, then comparing the tag-based compression results on different corpora that were tagged using the previous two taggers.

8.3 Review of Research Questions

This section reviews the research questions which were laid out in section 1.3. It will list the question and the discussion of the experimental findings which relate to that question.

The research questions were as follows:

- *Can the PPM compression models be used to help reveal linguistic universals across languages*?
  The results in chapters 3, 4 and 6 show that there is a difference in quality between compression for CA and MSA text, which resulted from the tagging quality. This results

combined with the findings by Alkahtani in [30] indicate that PPM compression models can be utilised to reveal linguistic universals across single and multiple languages.

- *What is the best PPM compression model for compressing Arabic text*?
  The findings in Chapter 3 show that the tag-based compression of the MSA text outperforms both the word-based and character-based compression.

- *Can the tag-based compression of the Arabic text be utilised to measure the performance of various Arabic POS taggers*?
  The experimental findings in Chapter 6 illustrated the correlation between the quality of the tag-based compression and the accuracy of the tagger. This novel PPM compression-based criterion was utilised in the final chapter to estimate the tagging quality of the new tagger.

- *Can two types of non-colloquial written text for the Arabic language be classified using the PPM compression models*?
  PPM as a minimum cross-entropy text classifier was successfully adopted to classify and segment Classical Arabic (CA) and Modern Standard Arabic (MSA) texts. Further classification and segmentation experiments were conducted in Chapter 4 to analyse mixed Arabic corpora and the results showed that different Arabic corpora have a mixture of CA and MSA text.

- *Can a new POS annotated corpus be used to develop and train a new compression-based Arabic part-of-speech tagger that is effective at tagging Arabic text*?
  A new MSA corpus, that contains 50K words manually annotated by part-of-speech tags, was presented in Chapter 5. The corpus was successfully utilised in Chapter 7 to train the TAPT tagger. The new tagger achieved an accuracy of 93%.

- *Will the adoption of the PPM compression models to tag the Arabic text increase the performance of tagging MSA text compared to other Arabic taggers*?
  The tag-based compression results of the second experiment in Chapter 7 show that the use of the BAAC corpus to train the new tagger increases the quality of the tag-based compression when TAPT is used to tag MSA text.

8.4 Limitations

The limitations are as follows:

- Since PPM tag-based model uses three streams, a tag stream, a word stream and a character stream to build its model, compressing the text using the tag-based model will require more time and resources compared to a character-based model by itself.

- The quality of the tag-based compression depends on the quality of the tagging process. This means that the compression of certain text, CA text for example, may not produce results similar to the compression of MSA text.

- Compared to other paid annotated corpora, such as the ATB corpus, the number of terms in the BAAC is modest.

- The BAAC is only annotated by POS, which limits the use of the corpus by other NLP applications.

- Similar to most known Arabic POS taggers, TAPT is trained on MSA text, therefore, the quality of tagging CA text will be affected.

- A manual similarity analysis was performed on a sample of unpublished Arabic corpora, such as Ajdir Corpora [5], which were gathered using a Web crawler, and text duplicates were discovered. The duplicate content may bias results derived from the processing of such corpora by artificially inflating frequencies of some words and expressions.

8.4 Future Work

The future work is as follows:

- TAPT was trained using the BAAC, which is written in MSA text. Future enhancements to the tagger can be made by utilising more Arabic resources, such as the ‘Sunnah Arabic Corpus' [38] which is a set of CA text that is popularly cited in Islamic books.

Including such resources might increase the accuracy of TAPT when utilised to tag CA text. The quality of tagging MSA can also be improved by utilising more MSA resources such as the ATB corpus [111].

- Different POS taggers adopt various segmentation schemes. This scheme differs from tagging every prefix/infix/suffix of the word, such as the three degrees of segmentation structure described by Habash and Sadat [107], to neglecting some of the text, such as punctuations, numbers, dates, etc. Aligning the segmentation output of different taggers is proposed to evaluate the segmentation scheme [38], [50], however, this process is "quite sophisticated" [50]. Other ways, such as the GRACE evaluation task [11], the AMALGAM project [50], are proposed also for the evaluation of different segmentation schemes. Since the quality of tag-compression improves when the text is segmented, as shown in Chapter 3, further investigations are required to examine where the tag-based compression can be utilised as a way for evaluating the performance of different segmentation schemes.

- Many metrics are available for measuring the similarity of documents, such as Levenshtein edit distance [134], [24] and Broder's resemblance [62]. For small data sets, the duplicate lines can be detected by comparing the similarity value between the two lines, and near-duplicates lines can also be identified by reporting lines that have similarity value above a certain threshold. However, the resources required to apply the previous approaches to large data sets may be computationally expensive, therefore, applying other approaches, such as Charikar's algorithm [67], Pugh and Henzinger's algorithm [167] and Shivakumar and Garcia-Molina fingerprinting scheme [183], may become more applicable. Future investigations are required to examine the application of PPM for the detection of duplicates and near-duplicates found in the text by utilising the codelength and the cross-entropy of the compressed text.

- The utilisation of PPM compression scheme by TAPT has successfully increased the tagging quality of MSA text. Further investigations are required to utilise the scheme in more natural language processing tasks for the Arabic language such as tokenization and phrase chunking. This can be performed by training TAPT using, for example, tokenized resources, such as the ATB corpus [111].

- The BAAC corpus was presented in Chapter 5. Further work is needed to increase the number of MSA terms and include CA text to increase the possible NLP applications of the corpus. More linguistic information, such as the structural annotation, and morphological features should also be added to the BAAC.

References

[1] M. J. A. ˇZeljko Agic *et al.*, “Universal dependencies 1.1,” *LINDAT/CLARIN Digit. Libr. Inst. Form. Appl. Linguist. Charles Univ. Prague*, 2015.

[2] M. Abbas, K. Smaili, and D. Berkani, “Evaluation of Topic Identification Methods for Arabic Texts and their Combination by using a Corpus Extracted from the Omani Newspaper Alwatan,” *Arab Gulf J. Sci. Res.*, vol. 29, no. 3–4, pp. 183–191, 2011.

[3] M. Abbas, “Khaleej corpus.” [Online]. Available: https://sourceforge.net/projects/arabiccorpus/. [Accessed: 22-Sep-2018].

[4] A. Abbasi, H. Chen, and A. Salem, “Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums,” *ACM Trans. Inf. Syst.*, vol. 26, no. 3, p. 12, 2008.

[5] A. Abdelali, “Ajdir Corpora | E3rab.com.” [Online]. Available: http://aracorpus.e3rab.com/argistestsrv.nmsu.edu/AraCorpus/. [Accessed: 22-Sep-2018].

[6] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak, “Farasa: A fast and furious segmenter for arabic,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 2016, pp. 11–16.

[7] N. Abdulla, N. Mahyoub, M. Shehab, and M. Al-Ayyoub, “Arabic sentiment analysis: Corpus-based and lexicon-based,” in *Proceedings of The IEEE conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, 2013.

[8] K. Abu Kwaik, M. K. Saad, S. Chatzikyriakidis, and S. Dobnik, “Shami: A Corpus of Levantine Arabic Dialects,” *Shami A Corpus Levantine Arab. Dialects*, 2018.

[9] R. A. Abumalloh, H. M. Al-Sarhan, and W. Abu-Ulbeh, “Building Arabic Corpus Applied to Part-of-Speech Tagging,” *Indian J. Sci. Technol.*, vol. 9, no. 46, 2016.

[10] R. A. Abumalloh, H. M. Al-Sarhan, O. Ibrahim, and W. Abu-Ulbeh, “Arabic Part-of-Speech Tagging,” *J. Soft Comput. Decis. Support Syst.*, vol. 3, no. 2, pp. 45–52, 2016.

[11] G. Adda, J. Mariani, J. Lecomte, P. Paroubek, and M. Rajman, “The GRACE French part-ofspeech tagging evaluation task,” in *in Proceedings of the First International Conference on Language Resources and Evaluation*, 1998, pp. 433–441.

[12] C. C. Aggarwal and C. Zhai, “A survey of text classification algorithms,” in *Mining text data*, Springer, 2012, pp. 163–222.

[13] S. Al-Harbi, A. Almuhareb, A. Al-Thubaity, M. S. Khorsheed, and A. Al-Rajeh, “Automatic Arabic text classification,” in *Proceedings of The 9th International Conference on the Statistical Analysis of Textual Data*, 2008.

[14] M. N. Al-Kabi, N. A. Abdulla, and M. Al-Ayyoub, “An analytical study of arabic sentiments: Maktoob case study,” in *Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for*, 2013, pp. 89–94.

[15] N. R. Al-Kazaz, S. A. Irvine, and W. J. Teahan, “An Automatic Cryptanalysis of Transposition Ciphers Using Compression,” in *International Conference on Cryptology and Network Security*, 2016, pp. 36–52.

[16] L. Al-Sulaiti and E. S. Atwell, “The design of a corpus of Contemporary Arabic,” *Int. J. Corpus Linguist.*, vol. 11, no. 2, pp. 135–171, 2006.

[17] A. O. Al-Thubaity, “A 700M+ Arabic corpus: KACST Arabic corpus design and construction,” *Lang. Resour. Eval.*, vol. 49, no. 3, pp. 721–751, 2015.

[18] M. Alabbas and A. Ramsay, “Improved POS-tagging for Arabic by combining diverse taggers,” in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, 2012, pp. 107–116.

[19] M. M. Alamri and W. J. Teahan, “Automatic Correction of Arabic Dyslexic Text,” *Computers*, vol. 8, no. 1, p. 19, 2019.

[20] S. Alansary and M. Nagi, “The international corpus of Arabic: Compilation, analysis and evaluation,” in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, 2014, pp. 8–17.

[21] A. Y. G. Alfaifi, E. Atwell, and I. Hedaya, “Arabic learner corpus (ALC) v2: a new written and spoken corpus of Arabic learners,” in *Proceedings of Learner Corpus Studies in Asia and the World 2014*, 2014, vol. 2, pp. 77–89.

[22] A. Y. G. Alfaifi, “Building the Arabic Learner Corpus and a System for Arabic Error Annotation,” University of Leeds, 2015.

[23] S. AlGahtani and J. McNaught, “Joint Arabic Segmentation and Part-Of-Speech Tagging,” in *Proceedings of the Second Workshop on Arabic Natural Language Processing*, 2015, pp. 108–117.

[24] M. A. Alghamdi, I. S. Alkhazi, and W. J. Teahan, “Arabic OCR Evaluation Tool,” in *Computer Science and Information Technology (CSIT), 2016 7th International Conference on*, 2016, pp. 1–6.

[25] M. A. Alghamdi and W. J. Teahan, “A New Thinning Algorithm for Arabic Script,” *Int. J. Comput. Sci. Inf. Secur.*, vol. 15, no. 1, p. 204, 2017.

[26] K. M. Alhawiti, “Adaptive models of Arabic text,” Ph.D. thesis, Bangor University, 2014.

[27] A. H. Aliwy, “Arabic morphosyntactic raw text part of speech tagging system,” Ph.D. thesis, 2013.

[28] Aljazeera.net, “الجزيرة.نت.” [Online]. Available: http://www.aljazeera.net/portal. [Accessed: 18-Mar-2017].

[29] N. O. M. Aljehane, “Grammar-based Preprocessing for PPM Compression and Classification,” Ph.D. thesis, Bangor University, 2018.

[30] S. Alkahtani, “Building and verifying parallel corpora between Arabic and English,” Ph.D. thesis, Bangor University, 2015.

[31] S. Alkahtani and W. J. Teahan, “Aligning a New Parallel Corpus of Arabic-English,” in *Proceedings of the Eighth Saudi Students Conference in the UK*, 2015, p. 279.

[32] S. Alkahtani and W. J. Teahan, “A new parallel corpus of Arabic/English,” in *Proceedings of the Eighth Saudi Students Conference in the UK*, 2016, pp. 279–284.

[33] M. I. Alkanhal, M. A. Al-Badrashiny, M. M. Alghamdi, and A. O. Al-Qabbany, “Automatic stochastic arabic spelling correction with emphasis on space insertions and deletions,” *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 20, no. 7, pp. 2111–2122, 2012.

[34] Alkunuz.co.uk, “Islamic books | Leicester | Al Kunuz.” [Online]. Available: https://www.alkunuz.co.uk/. [Accessed: 23-Sep-2018].

[35] A. Almahdawi and W. J. Teahan, “Automatically Recognizing Emotions in Text Using Prediction by Partial Matching (PPM) Text Compression Method,” in *International Conference on New Trends in Information and*

Communications Technology Applications, 2018, pp. 269–283.

[36] K. Almeman, M. Lee, and A. A. Almiman, “Multi dialect Arabic speech parallel corpora,” in *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on*, 2013, pp. 1–6.

[37] A. Alosaimy and E. Atwell, “Tagging Classical Arabic Text using Available Morphological Analysers and Part of Speech Taggers,” *J. Lang. Technol. Comput. Linguist.*, 2017.

[38] A. M. S. Alosaimy, “Ensemble Morphosyntactic Analyser for Classical Arabic,” Ph.D. thesis, University of Leeds, 2018.

[39] S. Alqrainy, “A morphological-syntactical analysis approach for Arabic textual tagging,” 2008.

[40] M. Alrabiah, A. Al-Salman, and E. S. Atwell, “The design and construction of the 50 million words KSUCCA,” in *Proceedings of WACL’2 Second Workshop on Arabic Corpus Linguistics*, 2013, pp. 5–8.

[41] M. Alrabiah, A. Al-Salman, and E. S. Atwell, “The design and construction of the 50 million words KSUCCA,” in *Proceedings of WACL’2 Second Workshop on Arabic Corpus Linguistics*, 2013, pp. 5–8.

[42] M. S. Alrabiah, “Building A Distributional Semantic Model for Traditional Arabic and Investigating its Novel Applications to The Holy Quran,” Ph.D. thesis, King Saud University, 2014.

[43] M. S. Alrabiah, “King Saud University Corpus of Classical Arabic (KSUCCA) | Maha Al-Rabiah – Blog.” [Online]. Available: https://mahaalrabiah.wordpress.com/2012/07/20/king-saud-university-corpus-of-classical-arabic-ksucca/. [Accessed: 22-Sep-2018].

[44] K. Alsmearat, M. Al-Ayyoub, and R. Al-Shalabi, “An extensive study of the bag-of-words approach for gender identification of arabic articles,” in *Computer Systems and Applications (AICCSA), 2014 IEEE/ACS 11th International Conference on*, 2014, pp. 601–608.

[45] M. Altamimi, O. Alruwaili, and W. J. Teahan, “BTAC: A Twitter Corpus for Arabic Dialect Identification,” in *Proceedings of the 6th Conference on Computer-Mediated Communication (CMC) and Social Media Corpora (CMC-corpora 2018)*, 2018, pp. 5–10.

[46] M. Altamimi and W. J. Teahan, “Gender and authorship categorisation of

Arabic text from Twitter using PPM," *Int. J. Comput. Sci. Inf. Technol*, vol. 9, pp. 131–140, 2017.

[47] A. Alwajeeh, M. Al-Ayyoub, and I. Hmeidi, "On authorship authentication of arabic articles," in *Information and Communication Systems (ICICS), 2014 5th International Conference on*, 2014, pp. 1–6.

[48] S. Atkins, J. Clear, and N. Ostler, "Corpus design criteria," *Lit. Linguist. Comput.*, vol. 7, no. 1, pp. 1–16, 1992.

[49] M. Attia, P. Pecina, A. Toral, L. Tounsi, and J. van Genabith, "A lexical database for modern standard Arabic interoperable with a finite state morphological transducer," in *International Workshop on Systems and Frameworks for Computational Morphology*, 2011, pp. 98–118.

[50] E. S. Atwell, G. Demetriou, J. Hughes, A. Schiffrin, C. Souter, and S. Wilcock, "A comparative evaluation of modern English corpus grammatical annotation schemes," *ICAME J. Int. Comput. Arch. Mod. Mediev. English J.*, vol. 24, pp. 7–23, 2000.

[51] BBC, "أخبار الشرق الأوسط - BBC Arabic." [Online]. Available: http://www.bbc.com/arabic/middleeast. [Accessed: 18-Mar-2017].

[52] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation," *Mach. Learn.*, vol. 34, no. 1–3, pp. 177–210, 1999.

[53] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990.

[54] Y. Benajiba, P. Rosso, and J. M. Benedíruiz, "Anersys: An arabic named entity recognition system based on maximum entropy," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2007, pp. 143–153.

[55] Y. Benajiba, P. Rosso, and J. M. G. Soriano, "Adapting the JIRS passage retrieval system to the Arabic language," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2007, pp. 530–541.

[56] I. Bensalem, P. Rosso, and S. Chikhi, "A new corpus for the evaluation of arabic intrinsic plagiarism detection," in *International Conference of the Cross-Language Evaluation Forum for European Languages*, 2013, pp. 53–58.

[57] M. S. Bin-Muqbil, “Phonetic And Phonological Aspects Of Arabic Emphatics And Gutturals,” Ph.D. thesis, The University Of Wisconsin-Madison, 2006.

[58] Blogs.transparent.com, “Arabic diacritics; importance | Arabic Language Blog.” [Online]. Available: https://blogs.transparent.com/arabic/arabic-diacritics-important-but-neglected/. [Accessed: 08-Mar-2019].

[59] T. Brants, “TnT: a statistical part-of-speech tagger,” in *Proceedings of the sixth conference on Applied natural language processing*, 2000, pp. 224–231.

[60] E. Brill, “A simple rule-based part of speech tagger,” in *Proceedings of the third conference on Applied natural language processing*, 1992, pp. 152–155.

[61] Britannica.com, “Arabic alphabet | Britannica.com.” [Online]. Available: https://www.britannica.com/topic/Arabic-alphabet. [Accessed: 08-Mar-2019].

[62] A. Z. Broder, “Identifying and filtering near-duplicate documents,” in *Annual Symposium on Combinatorial Pattern Matching*, 2000, pp. 1–10.

[63] P. F. Brown, P. V Desouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai, “Class-based n-gram models of natural language,” *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, 1992.

[64] P. F. Brown, V. J. Della Pietra, R. L. Mercer, S. A. Della Pietra, and J. C. Lai, “An estimate of an upper bound for the entropy of English,” *Comput. Linguist.*, vol. 18, no. 1, pp. 31–40, 1992.

[65] T. Buckwalter, “Buckwalter Arabic Transliteration.” [Online]. Available: http://www.qamus.org/transliteration.htm. [Accessed: 29-Jan-2019].

[66] Z. Chang, “A PPM-based Evaluation Method for Chinese-English Parallel Corpora in Machine Translation,” no. September, pp. 1–106, 2008.

[67] M. S. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002, pp. 380–388.

[68] N. Cheng, R. Chandramouli, and K. P. Subbalakshmi, “Author gender identification from text,” *Digit. Investig.*, vol. 8, no. 1, pp. 78–88, 2011.

[69] D. Chiang, M. Diab, N. Habash, O. Rambow, and S. Shareef, “Parsing Arabic dialects,” in *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

[70] G. G. Chowdhury, “Natural language processing,” *Annu. Rev. Inf. Sci. Technol.*, vol. 37, no. 1, pp. 51–89, 2003.

[71] J. G. Cleary, W. J. Teahan, and I. H. Witten, “Unbounded length contexts for PPM,” in *Data Compression Conference, 1995. DCC’95. Proceedings*, 1995, pp. 52–61.

[72] J. Cleary and I. Witten, “Data compression using adaptive coding and partial string matching,” vol. C, no. 4, pp. 396–402, 1984.

[73] J. Cohen, “A coefficient of agreement for nominal scales,” *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[74] Columbia University, “Arabic Language Disambiguation for Natural Language Processing Applications - cu14012 - Columbia Technology Ventures.” [Online]. Available: http://innovation.columbia.edu/technologies/cu14012_arabic-language-disambiguation-for-natural-language-processing-applications. [Accessed: 14-Dec-2018].

[75] P. Damien, N. Wakim, and M. Egea, “Phoneme-viseme mapping for Modern, Classical Arabic language,” in *ACTEA’09. International Conference on Advances in Computational Tools for Engineering Applications, 2009.*, 2009, pp. 547–552.

[76] K. Darwish *et al.*, “Multi-Dialect Arabic POS Tagging: A CRF Approach,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

[77] M. Dattia, “Arabic Wordlist for Spellchecking.” [Online]. Available: https://sourceforge.net/projects/arabic-wordlist/. [Accessed: 29-Sep-2018].

[78] U. Dependencies, “UD_Arabic-PADT.” [Online]. Available: https://universaldependencies.org/treebanks/ar_padt/index.html. [Accessed: 27-May-2019].

[79] U. Dependencies, “UD_Arabic-NYUAD.” [Online]. Available: https://universaldependencies.org/treebanks/ar_nyuad/index.html. [Accessed: 27-May-2019].

[80] M. Diab, K. Hacioglu, and D. Jurafsky, “Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks,” in *Proceedings of HLT-NAACL 2004: Short papers*, 2004, pp. 149–152.

[81] M. Diab, K. Hacioglu, and D. Jurafsky, “Automatic processing of modern standard Arabic text,” in *Arabic Computational Morphology*, Springer, 2007, pp. 159–179.

[82] M. T. Diab, “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*, 2007, pp. 89–96.

[83] J. Diederich, J. Kindermann, E. Leopold, and G. Paass, “Authorship attribution with support vector machines,” *Appl. Intell.*, vol. 19, no. 1–2, pp. 109–123, 2003.

[84] K. Dukes, “Statistical parsing by machine learning from a Classical Arabic treebank,” Ph.D. thesis, University of Leeds, 2013.

[85] K. Dukes and T. Buckwalter, “A dependency treebank of the Quran using traditional Arabic grammar,” in *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, 2010, pp. 1–7.

[86] K. Dukes and N. Habash, “Morphological Annotation of Quranic Arabic.,” in *LREC*, 2010.

[87] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, “Inductive learning algorithms and representations for text categorization,” in *Proceedings of the seventh international conference on Information and knowledge management*, 1998, pp. 148–155.

[88] M. EL-Haj, “Arabic in Business and Management Corpora (ABMC).” [Online]. Available: http://www.lancaster.ac.uk/staff/elhaj/corpora.htm. [Accessed: 27-Mar-2017].

[89] M. El-Haj, U. Kruschwitz, and C. Fox, “Creating language resources for under-resourced languages: methodologies, and experiments with Arabic,” *Lang. Resour. Eval.*, vol. 49, no. 3, pp. 549–580, 2015.

[90] S. El-Kareh and S. Al-Ansary, “An interactive multi-features POS tagger,” in *the Proceedings of the International Conference on Artificial and Computational Intelligence for Decision Control and Automation in Intelligence for Decision Control and Automation in Engineering and Industrial Applications*, 2000, p. 83Y88.

[91] G. Elbeheri, J. Everatt, G. Reid, and H. al Mannai, “Dyslexia assessment in

Arabic," *J. Res. Spec. Educ. Needs*, vol. 6, no. 3, pp. 143–152, 2006.

[92] Y. O. M. Elhadj, A. Abdelali, R. Bouziane, and A. H. Ammar, "Revisiting Arabic Part of Speech Tagsets," *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA*, vol. 2014, pp. 793–802, 2015.

[93] M. Elsheikh and E. Atwell, "Timeline of the development of Arabic PoS taggers and Morphological analysers," vol. 9, 2018.

[94] A. F. Emdad, M. Badamas, and S. Mouakket, "Factors and impacts of low utilization of Internet: The case of Arab countries," *J. Int. Technol. Inf. Manag.*, vol. 18, no. 3, p. 2, 2009.

[95] S. Eyheramendy, D. D. Lewis, and D. Madigan, "On the naive bayes model for text categorization," 2003.

[96] C. France, "Meedan's Open Source Arabic/English Translation Memory." [Online]. Available: https://github.com/anastaw/Meedan-Memory. [Accessed: 23-Sep-2018].

[97] W. N. Francis and H. Kučera, "The brown corpus: A standard corpus of present-day edited american english," *Provid. RI Dep. Linguist. Brown Univ. [producer Distrib.*, 1979.

[98] P. Gardner-Chloros, *Code-switching*. Cambridge University Press, 2009.

[99] Googleblog.blogspot.com, "Official Google Blog: Unicode nearing 50% of the web." [Online]. Available: https://googleblog.blogspot.com/2010/01/unicode-nearing-50-of-web.html. [Accessed: 30-Mar-2019].

[100] S. Green and C. Manning, "Better Arabic parsing: Baselines, evaluations, and analysis," *COLING '10 Proc. 23rd Int. Conf. Comput. Linguist.*, no. August, pp. 394–402, 2010.

[101] S. Green, M.-C. de Marneffe, and C. D. Manning, "Parsing Models for Identifying Multiword Expressions," *Comput. Linguist.*, vol. 39, no. 1, pp. 195–227, Mar. 2013.

[102] B. B. Greene and G. M. Rubin, "Automated grammatical tagging of English," 1971.

[103] N. Habash and O. Rambow, "Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005, pp. 573–580.

[104] N. Habash, O. Rambow, and R. Roth, “MADA+ TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization,” in *Proceedings of the 2nd international conference on Arabic language resources and tools (MEDAR), Cairo, Egypt*, 2009, pp. 102–109.

[105] N. Habash and R. M. Roth, “Catib: The columbia arabic treebank,” in *Proceedings of the ACL-IJCNLP 2009 conference short papers*, 2009, pp. 221–224.

[106] N. Habash, R. Roth, O. Rambow, R. Eskander, and N. Tomeh, “Morphological Analysis and Disambiguation for Dialectal Arabic.,” in *Hlt-Naacl*, 2013, pp. 426–432.

[107] N. Habash and F. Sadat, “Arabic preprocessing schemes for statistical machine translation,” in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, 2006, pp. 49–52.

[108] Y. El Hadj, I. Al-Sughayeir, and A. Al-Ansari, “Arabic part-of-speech tagging using the sentence structure,” in *Proceedings of the Second International Conference on Arabic Language Resources and Tools, Cairo, Egypt*, 2009.

[109] M. Hadni, S. A. Ouatik, A. Lachkar, and M. Meknassi, “Hybrid part-of-speech tagger for non-vocalized Arabic text,” *Int. J. Nat. Lang. Comput. Vol*, vol. 2, 2013.

[110] J. Hajič *et al.*, “Prague Arabic dependency treebank 1.0,” 2009.

[111] J. Hajič, O. Smrz, P. Zemánek, J. Šnaidauf, and E. Beška, “Prague Arabic dependency treebank: Development in data and tools,” in *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, 2004, pp. 110–117.

[112] J. A. Haywood, H. M. Nahmad, and G. W. Thatcher, *A new Arabic grammar of the written language*. Lund Humphries London, 1965.

[113] J. He, A. H. Tan, and C. L. Tan, “On machine learning methods for Chinese document categorization,” *Appl. Intell.*, vol. 18, no. 3, pp. 311–322, 2003.

[114] M. A. Hearst, “Multi-paragraph segmentation of expository text,” in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 1994, pp. 9–16.

[115] Internet World Stats, “Middle East Internet Usage Stats and Facebook Statistics,” *Internet World Stats*, 2014. [Online]. Available: http://www.internetworldstats.com/middle.htm. [Accessed: 04-Oct-2016].

[116] internetworldstats.com, “Arabic Speaking Internet Users and Population Statistics.” [Online]. Available: https://www.internetworldstats.com/stats19.htm#arabic. [Accessed: 27-Mar-2019].

[117] R. R. Jablonkai and N. Čebron, “Corpora as Tools for Self-Driven Learning: A Corpus-Based ESP Course,” in *Student-Driven Learning Strategies for the 21st Century Classroom*, IGI Global, 2017, pp. 274–298.

[118] S. Jeblee, H. Bouamor, W. Zaghouani, and K. Oflazer, “CMUQ@QALB-2014: An SMT-based System for Automatic Arabic Error Correction,” in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, 2014, pp. 137–142.

[119] F. Jelinek, “Self-organized language modeling for speech recognition,” *Readings speech Recognit.*, pp. 450–506, 1990.

[120] T. Joachims, *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers, 2002.

[121] P. Juola and others, “Authorship attribution,” *Found. Trends{®} Inf. Retr.*, vol. 1, no. 3, pp. 233–334, 2008.

[122] A. El Kah, I. Zeroual, and A. Lakhouaja, “Application of Arabic language processing in language learning,” in *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*, 2017, p. 35.

[123] S. Katz, “Estimation of probabilities from sparse data for the language model component of a speech recognizer,” *IEEE Trans. Acoust.*, vol. 35, no. 3, pp. 400–401, 1987.

[124] D. V Khmelev and W. J. Teahan, “A repetition based measure for verification of text collections and for text categorization,” in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, 2003, pp. 104–110.

[125] S. Khoja, “APT: An automatic arabic part-of-speech tagger,” Ph.D. thesis, Lancaster University, 2003.

[126] S. Khoja, “APT: Arabic part-of-speech tagger,” in *Proceedings of the Student*

Workshop at NAACL, 2001, pp. 20–25.

[127] S. Khoja, R. Garside, and G. Knowles, “A tagset for the morphosyntactic tagging of Arabic,” *Proc. Corpus Linguist. Lancaster Univ.*, vol. 13, 2001.

[128] S. Klein and R. F. Simmons, “A computational approach to grammatical coding of English words,” *J. ACM*, vol. 10, no. 3, pp. 334–347, 1963.

[129] V. Korde and C. N. Mahender, “Text classification and classifiers: A survey,” *Int. J. Artif. Intell. Appl.*, vol. 3, no. 2, p. 85, 2012.

[130] H. Kozima, “Text segmentation based on similarity between words,” in *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, 1993, pp. 286–288.

[131] R. Kuhn and R. De Mori, “A cache-based natural language model for speech recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 6, pp. 570–583, 1990.

[132] W. Lam and Y. Han, “Automatic textual document categorization based on generalized instance sets and a metamodel,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 628–633, 2003.

[133] J. R. Landis and G. G. Koch, “The measurement of observer agreement for categorical data,” *Biometrics*, pp. 159–174, 1977.

[134] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet physics doklady*, 1966, vol. 10, no. 8, pp. 707–710.

[135] M. P. Lewis, G. F. Simons, and C. D. Fennig, *Ethnologue: Languages of the world*, vol. 16. SIL international Dallas, TX, 2009.

[136] T. Li, S. Zhu, and M. Ogihara, “Efficient multi-way text categorization via generalized discriminant analysis,” in *Proceedings of the twelfth international conference on Information and knowledge management*, 2003, pp. 317–324.

[137] libya-unesco.org, “المدونات العربية وتعليم العربية لغة ثانية.” [Online]. Available: http://www.libya-unesco.org/dyn/wp-content/uploads/2018/04/AlTawasol-Allisany-N19.pdf. [Accessed: 30-May-2019].

[138] Linguistic Data Consortium., *Buckwalter Arabic morphological analyzer : Version 1.0*. Linguistic Data Consortium, 2002.

[139] S. Al Maadeed, W. Ayouby, A. Hassaine, and J. M. Aljaam, “QUWI: an Arabic and English handwriting dataset for offline writer identification,” in *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference*

on, 2012, pp. 746–751.

[140] M. Maamouri and A. Bies, “Developing an Arabic treebank: Methods, guidelines, procedures, and tools,” in *Proceedings of the Workshop on Computational Approaches to Arabic Script-based languages*, 2004, pp. 2–9.

[141] M. Maamouri, A. Bies, T. Buckwalter, H. Jin, and W. Mekki, “Arabic Treebank: Part 3 (full corpus) v 2.0 (MPG + Syntactic Analysis),” *LDC2005T20*, 2005. [Online]. Available: https://catalog.ldc.upenn.edu/LDC2005T20. [Accessed: 25-Nov-2016].

[142] M. Maamouri, A. Bies, and S. Kulick, “Enhancing the Arabic Treebank: a Collaborative Effort toward New Annotation Guidelines.,” in *LREC*, 2008, pp. 3–192.

[143] S. Malmasi, E. Refaee, and M. Dras, “Arabic dialect identification using a parallel multidialectal corpus,” in *International Conference of the Pacific Association for Computational Linguistics*, 2015, pp. 35–53.

[144] A. R. Martinez, “Part-of-speech tagging,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 4, no. 1, pp. 107–113, 2012.

[145] M. L. McHugh, “Interrater reliability: the kappa statistic,” *Biochem. medica Biochem. medica*, vol. 22, no. 3, pp. 276–282, 2012.

[146] C. F. Meyer, *English corpus linguistics: An introduction*. Cambridge University Press, 2002.

[147] R. Mitkov, *The Oxford handbook of computational linguistics*. Oxford University Press, 2005.

[148] A. Moffat, “Implementing the PPM data compression scheme,” *IEEE Trans. Commun.*, vol. 38, no. 11, pp. 1917–1921, 1990.

[149] A. Moffat, R. M. Neal, and I. H. Witten, “Arithmetic coding revisited,” *ACM Trans. Inf. Syst.*, vol. 16, no. 3, pp. 256–294, 1998.

[150] E. Mohamed and S. Kübler, “Arabic Part of Speech Tagging.,” in *LREC*, 2010.

[151] B. Mohit, “Named Entity Recognition,” in *Natural Language Processing of Semitic Languages*, I. Zitouni, Ed. Springer, USA, 2014.

[152] B. Mohit, “AQMAR Arabic Dependency Corpus.” [Online]. Available: http://www.cs.cmu.edu/~ark/ArabicDeps/. [Accessed: 18-Sep-2018].

[153] B. Mohit, “BRAT Rapid Annotation Tool.” [Online]. Available:

http://brat.nlplab.org/. [Accessed: 16-Sep-2018].

[154] T. Morton, J. Kottmann, J. Baldridge, and G. Bierner, “Opennlp: A java-based nlp toolkit,” 2005.

[155] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, “Natural language processing: an introduction,” *J. Am. Med. Informatics Assoc.*, vol. 18, no. 5, pp. 544–551, 2011.

[156] M. M. Najeeb, A. A. Abdelkader, and M. B. Al-Zghoul, “Arabic natural language processing laboratory serving Islamic sciences,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 3, 2014.

[157] D. Q. Nguyen, D. Q. Nguyen, D. D. Pham, and S. B. Pham, “RDRPOSTagger: A ripple down rules-based part-of-speech tagger,” in *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 2014, pp. 17–20.

[158] nltk.org, “Simple Pipeline Architecture for an Information Extraction System.” [Online]. Available: http://www.nltk.org/book/ch07.html. [Accessed: 08-Feb-2019].

[159] P. Pajas and J. Štěpánek, “Recent advances in a feature-rich framework for treebank annotation,” in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, 2008, pp. 673–680.

[160] R. Parker and Linguistic Data Consortium., *Arabic gigaword fifth edition*. Linguistic Data Consortium, 2011.

[161] A. Pasha *et al.*, “MADAMIRA : A Fast , Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic,” *Proc. 9th Lang. Resour. Eval. Conf.*, vol. 14, pp. 1094–1101, 2014.

[162] A. Pasha *et al.*, “Madamira Arabic Analyzer - Online.” [Online]. Available: https://camel.abudhabi.nyu.edu/madamira/. [Accessed: 17-Feb-2019].

[163] F. Peng, D. Schuurmans, and S. Wang, “Augmenting Naive Bayes classifiers with statistical language models,” *Inf. Retr. Boston.*, vol. 7, no. 3–4, pp. 317–345, 2004.

[164] Perseus.tufts.edu/hopper/opensource/download, “Arabic-English Learner’s Dictionary.” [Online]. Available: http://www.perseus.tufts.edu/hopper/opensource/download. [Accessed: 22-Sep-2018].

[165] P. Petr and S. Jan, “TrEd User’s Manual.” [Online]. Available: https://ufal.mff.cuni.cz/tred/documentation/tred.html. [Accessed: 20-Sep-2018].

[166] S. Pradhan and L. Ramshaw, “OntoNotes: Large Scale Multi-Layer, Multi-Lingual, Distributed Annotation,” in *Handbook of Linguistic Annotation*, Springer, 2017, pp. 521–554.

[167] W. Pugh and M. H. Henzinger, “Detecting duplicate and near-duplicate files.” Google Patents, 2008.

[168] D. Reidsma and J. Carletta, “Reliability measurement without limits,” *Comput. Linguist.*, vol. 34, no. 3, pp. 319–326, 2008.

[169] J. C. Reynar, “An automatic method of finding topic boundaries,” in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 1994, pp. 331–333.

[170] D. Richards, “Two decades of ripple down rules research,” *Knowl. Eng. Rev.*, vol. 24, no. 2, pp. 159–184, 2009.

[171] J. Rissanen and G. G. Langdon, “Arithmetic coding,” *IBM J. Res. Dev.*, vol. 23, no. 2, pp. 149–162, 1979.

[172] M. E. Ruiz and P. Srinivasan, “Hierarchical neural networks for text categorization (poster abstract),” in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 281–282.

[173] K. C. Ryding, *A reference grammar of modern standard Arabic*. Cambridge University Press, 2005.

[174] M. Sawalha and E. S. Atwell, “Linguistically informed and corpus informed morphological analysis of Arabic,” in *Proceedings of the 5th Corpus Linguistics Conference*, 2009.

[175] M. Sawalha and E. S. Atwell, “Fine-grain morphological analyzer and part-of-speech tagger for Arabic text,” in *proceedings of the Seventh conference on International Language Resources and Evaluation (LREC’10)*, 2010, pp. 1258–1265.

[176] M. Sawalha and E. Atwell, “A standard tag set expounding traditional morphological features for Arabic language part-of-speech tagging,” *Word Struct.*, vol. 6, no. 1, pp. 43–99, 2013.

[177] H. Schendl and L. Wright, *Code-switching in early English*, vol. 76. Walter de Gruyter, 2012.

[178] N. Schneider, B. Mohit, K. Oflazer, and N. A. Smith, “Coarse lexical semantic annotation with supersenses: an Arabic case study,” in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, 2012, pp. 253–258.

[179] F. Sebastiani, “Machine learning in automated text categorization,” *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.

[180] F. Al Shamsi and A. Guessoum, “A hidden Markov model-based POS tagger for Arabic,” in *Proceeding of the 8th International Conference on the Statistical Analysis of Textual Data, France*, 2006, pp. 31–42.

[181] A. El Shamsy, “Al-Shāfi’ī’s Written Corpus: A Source-Critical Study,” *J. Am. Orient. Soc.*, vol. 132, no. 2, pp. 199–220, 2012.

[182] C. E. Shannon, “A Mathematical Theory of Communication,” *Bell Syst. Tech. J.*, vol. 27, pp. 623--656, 1948.

[183] N. Shivakumar and H. Garcia-Molina, “Finding near-replicas of documents on the web,” in *International Workshop on the World Wide Web and Databases*, 1998, pp. 204–212.

[184] Skynewsarabia, “سكاي نيوز عربية | أخبار اليوم.” [Online]. Available: http://www.skynewsarabia.com/web/home. [Accessed: 23-Mar-2017].

[185] A. Soudi, A. Farghaly, G. Neumann, and R. Zbib, *Challenges for Arabic machine translation*, vol. 9. John Benjamins Publishing, 2012.

[186] E. Stamatatos, “A survey of modern authorship attribution methods,” *J. Am. Soc. Inf. Sci. Technol.*, vol. 60, no. 3, pp. 538–556, 2009.

[187] R. Steinberger, B. Pouliquen, M. Kabadjov, and E. der Goot, “JRC-Names: A freely available, highly multilingual named entity resource,” *CoRR*, vol. abs/1309.6, 2013.

[188] H. Ta’amneh, E. A. Keshek, M. B. Issa, M. Al-Ayyoub, and Y. Jararweh, “Compression-based arabic text classification,” in *Computer Systems and Applications (AICCSA), 2014 IEEE/ACS 11th International Conference on*, 2014, pp. 594–600.

[189] A. Taylor, M. Marcus, and B. Santorini, “The Penn treebank: an overview,” in *Treebanks*, Springer, 2003, pp. 5–22.

[190] W. J. Teahan and J. G. Cleary, “Tag Based Models of English Text.,” in *Data Compression Conference*, 1998, pp. 43–52.

[191] W. J. Teahan and D. J. Harper, “Combining PPM models using a text mining approach,” in *Data Compression Conference, 2001. Proceedings. DCC 2001.*, 2001, pp. 153–162.

[192] W. Teahan, “A Compression-Based Toolkit for Modelling and Processing Natural Language Text,” *Information*, vol. 9, no. 12, p. 294, 2018.

[193] W. J. Teahan and J. G. Cleary, “Applying compression to natural language processing,” in *SPAE: The Corpus of Spoken Professional American-English.*, 1997.

[194] W. J. Teahan and D. J. Harper, “Using compression-based language models for text categorization,” in *Language modeling for information retrieval*, Springer, 2003, pp. 141–165.

[195] W. J. Teahan, Y. Wen, R. McNab, and I. H. Witten, “A compression-based algorithm for Chinese word segmentation,” *Comput. Linguist.*, vol. 26, no. 3, pp. 375–393, 2000.

[196] W. J. Teahan, “Modelling English text,” Ph.D. thesis, Waikato University, 1998.

[197] W. J. Teahan, “Text classification and segmentation using minimum cross-entropy,” in *Content-Based Multimedia Information Access-Volume 2*, 2000, pp. 943–961.

[198] W. J. Teahan, S. Inglis, J. G. Cleary, and G. Holmes, “Correcting English text using PPM models,” in *Data Compression Conference, 1998. DCC’98. Proceedings*, 1998, pp. 289–298.

[199] K. Toutanova and C. Manning, “The Stanford Natural Language Processing Group.” [Online]. Available: https://nlp.stanford.edu/software/tagger.shtml. [Accessed: 17-Feb-2019].

[200] K. Toutanova and C. D. Manning, “Enriching the knowledge sources used in a maximum entropy part-of-speech tagger,” in *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, 2000, pp. 63–70.

[201] J. J. Tsay and J. D. Wang, “Improving linear classifier for Chinese text

categorization," *Inf. Process. Manag.*, vol. 40, no. 2, pp. 223–237, 2004.

[202] UN, "UN Corpus(Arabic portion)." [Online]. Available: http://www.sibawayh-nlp.com/?q=node/1158. [Accessed: 23-Sep-2018].

[203] C. Unicode Staff, *The Unicode standard: worldwide character encoding*. Addison-Wesley Longman Publishing Co., Inc., 1991.

[204] E. Weekley, *An etymological dictionary of modern English*. Courier Corporation, 2012.

[205] Y. Wen, I. H. Witten, and D. Wang, "Token identification using HMM and PPM models," in *Australasian Joint Conference on Artificial Intelligence*, 2003, pp. 173–185.

[206] S. Wintner, "Morphological processing of Semitic languages," in *Natural language processing of Semitic languages*, Springer, 2014, pp. 43–66.

[207] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[208] P. Wu, "Adaptive models of Chinese text," Ph.D. thesis, Bangor University, 2007.

[209] M. Wynne, "Archiving, distribution and preservation," *Dev. Linguist. corpora A Guid. to good Pract.*, pp. 71–78, 2005.

[210] R. Z. Xiao, "Theory-driven corpus research: using corpora to inform aspect theory," *Lüdeling, A., Kytö, M. Corpus Linguist. An Int. Handb.*, vol. 2, pp. 987–1008, 2008.

[211] W. Zaghouani, "Critical survey of the freely available Arabic corpora," in *In LREC'14 Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools (OSACT)*, 2017, pp. 1–8.

[212] O. F. Zaidan and C. Callison-Burch, "The arabic online commentary dataset: an annotated dataset of informal arabic with high dialectal content," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, 2011, pp. 37–41.

[213] T. Zerrouki and A. Balla, "Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems," *Data Br.*, vol. 11, pp. 147–151, Apr. 2017.

[214] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate

coding," *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[215] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. theory*, vol. 23, no. 3, pp. 337–343, 1977.