University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Ahmed H. Aliwy

Arabic Morphosyntactic Raw Text Part of Speech Tagging System

PhD dissertation

Supervisor
Prof. dr hab. Jerzy Tyszkiewicz
Institute of Informatics
University of Warsaw

January 2013

Author's declaration:

Aware of legal liability I hereby declare that I have written this dissertation my self and all the contents of the dissertation have been obtained by legal means.

January 14, 2013 ……

date *Ahmed H. Aliwy*

Supervisor's declaration:

The dissertation is ready to be reviewed

January 14, 2013 ……

date *Prof. dr hab. Jerzy Tyszkiewicz, UW*

Abstract

We present a comprehensive Arabic tagging system: from the raw text to tagging disambiguation. For each processing step in the tagging system, we analyze the existing solutions (if any) and use one of them or propose, implement and evaluate a new one.

This work began with designing a new Arabic tagset suitable for Classical Arabic (CA) and Modern Standard Arabic (MSA). In addition to the classical constructions in tag systems, we introduce interleaving of tags. Interleaving is a relation between tags which, in certain situations, can be attached to the same occurrence of a word, but each of them can also appear alone. Our tagset makes this relation explicit.

Then we deal with the preparatory stages for tagging system. The first initial stage is tokenization and segmentation. We use rule-based and statistical methods for this task. The second stage is analyzing and extracting the lemma from the word. We have created our own analyzer compatible with our requirements. Its main part is a dictionary which provides features, POS and lemma for each word.

The last part of our work is the tagging algorithm which produces one tag for each word. We use a hybrid method by combining rules-based and statistical methods. Three taggers, Hidden Markov Model (HMM), maximum match and Brill are combined by a new method, which we call master and slaves. Then handwritten rule-based tagger is also added to master-slaves. The rule based tagger eliminates incorrect tags, and the master chooses the best one among the remaining ones, assisted by the other slaves.

Our complete system is ready to be used for annotation of Arabic corpora.

Key words

Arabic tagset, Arabic tagger, Arabic Tokenization, Arabic segmentation, Arabic lemmatization, master-slaves tagging, tagset interleaving, handwritten rules-based tagger.

Published papers:

Chapter 3:

Ahmed H. Aliwy (2013): **Comparing Arabic tagsets and Designing a New One.** to appear in: Lingwistyka Stosowana "Applied Linguistics" nr (7) 2013, University of Warsaw, Poland.

Chapter 4:

Ahmed H. Aliwy (2012): **Tokenization as Preprocessing for Arabic Tagging System** . In proceeding of International Conference on Knowledge and Education Technology (ICKET 2012), Paris, 2012. Published in International Journal of Information and Education Technology Vol.2(4): 348-353.

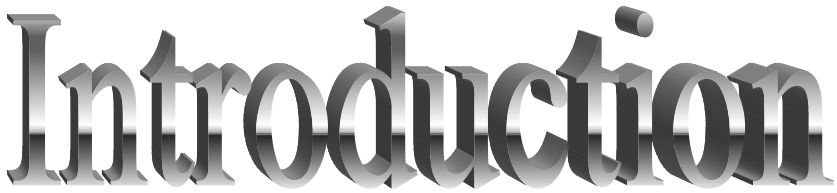
Chapter 5:

Ahmed H. Aliwy (2012): **Arabic Language Analyzer with Lemma Extraction and Rich Tagset**. In proceeding of JapTAL 2012, Japan. Lecture Notes in Computer Science vol. 7614, pp. 168-179.

Table of contents

Table of figures

Chapter 1



1.1 Introduction

The topic of this dissertation is morphosyntactic part of speech tagging (abbreviated POS tagging) for Arabic.

This topic has long and rich history for other languages, mainly for English.

POS tagging provides fundamental information about word forms used in sentences of natural language. The method of utilizing this information varies depending on the particular NLP application (information retrieval, machine translation …), in which it is used.

Tagging is a source of many challenges for researchers. These challenges depend very much on the language under consideration. In this dissertation we consider Arabic, a highly inflected language. Although Arabic language is generally quite regular and there are very few irregular forms, very rich and complicated structure of inflection, which in many cases changes the structure of the words, causes high degree of complexity of tagging. The other hard problem is the lack of Arabic language resources, corpora and other tools. We propose a new tagset in this dissertation and in this case the scarcity of resources makes the work

much more difficult. Tokenization schemes[1] are also a source of problems in tagging.

We can distinguish, in our dissertation, online and offline tagging. In both of them, the problem to be solved is the same, but the trade-off between quality of tagging and the speed of the process is different.

Online tagging is typically a part of another application, like machine translation. The speed in this scenario is very important, even at the price of somewhat decreased accuracy.

Offline tagging can be considered as an independent task, like annotating a corpus. The accuracy is in this case the crucial factor with much less emphasis on speed.

In this dissertation we have offline tagging in mind, hence we aim mainly at increasing accuracy of the process and the quality of information it provides, and generally disregard efficiency questions.

1.2 The overview of the dissertation

When we work on tagging, in the first place we have to choose a right tagset to be used. This choice affects the amount of information about forms of words generated in the process of tagging. One can use an existing tagset or decide to develop a new one. In this dissertation we present a new tagset, which improves on the existing ones.

POS tagging, similarly to other NLP tasks, needs a number of preprocessing stages. Most of these stages can be considered as separated tasks. We list here all the stages in our work. Some of these stages are optional in other works. For example, the analyzer misses in most of resent Arabic POS tagging techniques.

The first one is tokenization and segmentation, i.e., splitting the running text into tokens. This procedure can be split into several steps:

1. Normalization: unification of variants of letters, deleting Tatweel and the like.
2. Sentence segmentation: splitting running text into sentences

[1] See (Habash) [45] for more information on tokenization schema. Also, see (Benajiba & Zitouni) [19] for schema levels.

3. Word segmentation: splitting sentences into words.
4. Word tokenization: splitting words into morphemes.

Many other NLP tasks need this preprocessing, too. In our dissertation, this preprocessing is a separate task, and therefore our algorithms can be used independently of the tagging procedure.

The second level of preprocessing is analysis with lemma extraction, which extracts the lemma of each word, determines the part of speech and features for it. In many other approaches it is the task of a morphological analyzer to extract the root or stem of the word rather than the lemma. Extracting lemma for Arabic received little attention in the literature so far because it was considered to be a hard problem.

After these two preprocessing steps, the tagging will be achieved by applying one of the supervised or unsupervised techniques to disambiguate the results of the previous steps. Figure 1-1 shows the whole system which tries to solve all the tasks described in this introduction.

1.3 Related work

Our complete system has few counterparts in the literature, because it is a whole tagging system, and most of the existing papers deal with isolated fragments of the complete process. Therefore we will list the works which relate, partially or completely, to our work.

1.3.1 Tagset related works

Tagsets are intimately connected with taggers which use them and are generally not discussed as standalone objects. (Khoja tagset [57][59]; Al-Qrainy tagset [9], Sawalha tagset [82], Alhadj tagset [38], Buckwalter tagset, Reduced Buckwalter tagset (Bies tagset, Kulick tagset [65] and Extended Reduced tagset) [45], KATIB POS tagset [47][49] and PADT tagset [45]) are the most well-known Arabic tagsets. We discuss them and their limitations in Chapter 3. Our main goal in designing a new one was to cover specific elements of Arabic missing in those tagsets and eliminating unwanted tags. The other goal is for producing a tagset compatible with Classical Arabic (CA) and Modern Standard Arabic (MSA). See chapter 3 for more details.

We constructed a new tagset by avoiding the limitations of the above mentioned tasets. It was constructed depending on the Arabic literature and it is not derived from tagsets dedicated for other languages. Our tagset does not have interleaving, even though it has many tags. Interleaving is a novel notion introduced by us. It is likely to occur in highly inflected language with a huge tagset.

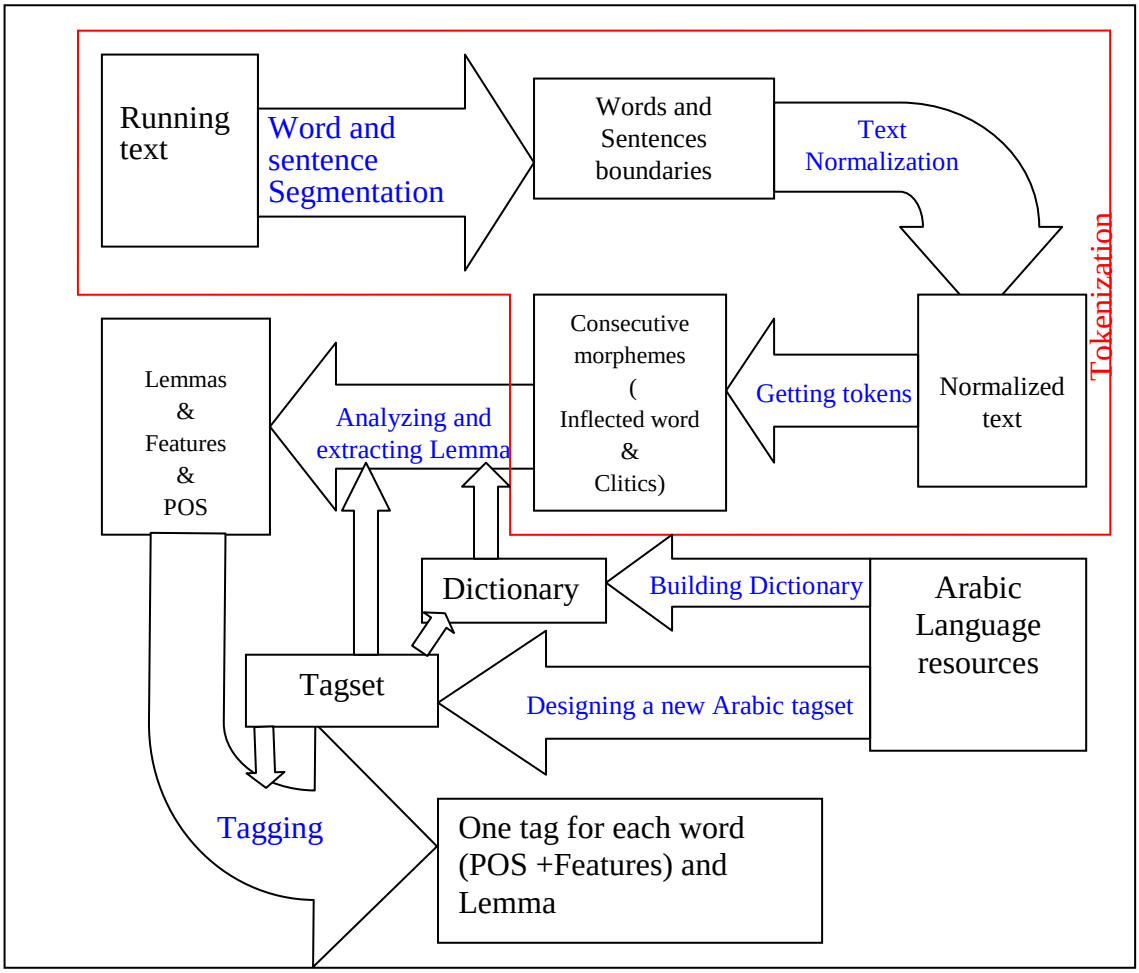


Figure 1-1: The overview of the system

1.3.2 Tokenization related works

Tokenization or segmentation procedures are fragments of the following tools: (MADA+TOKEN (Habash) [51], Buckwalter Arabic Morphological Analyzer BAMA (Buckwalter) [26][25], AMIRA (Mona Diab) [32], Xerox Arabic Morphological Analyzer and generator (Beesley) [17][18], Sakhr's Arabic Morphological Analyzer (Sakhr Software) [81], Khoja's stemmer (Khoja) [56] and almost morphological Analyzers) .

(Benajiba) [20] presents two segmentation schemes that are morphological segmentation and Arabic TreeBank segmentation and he shows their impact on an important natural language processing task: mention detection. Experiments on Arabic TreeBank corpus show 98.1% accuracy on morphological segmentation. He did not consider tokenization.

The approach of (Lee) [66] models the word as prefix*-stem-suffix*. The algorithm uses a trigram language model to determine the most probable morpheme sequence for a given input. The language model is initially estimated from a small manually segmented corpus of about 110,000 words. The resulting Arabic word segmentation system achieves around 97% exact match accuracy on a test corpus containing 29k words.

The systems of Benajiba and Lee deals with stem rather than lemma. According to (Habash) [45] stem need not be a legal Arabic word form, unlike lemma. See Chapter 4 for more details.

Our Arabic tokenizer is constructed using a hybrid unsupervised method, and is a stand-alone application. It produces all possible tokenizations for each word. Then, written rules and statistical methods are applied to solve the ambiguities. Its output is one tokenization for each word. The deleted and changed letters are retrieved by the tokenizer.

1.3.3 Analyzing and extracting lemma related works

In case of extracting lemma, (El-Shishtawy & El-Ghannam) [39] do lemmatization in three phases: analyzing, POS tagging and then lemma generation. This approach was proposed for information retrieval.

Concerning morphological analyzers, there are many works in this field. MAGEAD (Habash et, al.) [50] provides an analysis for a root+pattern. Darwish analyzer (Darwish) [31] was only concerned with generating the possible roots of a given Arabic word. (Gridach-Chenfour) [44] Their approach is based on Arabic morphological automaton technology. (Elixir-FM) [88] is a functional morphology systems which models templatic morphology and orthographic rules. BAMA Buckwalter [26] is based on a lexicon which has morphotactic and orthographic rules encoded inside it. See Chapter 5 for more details.

All of the above mentioned analyzers didn't meet our requirements, which prompted us to build a new one, because we wanted POS and features to be described by a new very rich tagset. It differs from most of the existing analyzers because it produces a lemma rather than stem or root, which is a significantly harder task in Arabic.

1.3.4 Tagging related works

(Diab et,al. & Diab) [33][32] used suppor vector machines (SVM) for tagging in her papers. (Habash & Rambow) [46] used SVM with a morphological analyzer, APT (Khoja) [57] used statistical and rule-based methods, AL-Shamsi and Guessoum [11] used HMM, (Freeman) [42] used Brill (Transformation) tagging, (AlGahtani et, al.) [5] used Brill (Transformation) with morphological analyzer, (Tlili-Guiassa) [89] used rules-based and memory-based methods, (Seth Kulick) [64] used classifier with regular expressions, (Van den Bosch) [23] used memory-based learning, (Mohamed and Kübler) [71] used statistical, (Selçuk) [62] used HMM without morphological analyzer or lexicon, (El Hadj et, al.) [36] used HMM with morphological analyzer, (Mansour et, al.) [69] used HMM with morphological analyzer with lexicon. All these Arabic taggers are summarized in Chapter 6.

1.3.5 Combining taggers related work

In the paper of (Glass & Bangay) [43] a few taggers are grouped to form a voting system, but in no case the combined results improve on the individual accuracies. (Yonghui et, al.) [92] presents a novel data fusion strategy in POS tagging - correlation voting. They proved that the correlative voting is better than other fusion methods. The paper (Henrich et, al.) [52] provides an algorithms for simple and weighted voting. It improved the accuracy by 1.26 – 1.58 % over the best method among its individual component taggers. The authors of (Loftsson) [67] used many combinations of several taggers in a simple voting approach using three taggers which are TBL, TNT and Ice. Taggers are described in Chapters 7 & 8 in more detail.

We used a new method for combining taggers which we call master-slaves. We also we used a rule-based tagger, with manually encoded rules, as a special slave.

1.3.6 Works related to the complete system

APT by (Khoja) [57] used Statistical and rule-based methods for tagging. Her tagset will be discussed in chapter 3. Her work did not have lemmatizer or tokenizer but she had her own stemmer. The statistical method was trained using a corpus of 50,000 words in Modern Standard Arabic (an extract from the Saudi Al-Jazirah newspaper). A lexicon derived from this corpus was used in this tagger.

MADA+TOKEN (Habash) [45] where MADA (Morphological Analysis and Disambiguation for Arabic) is a utility that, given raw Arabic text, adds as much lexical and morphological information as possible by disambiguating, in one operation, part-of-speech tags, lexemes, diacritizations and full morphological analyses. TOKEN is a general tokenizer for Arabic.

AMIRA (Diab) [32] is a successor suite to the ASVMTools (Diab et al.) [34]. The AMIRA toolkit includes a clitic tokenizer (TOK), part of speech tagger (POS) and base phrase chunker (BPC) - shallow syntactic parser. The accuracy of the ERTS (Extended Reduced TagSet) tagger is 96.13% and the accuracy of the RTS (Reduced TagSet) tagger is 96.15%.

The last two works are toolkits for Arabic language. They are composed from many research tools.

(Kulick) [64] describes an approach to simultaneous tokenization and part-of-speech tagging that is based on separating the closed and open-class items, and focusing on the likelihood of the possible stems of the open class words. He used regular expressions with a reduced tag set. The data set was Arabic Treebank (ATB3-v3.2) and the accuracy of tagging was 95.147%.

For more Arabic taggers see chapter 6.

1.4 Dissertation outline

The rest of our dissertation is constructed as follows:

Chapter two is a brief introduction to Arabic language; some of the details are described in later chapters, when they are needed.

Chapter three describes almost all Arabic tagsets with their limits and specifications and presents the design of a new one.

Chapter four is concerned with the first preprocessing task: normalization, tokenization and segmentation.

Chapter five is concerned with the next preprocessing task: lemmatization and analyzing, the relation between lemmatization and morphological analyzer.

Chapter six surveys the main tagging techniques which are used in general and in particular for Arabic language.

Chapter seven describes master-slave technique for combining taggers. It is implemented and tested on English and Arabic corpora.

Chapter eight describes our implementation of adding handwritten rule-based tagger to the master-slaves technique.

Chapter nine is the discussion of the results and future work.

Chapter 2

Introduction to Arabic language

2.1 Introduction

Arabic (العربية *al-arabiyyah*) is a name applied to a group of dialects of the Central Semitic languages, thus related to and classified alongside other Semitic languages such as Hebrew and the Neo-Aramaic languages. Spoken Arabic varieties have more speakers than any other language in the Semitic language family. Arabic is the official language of 22 countries and it is the liturgical language of Islam since it is the language of the Qur'an, the Islamic Holy Book. It is the sixth official language in United Nations. It is written from right to left and the letters of each word are attached together. The words are split by spaces. The punctuation is used for specifying sentences, paragraphs and other specification of written text like.

The history of Arabic language is not exactly known but the grammars of Arabic language were begun before 1400 years ago.

2.2 Arabic letters

The Arabic formal word, in Classical Arabic (CA), is constructed from letters and diacritics. The diacritics are optional in Modern Standard Arabic (MSA) but, in general, are neglected. There are 28 letters, three of them are vowels. Appendix

A1 shows Unicode for Arabic letters. Figure 2-1 shows Arabic character. The italic letters are vowels. The underlined letters are not attached to the succeeding letter in the word. The letter Taa can be written as "p"-"ة"or "ـة" in some cases. There is a letter "|"-"آ" which represents two letters "O"-"أ"and "A"-"ا" i.e. أ+ا ≡ آ.

The diacritics are special symbols used to solve ambiguity in word spelling and meaning. It was shown in figure 2-2.

The Arabic numbers are shown in figure 2-3. Writing Arabic number follows the same rules as in English, i.e. they are written and read from left to right.

Transli-teration	**letter**	first	Middle	Last	Transli-teration	**letter**	first	Middle	Last
A	***Alef***	ا	ـا	ـا	D	Dhad	ضـ	ـضـ	ـض ، ض
O, I , {, W, '	Hamza	أ، إ, آ	ـئـ ، ـؤ ، ـأ	ـئ، ـؤ ، ـأ ، ـإ ، ـء	T	Daa	طـ	ـطـ	ـط ، ط
b	Baa	بـ	ـبـ	ـب ، ب	Z	Dhaa	ظـ	ـظـ	ـظ ، ظ
t	Taa	تـ	ـتـ	ـت ، ت	E	Ain	عـ	ـعـ	ـع ، ع
v	Thaa	ثـ	ـثـ	ـث ، ث	g	Gain	غـ	ـغـ	ـغ ، غ
j	Jeem	جـ	ـجـ	ـج ، ج	f	Faa	فـ	ـفـ	ـف ، ف
H	Haa	حـ	ـحـ	ـح ، ح	q	Qaf	قـ	ـقـ	ـق، ق
x	Khaa	خـ	ـخـ	ـخ ، خ	k	Kaf	كـ	ـكـ	ـك ، ك
d	Dal	د	ـد	ـد ، د	l	Lam	لـ	ـلـ	ـل ، ل
*	Thal	ذ	ـذ	ـذ ، ذ	m	Meem	مـ	ـمـ	ـم ، م
r	Raa	ر	ـر	ـر ، ر	n	Noon	نـ	ـنـ	ـن ، ن
z	Zai	ز	ـز	ـز ، ز	h	Haa	هـ	ـهـ	ـه ، ه
s	Seen	سـ	ـسـ	ـس ، س	w	***Waw***	و	ـو	ـو ، و
$	Sheen	شـ	ـشـ	ـش ، ش	y	***Yaa***	يـ	ـيـ	ـي ، ي
S	Sad	صـ	ـصـ	ـص ، ص					

Figure 2-1: Arabic letters[2]. The bold letters are vowels, the underlined letters are not attached to succeeding letter in the same word.

2.3 Arabic Language Varieties

Arabic texts could be either vowelled, as the language of Qur'an or children's books; or unvowelled ones, used in newspapers, books, and media. Handling the unvowelled texts is confusing since an unvowelled word may have more than one

[2] We depend on Buckwalter xml transliteration in this figure and we use it in all transliterations in our dissertation.

meaning (Atwell et, al.) [15]. This classification is similar to classification of Arabic to Classical Arabic and Modern Standard Arabic. Arabic language varieties are shown in figure 2-4.

Diacritic and controls	◌َ	◌ُ	◌ِ	◌ْ	◌ً	◌ٌ	◌ٍ	◌ّ
name	Fateha	Damh	Kasra	Skon	Tanween	Tanween	Tanween	Shada
English sound	/a/	/u/	/i/	-	/an/	/un/	/in/	-
Example and spelling	كَ Ka	كُ Ku	كِ Ki	كْ K	كً Kan	كٌ Kun	كٍ Kin	كّ كْ+ك KK

Figure 2-2: Arabic diacritics and controls

Original (Arabic)	0	1	2	3	4	5	6	7	8	9
Original (Indo)	٠	١	٢	٣	٤	٥	٦	٧	٨	٩

Figure 2-3: Arabic numbers

Many linguists make a distinction between Classical Arabic (CA), the name of the literary language of the previous eras, and the modern form of literary Arabic, commonly known (in English) as Modern Standard Arabic (MSA). In term of linguistic structure, CA and MSA are largely but not completely similar (Ryding) [80].

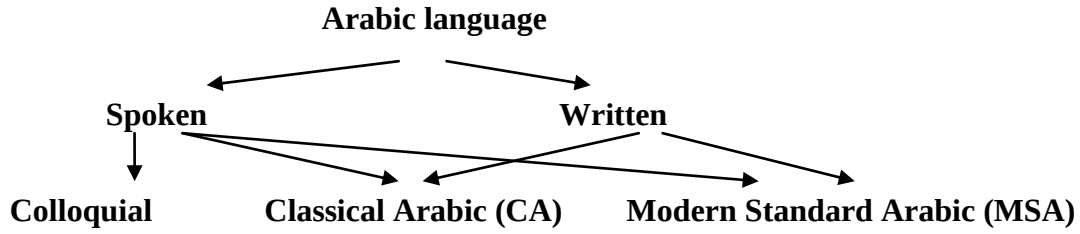


Figure 2-4: Arabic Language variations

In Classical Arabic words have diacritical marks which solve the ambiguity in the language. I.e., CA has less ambiguity than MSA. For example the word "kataba"-"كَتَبَ" (write (he)) has only one meaning "he writes". Removing

diacritics, in MSA, creates word-level ambiguity in segmentation process (Badr et, al.) [16].

MSA is the written language of contemporary literature, journalism, most of books etc. MSA is a descendant of CA and retains the basic syntactic, morphological, and phonological systems (Bin-Muqbil) [21]. MSA is highly ambiguous which results from removing diacritical marks from writing. For example the word "ktb"-"كتب" can be "kataba"-"كَتَبَ", "kutub"-"كُتُبْ", "kutiba"-"كُتِبَ" and "kat~aba"-"كَتَّبَ" which mean "he writes", " books", " was written" or "he caused to write", respectively.

2.4 Arabic Morphology

Morphologically, Arabic is a non-concatenative language. The basic problem with generating Arabic verbal morphology is the large number of variants that must be generated (Cavalli-Sforza et, al.) [27]. This problem is particularly difficult when a weak letter occurs in the word. Weak letters can be deleted or substituted by other letters because of Arabic linguistic theory (Shaalan) [85].

Affixing grammatical morphemes to the stem is a general property of most European languages, which have concatenative morphology where the word is prefix, stem[3] and suffixes. Although there are numerous exceptions, it enables us to analyze the structure of most words (Nugues) [74].

Concatenative morphology is not universal, however. The Semitic languages, like Arabic or Hebrew, for instance, have a templatic morphology that interweaves the grammatical morphemes to the stem (Nugues) [74].

We explain briefly how a word changes by adding clitics[4] and affixes to it. This subject is very rich and explaining all details is out of range of our dissertation; therefore we will explain the most important cases and leave the other to next chapters.

We have two opposite processes in any language, word generation (having the lemma/root and produce all possible words from it) and analyzing (having a word and extract the lemma/root with features from it). The first task is relatively easy

[3] Stem need not be an Arabic word.
[4] See chapter 4 for more details about clitics and affixes.

in Arabic language because of many unambiguous rules for this task. The second is very hard[5], especially if the lemma is the wanted base unit. For example if we have the lemma "Asrp"-"اسرة"(family) and the pronoun "hA"-"ها" (her) is attached to it, the result is "AsrthA"-"اسرتها" (her family) according to the rule *"if word ends by Taa marbuta and is attached to a pronoun then change this Taa marbuta to normal Taa"*. But if we have the word "AsrthA"-"اسرتها" and we want to get the lemma then we have many choices: "Asr~at"-"اسرَّت" "As~art"-"اسَّرت" "Asrp"-"اسرة", "Asrto"-"اسرتْ" and so on. This is a simple example but in most cases there are very hard cases to detect the lemma. The most famous case happens when one of the Arabic vowels exists in the root and one of the morphological rules is applied to it. In this case the analyzing is a very hard task. The important events in this case are deleting or changing the vowels as shown in figure 2-5.

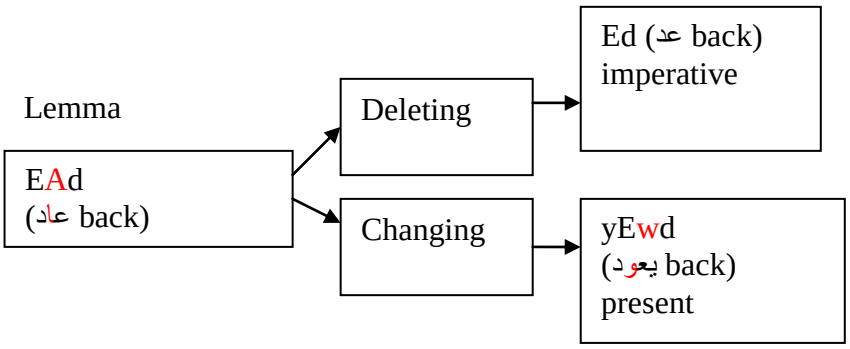


Figure 2-5: Inflection causes deleting and changing of a letter

Each Arabic word consists of **original** letters and possibly some **extra** letters. The original letters will not be deleted in any inflected form of that word, without morphological reasons. These original letters can be any letters of the alphabet except s, O, l, t, m, w, n, y, h and A. On the opposite side, the extra letters can be deleted in some inflections without any morphological reasons. The noun can consist of 3, 4 or 5 original letters. The verb can consist of 3 or 4 original letters.

2.5 Morphological rules

Morphology is the study of the structure and content of word forms. The rules of construction word forms are depending on the language under consideration.

[5] In case of Classical Arabic the ambiguity decreases which makes this task easier.

They are, in most cases, regular in Semitic languages like Arabic. Morphological rules can be either inflectional rules or word-formation rules.

2.5.1 Inflectional rules

Inflectional rules relate a lexeme to its forms (which uses kind of affix in order to form variants of the same word). Inflection is done by adding number, person, case, gender, tense mood … etc., to the word. Most of concatenative languages add affixes to the stem for this purpose. But the situation is different in Arabic language: letter deletion, insertion and replacing (especially with vowels) are used. The inflectional rules cover approximately almost all words, which means that Arabic inflection is regular. Examples of Arabic inflections are shown in figure 2-6.

Transliteration	**verb**	**meaning**	**Translit-eration**	**verb**	**meaning**
kataba	كَتَبَ	Wrote (he)	katabta	كَتَبتَ	Wrote (you-masc-sng)
yaktib	يَكتِب	Write (he)	taktub	تَكتُب	Write (you-masc-sng)
Iktub	إكتُب	Write (you)(imperative)	katabti	كَتَبتِ	Wrote (you-fem.-sng)
katabat	كَتَبَت	Wrote (she)	tkatubyn	تكتُبين	Write (you-fem.-sng)
taktub	تَكتُب	Write (you-masc. &she)	ktaabtmA	كتَبتما	Wrote (you-dual)
Iktuby	إكتُبي	Write (imperative)	taktubAn	تَكتُبان	Write (you-dual)
katabA	كَتَبا	Wrote (they-dual)	katabtuna	كَتَبتُنَ	Wrote (you-fem.-plural)
yaktubAn	يكتُبان	Write (they-dual)	taktubna	تَكتُبنَ	Write (you-fem.-plural)
IktubA	إكتُبا	Write (you-dual-imperative)	katabtum	كَتَبتُم	Wrote (you-masc.-plural)
katabna	كَتَبنَ	Wrote (they-fem)	taktabwn	تَكتبون	Write (you-masc.-plural)
yaktubna	يَكتُبنَ	Write (they-fem)	katabt	كَتَبت	wrote (I)
Iktubna	إكتُبنَ	Write (you-fem-imperative)	Oktub	أكتُب	write (I)
ktabwA	كتَبوا	Wrote (they-masc)	katbnA	كَتبنا	Wrote (we)
yktabwn	يكتبون	Write (they-masc)	nktbu	نكتبُ	Write (we)
IktabwA	إكتبوا	Write (you-masc-imperative)			

Figure 2-6: Inflection of merely verb “kataba”-“كَتَبَ” (write)
with gender, person and number.

2.5.2 Word formation

Word formation is the creation of new words. A number of languages have extensive non-concatenative morphology, in which morphemes are combined in complex ways (Jurafsky & Martin) [54]. A specific kind of non-concatenative morphology is called templatic morphology or root-and-pattern morphology. This is very common in Arabic, Hebrew, and other Semitic languages (Jurafsky & Martin) [54]. Word formation can be one of:

1. Derivational rules relate one lexeme to another lexeme (changes a word from one syntactic category into a word of another syntactic category or from one meaning to another). Some examples of Arabic derivation are shown in figures 2-7 and 2-8.

2. Compound (attaches two or more words together to make them one word). An example of an Arabic compound word is “HDrmwt”-“حضرموت” (Hadhramautt). It is compound from two words "HDr"-“حضر” which means (come) and "mwt"-“موت” which means (death), but the meaning of whole word is a name of a city in Yemen. There are many types of compound words in Arabic language; the previous example is the easiest one because there is no space between the compound words. Another example is “AslAm |bAd”-“اسلام آباد” (Islam Abad), i.e. two words separated by space, but the whole is a name of a city in Pakistan.

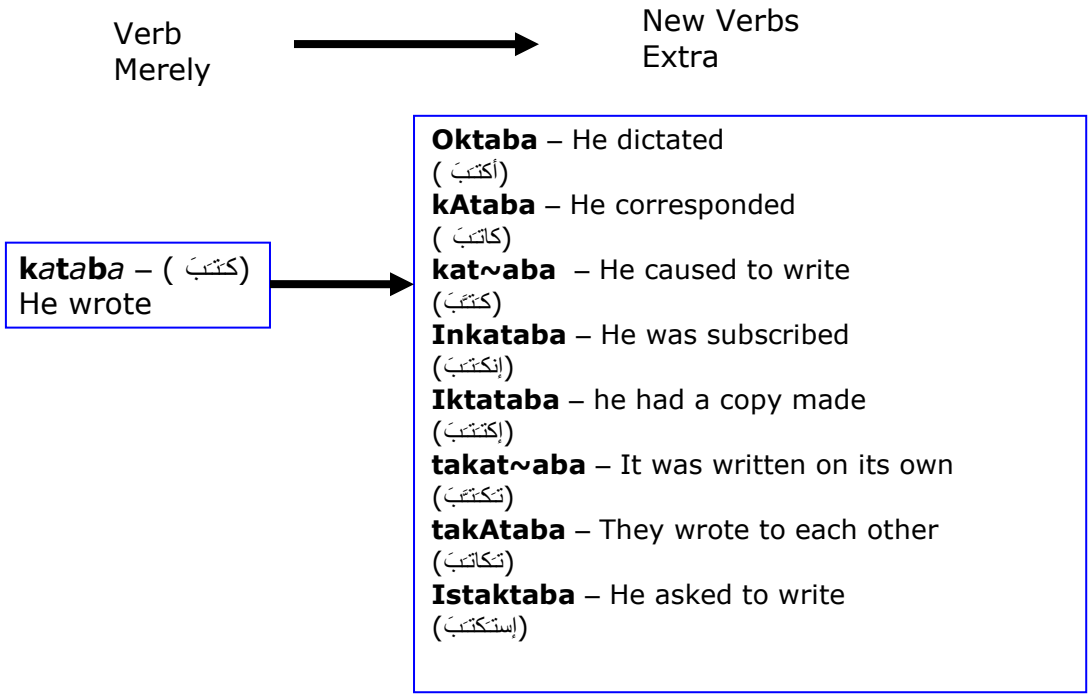


Figure 2-7: Deriving verbs from verb[6].

2.6 Arabic patterns (awzaan)

Because most of Arabic words are constructed in a regular way, the scientists describe them by morphological patterns (sometimes called balance). That pattern (wazen in Arabic) is composed of three origins (letters), which are denoted by f, E and l, where f corresponds to the first letter, E to the second letter and l to the third letter. The pattern describes the word construction (Al-Rajhi) [10] (Al-Hamlawy) [7]. By taking the root and applying the pattern to it, we will get another word construction. These rules are root–pattern morphology. Appendix A2 shows examples of using wazen (AL-Bidhani) [3] (Al-Galaiini) [6].

[6] Merely can be triple or quadruple. Extra can be made from triple or quadruple (by adding letters)

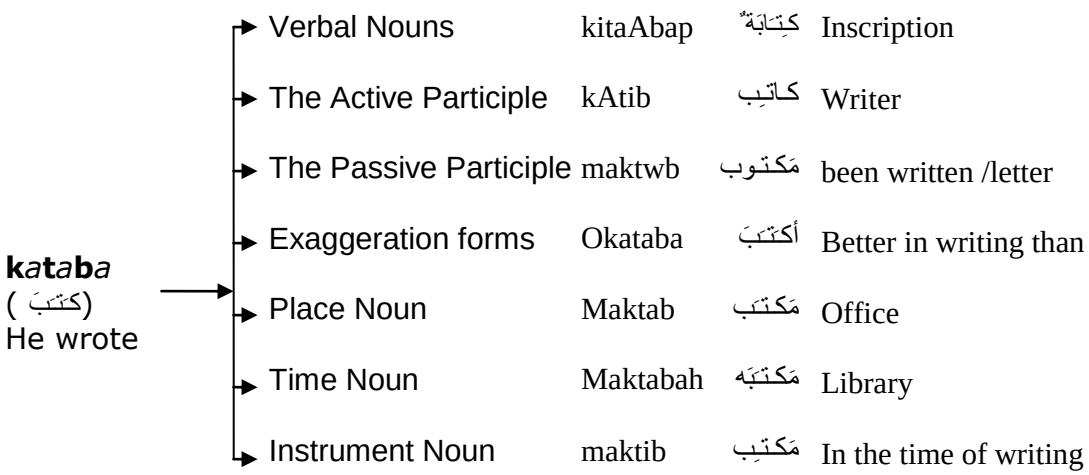


Figure 2-8: Deriving nouns from verb.

2.7 Words in the sentences

As we know Arabic is written from right to left where the letters are attached together to form the words. In most cases, the particles and pronouns are attached to the word, i.e., the word can be composed of more than one part of speech. It adds another problem to Arabic language, which must be solved by tagger. For example a complete sentence can be compressed in to one word:

wsyktbhA (وسيكتبها and he will write it)

When we talk about sentences, syntax comes into play. As we know, there are two distinct fields in languages which are morphology and syntax. Morphology describes the structure of words internally, syntax describes how words are composed to yield phrases and sentences (Habash) [45].

Arabic sentences can be divided into two types of sentences: verbal sentences and nominal sentences. Nominal sentences are also called copular/equational sentences (Habash) [45].

Each word inside a sentence can be **affective** (that affects what follows), **affected** (affected by what is before it) or **neither affective nor affected** as in the case of spatial words. The **effect** is the change of the form of the affected word enforced by the affective word (Al-Galaiini) [6]. Examples of effect are changes

the case to nominative, accusative … etc. The third category (**neither affective nor affected**) is special and very limited (Al-Galaiini) [6].

For example a preposition before a noun causes reduction of that noun. The reduction is, in this example, the effect (where the noun (affected) followed preposition (affective) will be in genitive case).

Arabic can be seen as a language with a network of dependency relations in every phrase or clause, which are key components of the grammatical structure of the language (Ryding) [80].

Chapter 3

Comparing Arabic tagsets and Designing a New One

3.1 Introduction

The first step for the annotation of corpora is the compilation of a tagset that can accurately describe and cover the whole information about the language (Khoja) [57]. A ***tagse****t* is *a set of tags (symbols) representing information about parts of speech and about values of grammatical categories (case, gender, etc.) of word forms.* Tagset is the basis of almost all NLP fields. A good tagset is very important in the fields of NLP and is the foundation stone in these fields.

We believe that before dealing with the Arabic language, we need an Arabic tagset which contains all or at least the most *important* Arabic language features.

In this chapter, 10 Arabic tagsets are compared and their limitations indicated. We present a new Arabic tagset avoiding these limits. The design is intended for Arabic language only and is not based on tagsets for other languages. It is a multilevel tagset compatible with CA and MSA. The noun classes have three levels (fixed POS types, grammatical feature and changed POS types), verbs have two levels (POS types and grammatical features) and particles have two levels (working and meaning). We also introduce the notion of tagset interleaving.

The third level (designed for noun only) is not yet implemented and is not mentioned in the remaining chapters of this disertation. Summary and comparison of Arabic tagsets

Most of the papers are interested in constructing a tagger and introduce its tagset as a by-product. In this chapter we consider the following tagsets for Arabic: Khoja tagset [57][59]; Al-Qarany tagset [9], Majdi Sawalha tagset [82], Yahya Alhadj tagset [38], Buckwalter tagset, Reduced Buckwalter tagset (and its variants: Bies tagset, Kulick tagset [65] and Extended Reduced tagset) (Hbash) [45], KATIB POS tagset [47][49] and PADT tagset (Habash) [45].

Almost all of these taggers either use tagsets derived from English (which is not appropriate for Arabic) or use summary of all Arabic features (which is more theoretical than practical).

We summarize the above mentioned Arabic language tagsets with their limits and specifications.

3.1.1 Khoja tagset

The Khoja tagset, developed by Shereen Khoja, is one of the earliest tagsets for Arabic (Khoja) [57][59]. Figure 3-1 shows Khoja POS.

The linguistic attributes of nouns and verbal attributes that have been used in this tagset are shown in figure 3-2. We have a few remarks on this tagset:

1. The attribute “person” in noun class is a mistake here because the word “كتاب” **book** has no person. In this way all researchers apply the person feature to the noun, but the noun is different from verb. The inflections of the verb always contain the pronoun, but there are inflections of a noun without any pronoun. So a noun cannot be treated in the same way as the verb.
2. Particles have no attributes. The classifications of particles are interleaved among their operation and meaning.
3. It is a very simple tagset, i.e., many of Arabic classes are not taken into account.

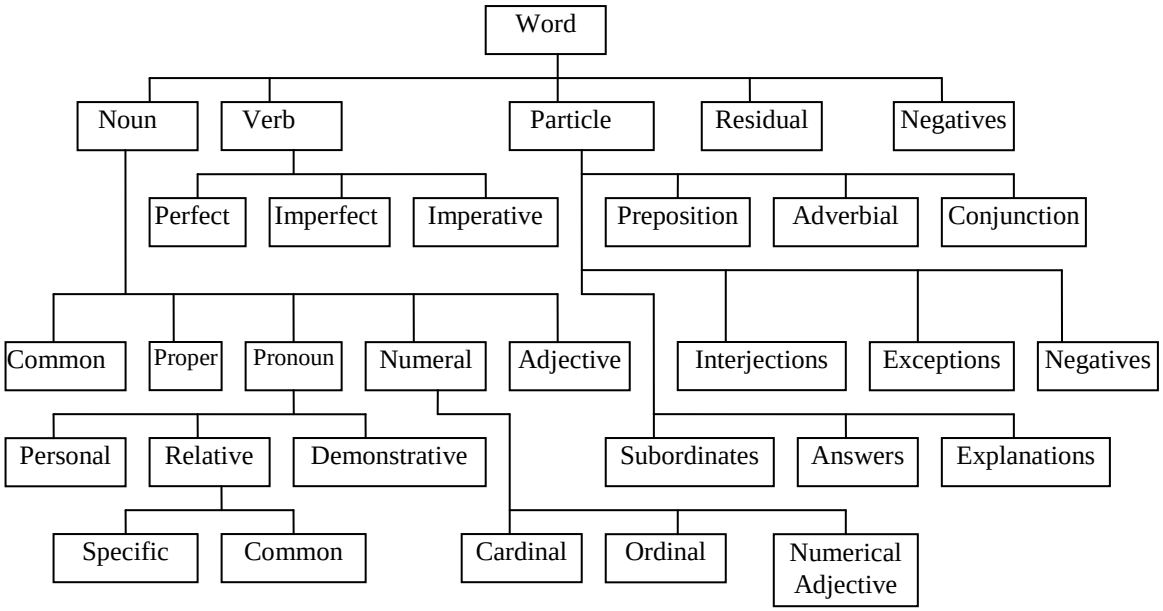


Figure 3-1: Khoja tagset

Noun attributes			
Gender	Masculine	Feminine	Neuter
Number	Singular	Dual	Plural
Person	First	Second	Third
Case:	Nominative	Accusative	Genitive
Definiteness	Definiteness	indefiniteness	

Verb attributes			
Gender:	M Masculine	F Feminine	N neuter
Number	S Singular	Du Dual	Pl Plural
Person:	1 First	2 Second	3 Third
Mood	I Indicative	S Subjunctive	J Jussive

Figure 3-2: The Noun and Verbal attributes of Khoja Tgaset

3.1.2 Al Qrainy tagset:

It was written by (AlQrainy & Ayesh) [9] for Automated POS tagging in Arabic. They take the classical classification of Arabic words into noun, verb and particle. Figure 3-3 shows the main classification of this tagset. The linguistic attributes of nouns and verbal attributes that have been used in this tagset are the same as in Khoja (Figure 3-2), but the neuter feature for the verb attribute does not exist.

The same remarks we have made about Khoja tagset apply here, and additionally punctuations and foreign words are not covered by the Al-Qrainy tagset. There is a technical error in the figure 3-3, which we took from (AlQrainy & Ayesh) [9]. If we look at the figure, we understand that the "common" is a part of "demonstrative", while indeed they should both be parts of "Noun".

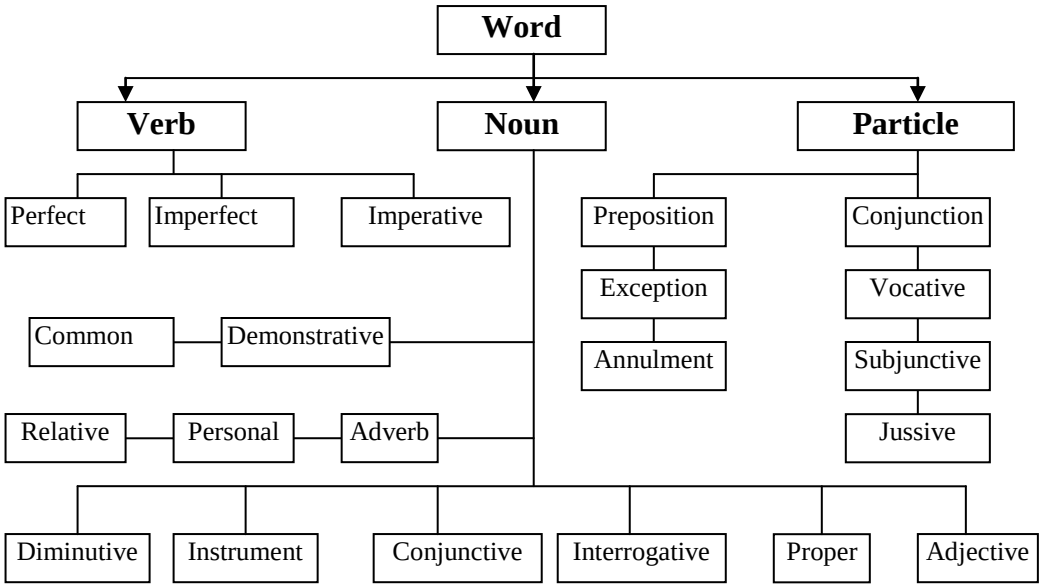


Figure 3-3: Al Qrainy tagset Hierarchy

3.1.3 Sawalha tagset

In the Sawalha tagset (Sawalha) [82], a tag consists of 22 characters; each position represents a feature and the letter at that location represents a value or attribute of the morphological feature; the dash "-" represents a feature not applicable to a given word. The first character shows the main Parts of Speech: noun, verb, particle, punctuation, and residual. The 2nd, 3rd and 4th characters are used to represent subcategories; traditional Arabic grammar recognizes 34 subclasses of noun (letter 2), 3 subclasses of verb (letter 3), 21 subclasses of particle (letter 4). Residuals and punctuations are represented in letters 5 and 6 respectively. The next letters represent traditional morphological features:

gender (7), number (8), person (9), morphology (10) case & mood (11), case & mood markers (12), definiteness (13), voice (14), emphasize (15), transitivity (16), humanness (17), variability and conjugation (18). Finally there are four characters representing morphological information which is useful in Arabic text analysis, although not all linguists would count these as traditional features: augmented and unaugmented (19), number of root letters (20), verb internal structure (21), noun finals (22).

The **Majdi Sawalha** tagset is not tied to a specific tagging algorithm or theory, and other tagsets could be mapped onto this standard, to simplify and promote

comparisons between, and reuse of Arabic taggers and tagged corpora. Figure 3-4 shows **Majdi Sawalha** main POS classification.

We have a few notes on this tagset. In spite of taking most of noun and verb classification, it neglects the variation of particles classification. Similarly as Khoja, this tagset does not distinguish between working and meaning of particles. For example "fklA Ax*nA b*nbh"-"فَكُلًّا أَخَذْنا بِذَنبِهِ" (We took each one **by/because** his sin) the particle b is for caution and preposition at the same time (it is preposition used for caution). It means that it should have two tags simultaneously[7]. There are many interleavings between types in this tagset.

Sawalha tagset summarizes almost all the Arabic classifications, especially for verbs and nouns. However, some of the classifications (attributes) are useless (redundant) tags, for tagging system. For instance, the value at position 20 "number of root letters", position 21 "verb root attribute" can be known if the root is known. The same case with position 13 "Definiteness" it is a feature for closed classes of noun categories. It seems that this tagset is more theoretical than practical.

3.1.4 Yahya Elhadj

(Elhadj) [38] presented the development of an Arabic part-of-speech tagger that can be used for analyzing and annotating traditional Arabic texts, especially the Qura'n text. The developed tagger employed an approach that combined morphological analysis with Hidden Markov Models (HMMs) based-on the Arabic sentence structure. For this purpose, Elhadj created his own tagset (2009). See figures (3-5, 3-6 & 3-7). Figure 3-5 represents the tagset as a DAG (directed acyclic graph), which is the choice of the author.

This tagset has the following limitations: particles have no attributes. It is particularly simple with respect to verb and noun classifications. The case of noun was excluded which is very important in syntax analyses. It does not show any features for verbs and this is not a good choice, because Arabic verbs often have implicit pronouns and so on.

[7] See section 3.4.2 for more details.

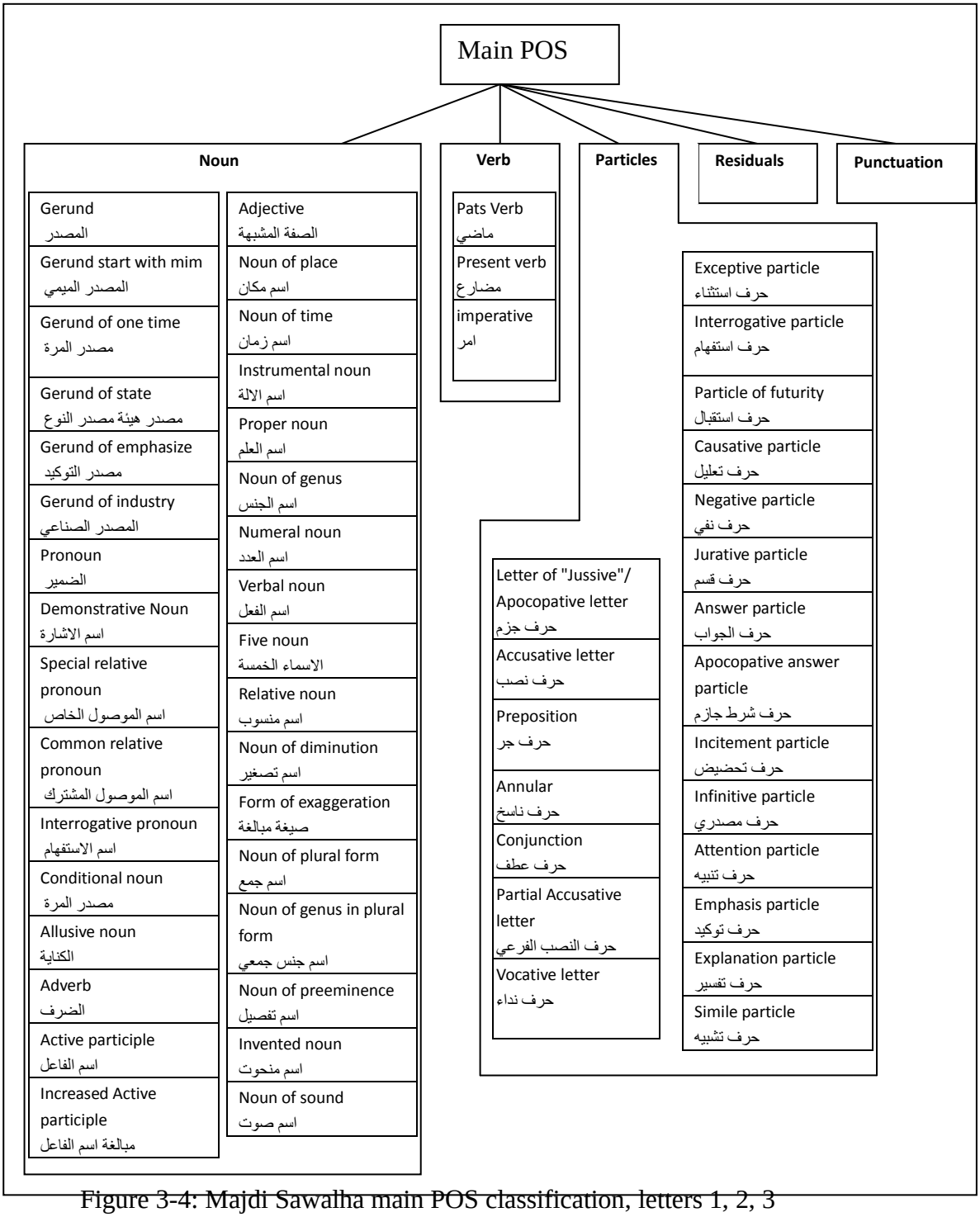


Figure 3-4: Majdi Sawalha main POS classification, letters 1, 2, 3 and 4 only.

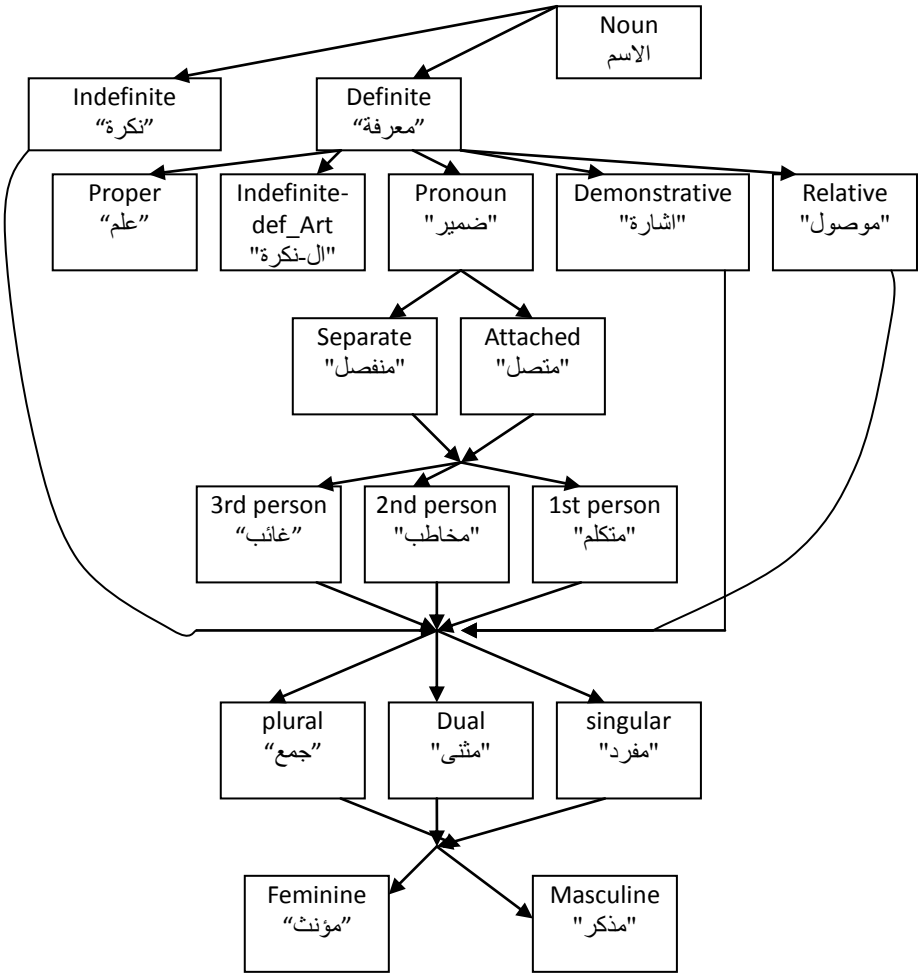


Figure 3-5: Noun and its sub-categories in Elhadj tagset.

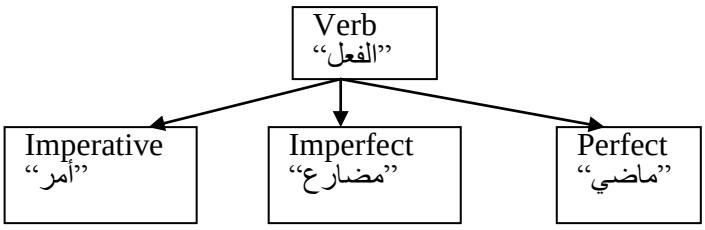


Figure 3-6: Verb and its temporal-forms in Elhadj tagset.

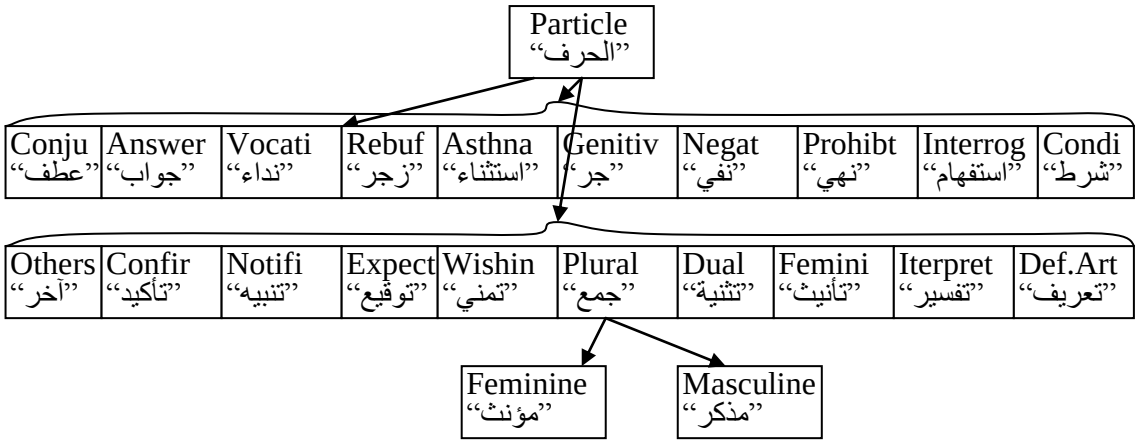


Figure 3-7: Main groups of particles in Elhadj tagset.

3.1.5 Buckwalter tagset

The Buckwalter tagset (figure 3-8), developed by Tim Buckwalter, is a form-based tagset. The Buckwalter tagset is considered very rich for many computational problems and approaches. Several tagsets have been developed that reduce it to a "manageable" size (Habash) [45].

In this tagset there is no distinction between categories and features for POS. The particle classification has no attributes. He does not distinguish between attached pronouns or other **clitics** and inflection of the word (suffixes). The Yaa Alnasabi is omitted, and treated as an attached pronoun.

VERB	
VERB	verb
PSEUDO_VERB	pseudo-verb
PV	perfective verb
PV_PASS	perfective passive verb
PVSUFF_DO:<PGN>	direct object of perfective
PVSUFF_SUBJ:<PGN>	verb
	subject of perfective verb
IV	imperfective verb
IV_PASS	imperfective passive verb
IVSUFF_DO:<PGN>	imperfective verb direct
IV<PGN>	object
IVSUFF_SUBJ:<PGN>	imperfective verb prefix
_MOOD: <Mood>	imperfective verb subject
	and mood suffix
CV	imperative (command) verb
CVSUFF_DO:<PGN>	imperative verb object
CVSUFF_SUBJ:<PGN>	imperative verb subject
Particles	
PREP	preposition
CONJ	conjunction
SUB_CONJ	subordinating conjunction
PART	particle
CONNEC_PART	connective particle
EMPHATIC_PART	emphatic particle
FOCUS_PART	focus particle
FUT_PART	future particle
INTERROG_PART	interrogative particle
JUS_PART	jussive particle
NEG_PART	negative particle
RC_PART	response conditional particle
RESTRIC_PART	restrictive particle
VERB_PART	verb particle
VOC_PART	Vocative Particle

Nominal	
NOUN	noun
NOUN_NUM	nominal/cardinal number
NOUN_QUANT	quantifier noun
NOUN.VN	deverbal noun
NOUN_PROP	proper noun
ADJ	adjective
ADJ_COMP	comparative adjective
ADJ_NUM	adjectival/ordinal number
ADJ.VN	deverbal adjective
ADJ_PROP	proper adjective
ADV	adverb
REL_ADV	relative adverb
INTERROG_ADV	interrogative adverb
PRON	pronoun
PRON_<PGN>	personal pronoun
POSS_PRON_<PGN>	Possessive personal pronoun
DEM_PRON_<GN>	demonstrative pronoun
REL_PRON	relative pronoun
INTERROG_PRON	interrogative pronoun
NSUFF<Gen><Num><Cas><Stt>	nominal suffix
CASE<Def><Cas>	nominal suffix
DET	determiner
Other	
PUNC	punctuation
ABBREV	abbreviation
INTERJ	interjection
LATIN	latin script
FOREIGN	foreign word
TYPO	typographical error
PARTIAL	partial word
DIALECT	dialect word

Figure 3-8: Buckwalter tagset components (the source is (Habash) [45]).

3.1.6 Reduced Buckwalter tagsets: BIES, KULICK and ERTS

3.1.6.1 BIES tagset

The Bies tagset (Figure 3-9) was developed by Ann Bies and Dan Bikel as a subset of Buckwalter tagset with around 24 tags variants. It was inspired by the Penn English Treebank POS tagset (Habash) [45].

It is a very simple set which misses many useful features, in particular many classes of nouns, verbs and particles. The nouns, verbs and particles have no attributes.

Nominals		DT	determiner / demonstrative pronoun,
NN	RP	RP	Particle
NNS	IN	IN	preposition or subordinating conjunction
NNP	singular proper noun	**Verbs**	
NNPS	plural/dual proper noun	VBP	active imperfect verb,
PRP	personal pronoun,	VBN	passive imperfect/perfect verb,
PRP$	possessive personal pronoun,	VBD	active perfect verb,
WP	relative pronoun	VB	imperative verb
JJ	adjective,	Others	
RB	adverb,	UH	interjection,
WRB	relative adverb,	PUNC	punctuation,
CD	cardinal number,	NUMERIC_COMMA	The letter r used as a comma,
FW	Foreign word	NO_FUNC	unanalyzed word
Particles			
CC	coordinating conjunction,		

Figure 3-9 : The Bies tagset

3.1.6.2 The Kulick tagset

The Kulick tagset [65] was developed by Seth Kulick and shown to be beneficial for Arabic parsing (Habash) [45]. The Kulick tagset contains 43 tags that extend the Bies tagset. It is a very simple set which misses many useful features and classes.

3.1.6.3 The Extended Reduced TagSet (ERTS)

ERTS is the base tagset used in the Amira system. ERTS has 72 tags. It is a subset of the full Buckwalter morphological set defined over tokenized text. ERTS is a superset of the Bies/RTS tagset. In addition to the information contained in the Bies tags, ERTS encodes additional morphological features such as number, gender, and definiteness on nominals only (Habash) [45]. Again, it is a very simple set. It misses many classes of particles. The particles have no attributes.

3.1.7 The CATIB POS tagset

The CATiB tagset (figure 3-10) was developed for the Columbia Arabic Treebank project (CATiB) (Habash)[47][49]. There are only six POS tags in CATiB. The

simplicity of the POS tagset is intended to speed up human annotation and yet maintain the most important distinctions. It is the simplest tagset, where many classes and features are missed.

Tag	**Remark**	**Tag**	**Remark**
VRB	All verb Types	PROP	proper nouns
VRB-PASS	passive-voice verbs	PRT	Particle
NOM	Nominal	PNX	punctuation marks

Figure 3-10: the CATIB POS tagset

3.1.8 The PADT tagset

The PADT tagset (see figure 3-11 & 3-12), used in the ElixirFM analyzer, was developed for use in the Prague Arabic Dependency Treebank (Habash) [45]. The PADT tagset is defined for ATB tokenized Arabic. Each tag consists of two parts: POS and Features. It misses many classes and features. Particles have no attributes.

Tag	Remark	Tag	Remark	Tag	Remark
VI	imperfect verb	Y	Abbreviation	C	Conjunction
VP	perfect verb	S	Pronoun	P	Preposition
VC	imperative verb	SD	demonstrative pronoun	I	Interjection
N	Noun	F	particle	G	Graphical symbol
A	Adjective	FI	interrogative particle	Q	Number
D	Adverb	FN	negative particle	--	Isolated definite article
Z	Proper noun				

Figure 3-11: POS for The PADT tagset

Mood	Indicative	Subjunctive	Jussive	D (ambiguous)
Voice	Active	Passive		
Person	1 speaker	2 addressee	3 others	
Gender	Masculine	Masculine		
Number	Singular	Dual	Plural	

Figure 3-12: the PADT features

3.2 Traditional Arabic POS

POS is the most studied field in the Arabic language. The distinctions between parts of speech were investigated and specified. We will show, in this section, the classical classifications. The detailed explanation of these classes is far too complicated to be presented in this dissertation, therefore we will describe only the most important classes and features. In this section we will show the main classification for Arabic word and the subclasses of these main POS.

3.2.1 Main Arabic POS

The first classification of a word in traditional and modern Arabic is noun, verb and particle (Al-Rajhi) [10] (Al-Galaiini) [6] (Al-Dahdah 1989) [4].

3.2.2 Arabic Noun Classes

There are many types of noun. A noun can be a described by more than one type or status. The summaries of noun classes according to their classification are in figure 3-13 & 3-14 (Al-Dahdah) [4]:

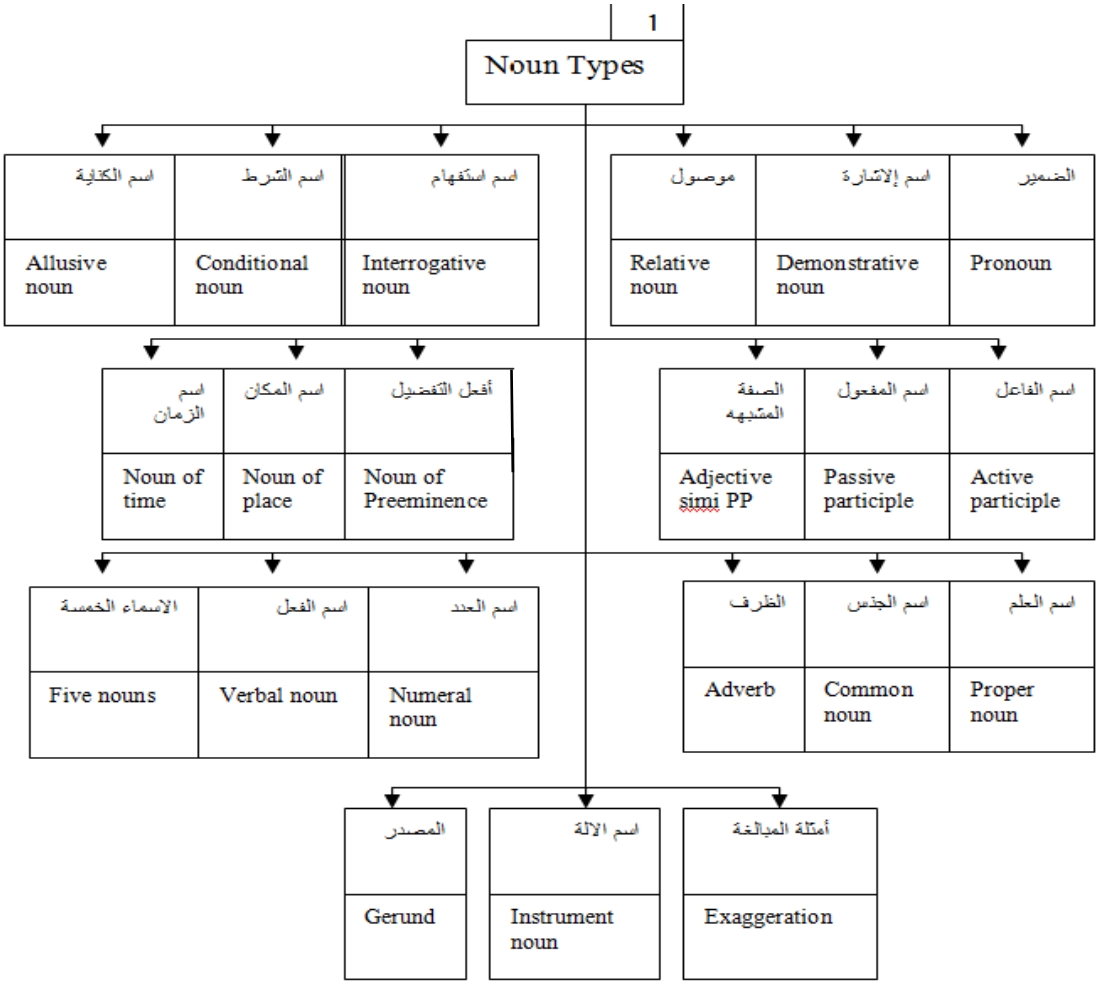


Figure 3-13 : Noun classification according to its types (the source is (Al-Dahdah) [4])

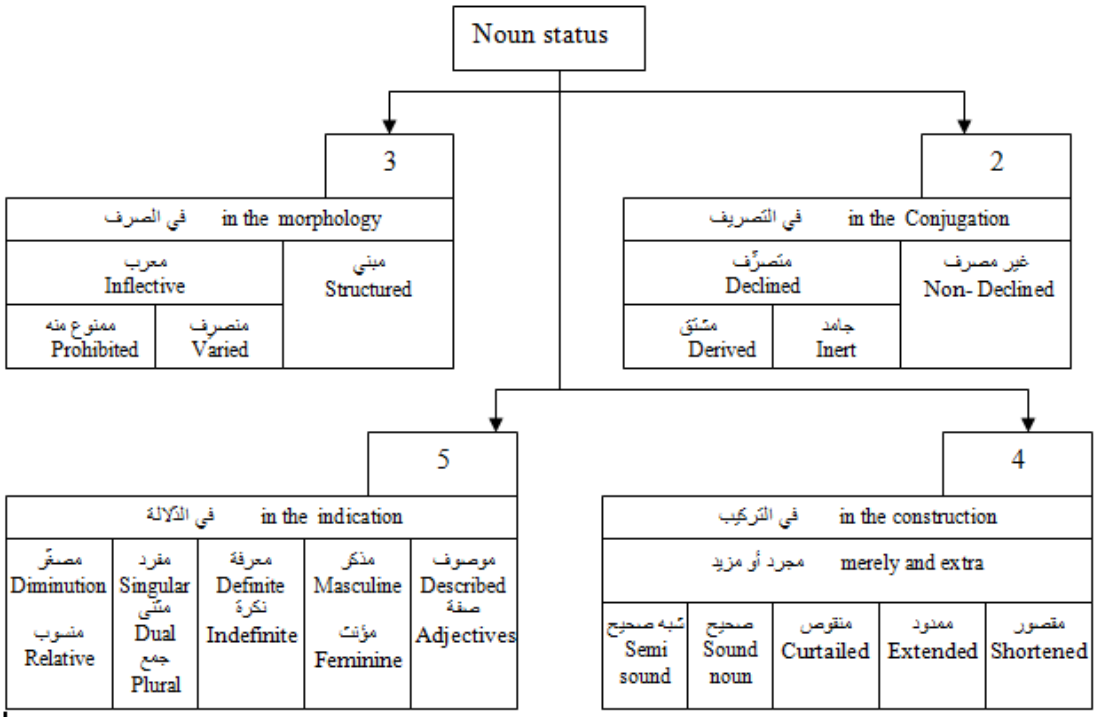


Figure 3-14: Noun classification according to its status (the source is (Al-Dahdah) [4])

3.2.3 Arabic Verbs

The verb can be classified according to:

1. If it has vowels or not: it has approximately 30 subtypes (see (Sawalha) [82]).
2. If it is complete or incomplete
3. Voice (passive or active).
4. If it is merely or has extra letter and the number of letters.
5. If it has certainty or not
6. Tense.
7. Transitivity.
8. If it has negation or not.
9. If verbs have special case (interjection verb form صيغة التعجب) or not.
10. Variability & Conjugation

Any verb has features [Gender + Number + Person + Mood]. We can see that there are interleavings among all these classifications.

3.2.4 Arabic Particles

There are two classifications for the particles according to.

1. Their working in the sentence.
2. Their meaning.

The first classification is done according to the effect of the particle on the following word (see Section 2.7). The classes are defined according to the effect: nominative, accusative, genitive, jussive… etc. There are also particles which have no effect and they are classified as "not working particles".

The second classification has many classes, in (Al-Galaiin) [6] there are 31 interleaved types: negative particle (حرف نفي), answer particle (حرف جواب), explanation particle(تفسير حرف), conditional particle (حرف شرط), exhortation particle (حرف تحضيض), offering particles (احرف العرض), warning particles (احرف التنبيه), subordinating conjunction (حرف مصدري), future particle (حرف استقبال), emphatic particle (حرف التوكيد), interrogative particle (حرف استفهام), wishing particles (حرف التمني), pleasing particles (حرف الترجي), simile particle (حرف التشبيه), relation particles (حرف الصلة), purpose particle (حرف التعليل), aversion particle (حرف ردع), l_letter meaning (لامات), feminine Taa (تاء التأنيث), stopping Haa (هاء السكت), request particles (حرف طلب), nunation particles (حرف تنوين), vocative particle (حرف نداء), coordinating conjunction (حرف عطف), accusative particle (حرف نصب), imperative particle (الامر), jussive particles (حرف جزم), prohibition particle (حرف نهي), preposition (حرف جر), particles similar to verbs (احرف مشبهه بالفعل), particles similar to Laisa (verb) (احرف مشبهه بليس).

In (Al-Dahdah) [4] there are 40 interleaved types (some types from (Al-Galaiini 1990) do not exist in (Al-Dahdah 1989)) which add the following particles: swearing (قسم), strike (إضراب), starting (استفتاح), palinode (استدراك), exceptive (استثناء) , beginning (ابتداء), surprise (مفاجأة), details (تفصيل), definition (ندبة) ,(تحقيق), (تخيير), (تصديق), (تعريف), intention (غاية), adverbial (ضرفية), superfluity (زيادة) , increasing (تكثير), decreasing (تقليل).

(Al-Moradi) [8] The grammarian limited the particle to approximately 50 types (in meaning).

3.3 Designing an Arabic Tagset

There are many reasons for designing a new Arabic tagset. We wanted to construct an Arabic tagset compatible with CA and MSA. Also, this tagset should not have the limits of other tagsets. We construct this tagset according to Arabic specification. The last reason is very practical – we plan to annotate a large Arabic corpus with this tagset. The annotators will be students of the departments of Arabic language in the University of Mustansiriyah (Baghdad). This idea has already got acceptance from the head of that department. Within a few years, we believe that we will have a huge annotated corpus, because all the students of this department will work on it. Therefore we needed a tagset familiar to them and easy to master in, and rich in information.

3.3.1 Designing criteria

(Elworthy) [40] The design of an appropriate tagset is subject to both external and internal criteria:

1. The external criterion is that the tagset must be capable of making the linguistic (for example, syntactic or morphological) distinctions required in the output corpora.
2. The internal criterion is that of making the tagging as effective as possible.

The first and second criteria must be balanced. As a part of point 2, we should note that very fine-grained distinctions may cause problems for automatic tagging if some words can change grammatical tag depending on function and context (Atwell) [14].

The problem of tagset design becomes particularly important for highly inflected languages. If all of the syntactic variations which are realized in the inflectional system were represented in the tagset, there would be a huge number of tags, and it would be practically impossible to implement or train a simple tagger. (Elworthy) [40] has suggested that what is important is to choose the tagset appropriate for the application, rather than to optimize it for the tagger.

(Feldman) [41] did test on several languages with tagsets of various sizes and found out, that there is no clear relationship between tagset size and tagging accuracy. However, generally smaller tagsets peform better on unknown words.

In this chapter we will design an Arabic tagset. The construction is based on the deficiencies of the other tagsets. It has two fields for each POS, one for classification or working and the second for feature or meaning. We will differentiate between classes of POS and grammatical features or between particles working and meaning. For example the plural noun is a noun with plural feature.

Another important factor for adding a tag of a given type is the analysis: is the tag useful in translation, semantics, and speech recognition, and so on, or not? From this point of view, we can select a tagset. All these criteria were taken into account when building the new Arabic POS tagset.

3.3.2 Tagset Interference or interleaving

We introduce another design decision to consider when designing a tagset: interference or interleaving. This question emerges when we use many syntactical classes and unifying many classifications into one. The tagset has interleaving if one word has more than one class (POS) at the same time and all these classes are true. It is often due to an error in the design of the tagset. According to our analysis of Arabic tagsets, the increase of POS numbers in a tagset, without augmentation, increases the possibility of interleaving. Most of the simple and small tagsets (such as CATIB and PADIT…) don't have interleaving. Let us take a practical example of a large tagset: Sawalha tagset (Sawalha) [82]. According to this tagset, for the following example "فَكُلًّاأَخَذْنا بِذَنبِهِ" "fklA Ax*nA b*nbh" "We took each one by/because his sin" the b (Baa) is for caution and preposition at the same time (it is preposition used for caution) and they are both true. It means that there are two tags (true) simultaneously. So this tagset has interleaving. This has happened because it is a large morphosyntactic tagset. We must see that interleaving is different than word sense where the word has different meanings or tagging where the word has many tags (non-interleaved tags).

When a word has more than one POS this does not mean there is interleaving but it depends on these classes. Let us consider another example for showing interleaving. Let the tagset consist of three tags only: noun, verb and particle. This tagset, for sure, does not have interleaving. Now, we want to extend this tagset and, mistakenly, we add subject as a new tag. Now, this tagset has interleaving because all subjects are nouns. If we have a word X, we cannot say it is subject or noun (if it is a subject) alone, but we say that it is subject and noun.

For this and similar cases, we have two solutions simultaneously. The first solution is that we add some of the interleaved classes as classes and the other ones as attributes. In the previous example, Noun class is a class and Subject class becomes an attribute. The second solution is that we divide the tagset into levels. In the previous example, we add a level for morphological classes and a level for syntactic classes. Any word will have more than one level.

In the proposed tagset we collect these two solutions according to the requirements as the reader can see in the next sections.

3.4 A New Arabic Tagset

3.4.1 Main POS

The first classification of a word is noun, verb and particle (Al-Rajhi) [10] (Al-Galaiini) [6] (Al-Dahdah) [4]. But there are symbols used in the written text as punctuations, foreign words, numbers, and so on. (Khoja) [57][59] used two other categories which are residuals and punctuation. This is true for normal Arabic text, but in Qur'an there are other symbols that do not exist in any other text which are stopping symbols. These symbols in some cases are taken as sentence ending (by force or optional). They can be made a part of the punctuation category or a new category (special) can be created for them. Figure 3-15 shows main POS for Arabic, the same as in most of other tagsets.

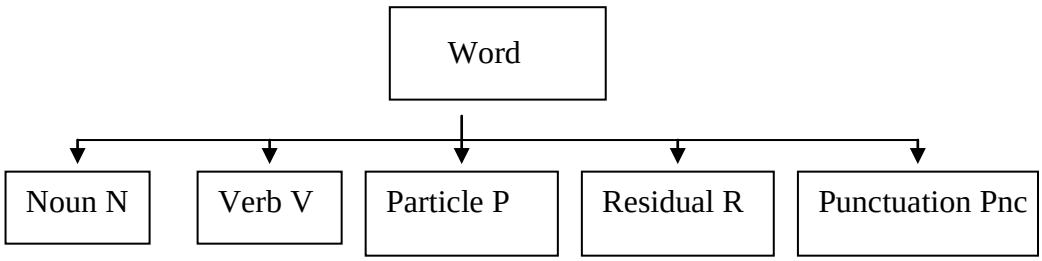


Figure 3-15: Main POS.

3.4.2 Arabic noun class in the proposed tagset

If we go back to figures 3-13 & 3-14, we cannot take all these classifications because they will cause highly ambiguous results due to the interleaving of these classes. According to the two levels idea of our tagset, the nouns classes can have class & features only. The final noun classes and subclasses in the proposed tagset are

shown in figure 3-16. The features of the noun in the proposed tagset are shown in Figure 3-17. One can observe the following:

1. Person attribute for nouns was not used here because of the example "ktAb"-"كتاب" (book). It is not a person. Therefore "ktAbhA"-"كتابها" (her book) has two POS.
2. The derived nouns are not taken into account because they are interleaved with other types as adjectives.
3. The constant adverb class was added, only, to this level.
4. The definedness feature was not taken because we deal with the definite particle as independent particle and the classes which have definiteness feature are constant: pronouns, demonstrative, proper nouns etc.

The tags of nouns start with letter N followed by Nouns POS followed by Features (Number+ Gender + Case + Structured) respectively. For example the tag NDem_SMAY is a Demonstrative Noun Singular Masculine Accusative structured.

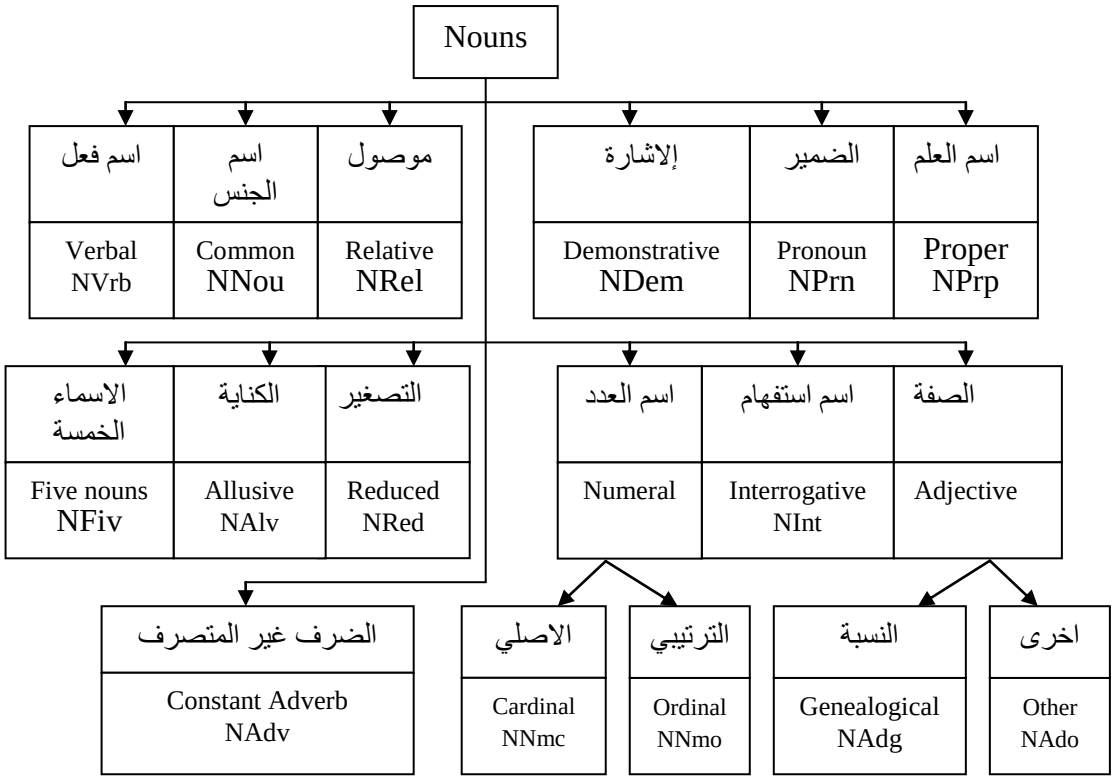


Figure 3-16: Arabic Noun Classes in the proposed tagset.

Gender:	Masculine	Feminine	Common
Number:	Singular	Plural	Dual
Case:	Nominative	Accusative	Genitive
Structured	Yes	No	

Figure 3-17: Noun features in the proposed tagset

3.4.3 Arabic Verb Classes and Attribuits in our Tagset

For the previous classification, we can take the verb classes and verbal attributes (features) as in figure 3-18 & 3-19 respectively. This classification will remove the interleaving which happened by variation of classification.

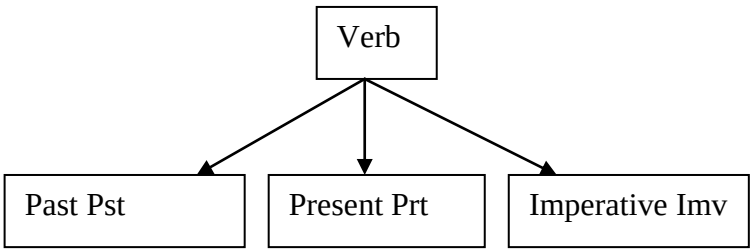


Figure 3-18: Verb classes in the proposed tagset

Gender:	Masculine	Feminine	Common (مشترك)	
Number:	Singular	Plural	Dual	
Person:	First	Second	Third	
Mood:	Nominative	Accusative	Jussive	Non
Certainty	Yes	No		
Structured	Yes	No		
Voice	Passive	Active		

Figure 3-19: Verbal attributes in the proposed tagset

3.4.4 Particles Classification in the proposed tagset

The classes of the particle, in our tagset, are defined according to the particle working. We summarized all of them in Figure 3-20.

The particle meaning is an attribute, in our tagset, of particles. As we can see the particles have 50 types (in meaning). Some of these classes can be combined into one class according to similarity of their meanings, therefore we can reduce the number

[8] The word ending will be changed (letter or diacritics) according to the case of the word(nominative accusative …). In the case of structured word, the word ending will be constatnt at all word cases (nominative, accusative …)

from 50 to 21 as in Figure 3-21. For example the classes: imperative, exhortation, pleasing, wishing, offering are unified to request class and so on.

Prepositions are a group containing almost all of the previous classes. Each preposition has multiple meanings which is a subset of the previous classes. For example the preposition "Baa" has 13 different meanings (Al-Galaiini) [6]. The interesting thing in preposition is that it has the same working in the sentence which is the reduction. Particles' working can be: for-jussive particles, for-reduction (for-genitive) particles (preposition), for nominative particles, for-accusative particles, for conjunction, not-working particles, Prevented. We want to show the difference between "for conjunction" and "not working" particles. The first particles translate the case of the word before it to the word following it. The second kind of particles does not do anything.

Finally, we will use the following important particle classes as in Figure 3-22 and the meaning of particles is shown in Figure 3-23:

للجزم	للجر	للعطف	للنصب	غير عاملة او مكفوفة عن العمل	النسخ (نصب ورفع)	كافة عن العمل
for Jussive[9] Jus	For Reduction[10] (preposition) Red	For Conjunction[11] Cnj	for Accusative[12] Acu	Not working[13] Or Preventive Non	Copier[14] Cop	Prevent[15] Prv

Figure 3-20: The classes of particles (working) in the proposed tagset.

[9] The present tense verb after these particles is in jussive mood.
[10] The noun after these particles is in genetive case.
[11] The nouns or verbs conjected by these particles must have the same case.
[12] The nouns or verbs after these particles are in accusative case.
[13] They do not have any effect on the following word.
[14] They have dual effect on the following words. One of the following words is in nominative and the other one in accusative case
[15] Any particle after this particle will be "not working" (i.e., prevented from working).

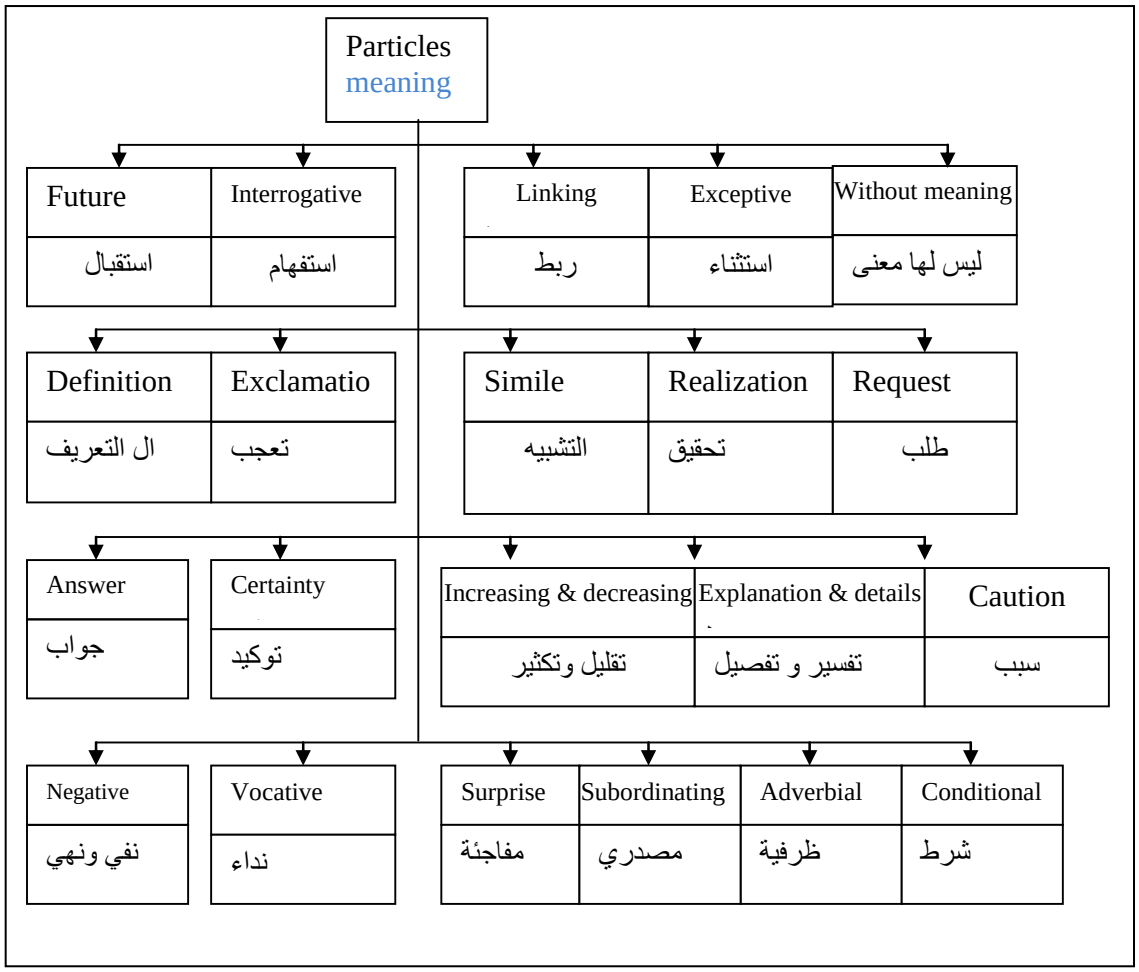


Figure 3-21: Particles meaning in the proposed tagset (features).

3.4.5 Residuals and punctuation

Residuals can be symbols of numbers, mathematical formulas, abbreviations, acronyms and so on. We must distinguish between the symbol of numbers (1, 2…) and nouns of number (one, two…). Figure 3-22 shows residuals classes.

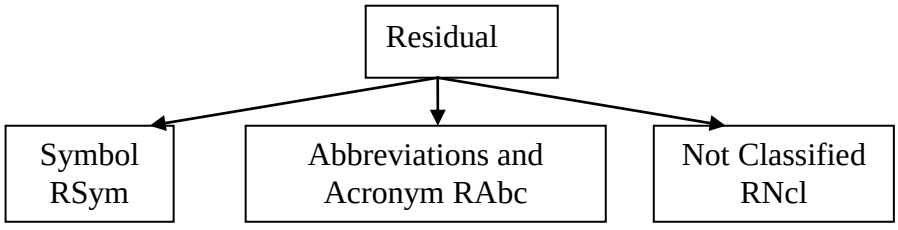


Figure 3-22: Residuals classes.

Punctuation category contains all punctuation symbols: “،”, “؟”, “...”, “:”, “؛”, “-”, “[”, “]”, “=”. All these have one class which is punctuation (CPnc).

Residuals and punctuations do not have features or meaning. It means that residuals and punctuations are not the same as noun, verb, nor particles; therefore, they only have one level.

3.5 Multilevel tagset

Residuals and punctuations do not have features or meaning, hence they have one level. For particles, there are two levels only, meaning and working. The same situation applies to verbs: they have two levels, type and features. The situation for nouns is different where there is a third level in addition to POS and feature. The first level of POS consists of the properties that do not be change when the position of the noun in the sentence is changed. The features of this POS are grammatical features (level two). The third level is for syntactic classes which are changed by changing the noun position in the sentence. It is well known, that the number of syntactic classes in Arabic is much larger than in English. The third level of classes of noun is shown in figure 3-23. These classes can be treated as additional features. We show the levels of POS tagset in Figure 3-24.

فاعل (Subject of a verb)	مفعول به (Object of a verb)	ضرف متصرف (Adverb)	منادى (Vocative)
نائب فاعل (Passive subject representative	مفعول مطلق (Cognate)	حال (Circumstantial accusative)	مضاف اليه (Possessive construction)
مبتدأ (Subject)	مفعول لأجله (Accusative of purpose)	تمييز (Specification),	بدل ,نعت (Apposition)
خبر (Predicate of a subject)	مفعول معه (Commutative object)	مستثنى (Excepted)	X NOT USED

Figure 3-23: syntactic classes of noun[16].

[16] We intend to design a tagset and build a POS tagger for Arabic. Level three is beyond what tagger needs and therefore I used the letter "X" to indicate an unused level for future use.

	First level	Second level	Third level (*not used in our practical tagset*)
Noun	Noun type which will not change at any position in the sentence	Grammatical features	Noun type which can change according to it's position in the sentence (mostly syntactic types).
verb	Verb type which will not change at any position in the sentence	Grammatical features	------
Particle	Particles working	Particles meaning	-------
Residuals	Residual symbol	------	------
punctuation	Punctuation symbol	------	------

Figure 3-24: the Levels of the proposed tagset

3.6 Practical representation of the proposed tagset

Practically, the tagset is representing classes and features in one block of symbols. Representation of tags in the proposed tagset is as follows:

1. Noun has the form: N+POS_ Number+Gender+Case+Structured
2. Verb has the form:

   V+POS_ Person+Number+Gender+Case+Structured+Certinity+Voice
3. Particles has the form: P+Working_Meaning
4. Residual has three tags: ROth, RSys or RAcb
5. Punctuation has one tag: CPnc

Appendix A shows a practical example of 186 tokens tagged with this tagset. Theoretically, the proposed tagset has 3552 tags (excluding the third level). Indeed, some of the tag combinations are impossible. By taking third level into account, the number of tags will increase to 14892.

3.7 Discussion

As we see, some researchers constructed tagsets based on English and missed some of the important features of Arabic. Other researchers created tagsets depending on the Arabic language and took some features from other languages, but those tagsets

didn't take all the important Arabic language features into account. There has been a tagset proposed that includes all Arabic language features, with many useless (redundant) tags.

Building a tagset, as large as possible to include all language features, and as small as possible in order to permit relatively efficient tagging, is a hard problem. We introduced a new multilevel Arabic tagset compatible with CA (Classical Arabic) and MSA (Modern Standard Arabic). It has almost all Arabic features and classes. Selecting classes, features and merging them is done carefully. The proposed tagset does not have interleaving. The third level of this tagset is beyond the range of this dissertation; therefore we will refer to its first two levels, only.

Chapter 4

Segmentation and Tokenization

4.1 Introduction

Tokenization is the task of separating out words (morphemes) from running text (Jurafsky & Martin) [54]. One of the mophemes typically corresponds to the word stem, and there ar ealso inflectional morphemes (Habash) [45]. We can use blanks (white space) to help in this task, but there are hard cases. This definition is valid for English, but for Arabic the situation is different. While discussing tokenization, it is important to remember that there is no single optimal tokenization. What is optimal for IR may not be optimal for SMT. Also, what is optimal for a specific SMT implementation may not be the same for another (Habash) [45].

Tokenization is a necessary and non-trivial step in natural language processing (Bird et, al.) [22] (Attia) [13]. It is closely related to the morphological analysis but usually it has been seen as an independent process (Chanod & Tapanainen) [28].

(Habash) [45] shows a number of different levels of tokenization schemes. It starts from simple tokenization which is limited to splitting off punctuation and numbers from words. Then orthographic normalization unifies various forms of letters. Then decliticization schemes split off clitics. The last can be done according to stem & affixial morphemes or lemmas & clitics.

In our work, there is a little distinction between segmentation and tokenization. Segmentation is related to splitting running text into sentences (sentence segmentation), into words (word segmentation) and the word to its segments, no matter how this word was constructed. On the other hand, tokenization is related to getting tokens from running text. But in most cases these two tasks overlap. In other words, segmentation is related to splitting all affixes and clitics[17] and tokenization is splitting clitics only with retriving the changed or the deleted letters resulting from the inflections. We take the segmentation process as splitting running text into sentences (sentence segmentation), into words (word segmentation) (Jurafsky & Martin) [54], and tokenization as splitting the words into morphemes.

In this chapter we propose a hybrid unsupervised method for Arabic tokenization, considered as a stand-alone problem. After getting words from sentences by segmentation, we use our own analyzer to produce all possible tokenizations for each word. Then, manually written rules and statistical methods are applied to solve the ambiguities. The output is one tokenization for each word. The statistical method was trained using 29k words, manually tokenized (data available from http://www.mimuw.edu.pl\~aliwy) from Al-Watan 2004 corpus (available from http://sites.google.com/site/mouradabbas9/corpora). The final accuracy was 98.83%.

4.2 Tokenization System

The whole pre-processing for Arabic tagging system consists of tokenization and analyzing. Figure 4-1 shows the whole pre-processing for tagging system. After completing all these stages, the final results are lemma and clitics with their features. We should note that lemma is an ambiguous term in Arabic and there is no consensus among the researchers about its definition. In this dissertation we depend on the definition in (Habash) [45]. In this chapter, we will focus on tokenization only.

[17] See section 4.7.1 for clitics definition.

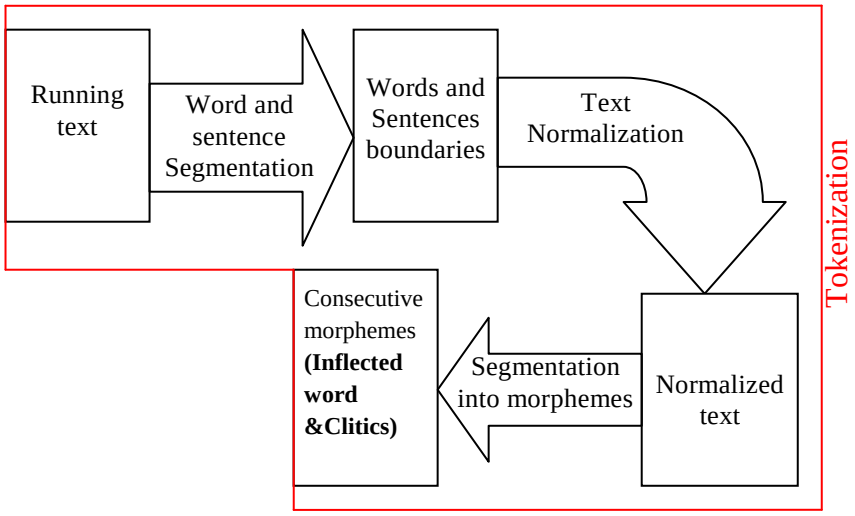


Figure 4-1: The Tokenization as pre-processing task for tagging process. The output is **inflected word + clitics** for each word.

4.3 Related Work

In some works (e.g. MADA+TOKEN (Habash) [51], BAMA (Buckwalter) [25][26], AMIRA (Diab) [32], Xerox Arabic Morphological Analyzer and generator (Beesley's) [17][18], Sakhr's Arabic Morphological Analyzer (Sakhr Software) [81], Khoja's stemmer (Khoja) [58] this step of natural language processing is performed (partially or completely) as a preprocessing step.

(Benajiba) [20] presents two segmentation schemes that are morphological segmentation and Arabic TreeBank segmentation. He shows their impact on an important natural language processing task, which is mention detection. Experiments on Arabic TreeBank corpus show 98.1% accuracy on morphological segmentation.

(Lee) [66] depends on the word representation as prefix*-stem-suffix*. The algorithm uses a trigram language model to determine the most probable morpheme sequence for a given input. The language model is initially estimated from a small manually segmented corpus of about 110,000 words. The resulting Arabic word segmentation system achieves around 97% exact match accuracy on a test corpus containing 28,449 word tokens.

The systems of Benajiba [20] and Lee [66] deal with stem rather than lemma. According to Habash [45] stem is not a legal Arabic word form, unlike lemma.

In AMIRA (Diab) [32] and MADA+TOKEN (Habash) [51] are packages and the tokenization is not a separate task. They use Support Vector Machine (SVM), but Habash [51] uses morphological analyzer with SVM. They have accuracy of tokenization 99.12% and 99.21% respectively.

4.4 Word and Sentence Segmentation

4.4.1 Sentence segmentation

It is the first step in text processing, a crucial one. Segmentation a text into sentences is generally based on punctuation (Jurafsky & Martin) [54]. In Arabic, estimating boundaries of a sentence is a relatively simple task, about as difficult as in English. The average number of words per sentence is larger than the average in English, but it does not affect the segmentation process. The sentence boundaries and phrase boundaries can be estimated according to Arabic punctuation marks which are ،,؟, ..., :, ., ؛ ,"" ,- ,[ ] ,=.

4.4.2 Word segmentation

Word segmentation is the process of getting words from text. The space is a good separator for this task but it will not work in special cases, such as compound words. Some compound words are written with a space in the middle even though they are single words. Such cases must be solved at this stage. For example the word "IslAm |bAd"-"إسلام آباد" (**Islamabad**) is a name of a city in Pakistan. It means that we must have knowledge base with such words. After solving this problem, this stage is relatively easy. There is another difficulty, when a few words are attached together without spaces, which can happen when the first one ends with one of the letters "w"-"و", "d"-"د", "r"-"ر", "z"-"ز", "*"-"ذ". It is formally a mistake, but may happen when dealing with informal texts. Our system assumes to work with correct texts hence we do not offer any solution of this particular problem.

4.5 Normalization

Orthographic normalization is a basic task which reduces noise in the data (Habash) [45]. This is true regardless of the task: preparing parallel text for machine translation, documents for information retrieval or text for language modeling. Normalization can be Tatweel removal (removing Tatweel symbol),

diacritic removal and letter normalization (variant forms to one form conversion). Figure 4-2 shows letter normalization example.

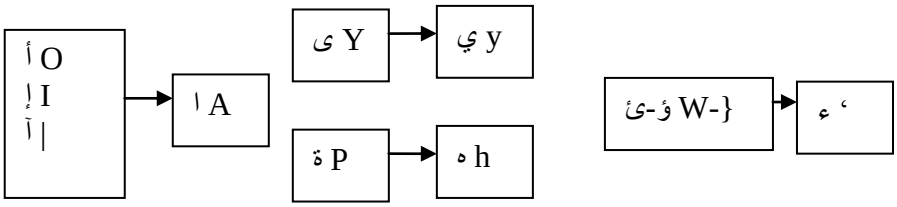


Figure 4-2: An example of Arabic letter normalization

This normalization will help us in searching or matching process but after this stage, the normalization process will increase the ambiguity in tokenization. For example, if we normalize "P"-"ة" (Taa-Marbuta) to "h"-"ه" (Ha), the latter will be tokenized as a pronoun. For this reason, in our work we consider normalization as a temporary stage for matching and searching the dictionaries.

4.6 Arabic Tokenization

Arabic words are often ambiguous in their morphological analysis. This is due to Arabic's rich system of affixation and clitics and the omission of disambiguating short vowels and other orthographic diacritics in standard orthography ("undiacritized orthography"). On average, a word form in the ATB has about 2 morphological analyses (Habash & Rambow) [46].

Arabic word is of the form [Proclitics] + [inflected word] + [Enclitics]. Then, tokenization here is similar to word segmentation in Chinese, where Arabic word corresponds to a sentence in Chinese[18].

4.7 Arabic word form

In written, it is possible that a single word has two or more part of speech (POS) categories. It leads to problems in stemming and segmentation. Let's consider the word "wbsyArthm"-"وبسيارتهم" (and by their car). Is it a word? How is it constructed? According to the classical[19] definition of a word, it is a word but, as we can see, it has four POSs.

[18] Chinese does not delimit words by white-space. Word segmentation is therefore fundamental for other language processing tasks in this languages (Peng et, al.)[76].
[19] The word is a sequence letters enclosed by two spaces

In this chapter we will distinguish constructing of a word from a number of POSs and the inflected word (construction perfect, imperfect, imperative, mood, person and so on). I.e., we will distinguish clitics and affixes.

Arabic clitics attach to the inflected base word (see the next Section 4.7.1) in a strict order that can be represented as follows, using general class names (Habash) [45]:

[QST+ [CNJ+ [PRT+ [DET+ BASE +PRO]]]][20]

where QST is question, CNJ is conjunction, PRT is particle, DET is determinant, BASE is base of the word, and PRO is pronouns, respectively.

In a more general way, we can represent the word as:

BASE + affixes + clitics

≡ lemma+ morphological features+clitics

≡ stem + affixes + clitics

≡ inflected word +clitics

The previous example "wbsyArthm"-"وبسيارتهم" will be "w+b#syArp#hm" according to the last form where: w, b and hm are clitics and syArp is the inflected word.

Some works do not differentiate between affixes and clitics, assuming the Arabic word generally to be of the form (prefixes + stem + suffixes). In our work, we will focus on the form (inflected word + clitics), where inflected word consists of lemma and morphological features. This will help us encoding word features and POS without doing an unwanted segmentation.

4.7.1 Word Clitics

Clitic is a unit whose status lies in between that of an affix and a word. The phonological behavior of clitics is like affixes; they tend to be short and unaccented; their syntactic behavior however is more like words, often acting as pronouns, articles, conjunctions or verbs (Jurafsky & Martin) [54]. **Clitics** can be **proclitics** which precede the word (like a prefix) or **enclitics** which follow the

[20] Any transliteration written in English should be read from left to right, while the corresponding Arabic original phrase should be read from right to left.

word (like a suffix). **Proclitics** can be preceding the verb, noun, pronoun and particles**.** Figures 4-3 & 4-4 list almost all known combinations of verbs and nouns proclitics, respectively. There are three levels of verb proclitics, always attached in the same order. The use of them is optional. For noun the structure is similar, but there are four levels.

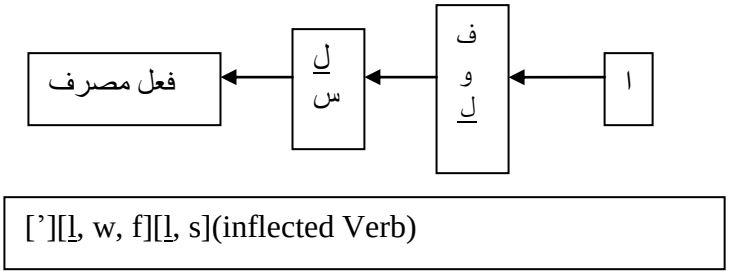


Figure 4-3: Verb proclitics.

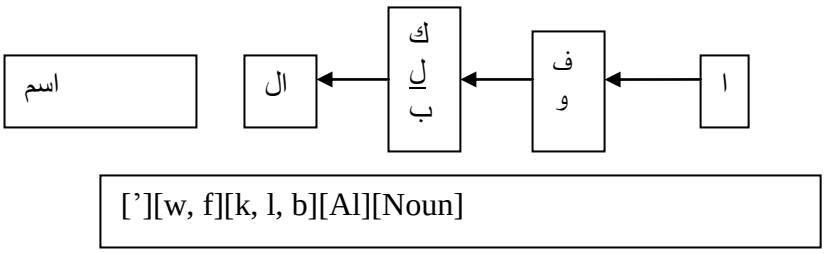


Figure 4-4 Noun proclitics.

Figure 4-5 shows cliticization of attached pronouns[21] with particles. Selecting which is the base (inflected word) depends on the priority shown in Figure 4-5 by number. The numbers (1, 2 and 3) which are used in figure 4-5 are the priority of taking the base of the word. If one word from box 1 exists in the word, then it is the base and the remaining ones are clitics; else, if a word from list box is present, then it is the base and the other ones are clitics; else the word from box 3 is the base and there are no proclitics. Note that at least one word from those lists must be present. For example “افإنهم” “AfInhm” “then, are that they” is cliticized as follows: “A”-“ا” (are/is) and “f”-“ف” (then) are proclitcs, “In”-“إن” (that) is the base and “hm”-“هم” (they) is an enclitic. The book (Habash [45], pages 48-50) is a good reference for other special cases in cliticization.

The particles can appear combined for constructing words, but the easy way for dealing with them is by taking these combinations as stop words.

[21] In Arabic there are two types of pronouns: attached to a word (us, me..) and separated (I,we…).

Enclitics follow verb or noun. The enclitic "nA"-"نا" (we-our) is ambiguous and has two possible roles (either a clitic or an inflection suffix). For example the word "qtlnA"-"قتلنا" can mean "we killed" or "he killed us". "nA"-"نا" is an affix in the first context and an enclitic in the second context.

All enclitics are pronouns and therefore pronouns themselves don't have enclitics. Figure 4-6 shows all common enclitics for nouns and verbs with their order. They are optional.

This set of clitics and their order of precedence (summarized here and described also in other papers and books) are the base of our algorithm. Adding a few rules for deleting unwanted combinations of clitics we can get a good segmentation program, as we will see in the implementation section later in this chapter.

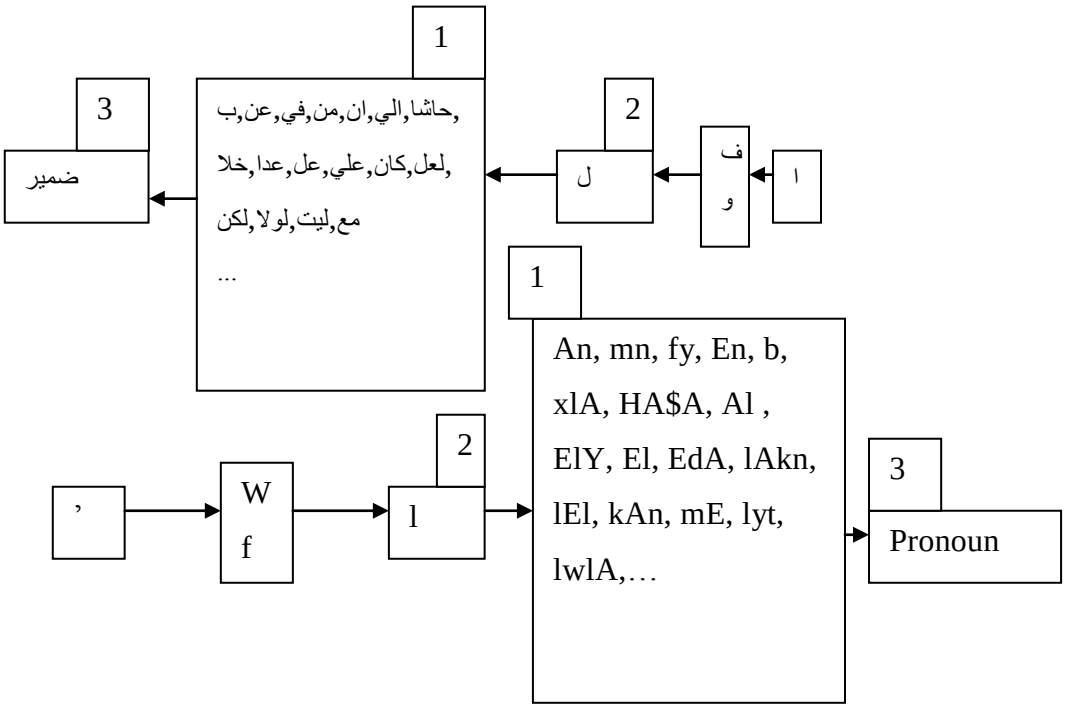


Figure 4-5: Proclitics for pronoun and pronoun as an enclitic according to the priority number of taking the base.

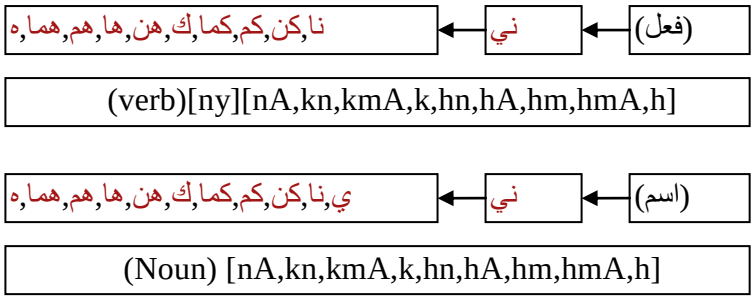


Figure 4-6: Enclitics for Noun and Verb

4.8 Tokenization and segmentation techniques and schemes

Habash [48] shows that tokenization techniques can be as simple as regular expressions and/or as complex as morphological analysis (form-based and functional). The main classification of tokenization algorithms is into supervised and unsupervised ones. Manual analysis of text and writing custom software, unsupervised Language Model Based (Lee et al.) [66] are examples of unsupervised methods. Annotating the sample corpus with boundary information and using machine learning (ML) is an example of a supervised method. The other classification is into language dependent (methods used for one language or group of languages, there are many methods of this type) and language independent methods.

Arabic has a middle level of segmentation complexity; it is between English (and similar languages) and Chinese (and similar languages). In Arabic words are typically separated by spaces (as in English), but it is possible that an Arabic word is a whole sentence, like in Chinese. Therefore we should use a hybrid method for dealing with segmentation or split the segmentation task into two steps. The helpful thing is that the forms of Arabic words are known, which simplifies the segmentation of words when compared to Chinese, where one has to apply segmentation to sentences.

Schema defines what the target tokenization is (Habash) [45]. The same paper lists some examples of schemes used in tokenization of Arabic. In this dissertation we use scheme D3+ LEM. D3 (decliticization of degree 3) is a scheme that splits off clitics: the class of conjunction clitics (w+ and f+), the infrequent interrogative

clitic, the class of particles (l+, k+, b+ and s+), the definite article Al+ and all pronominal enclitics. LEM reduces every word to its lemma.

4.9 Challenges of Arabic tokenization

There are many challenges to Arabic tokenization. The complexity of the morphology together with the under-specification of the orthography creates a high degree of ambiguity (Habash et, al.) [51]. Some of these ambiguities can be summarized by:

- Orthography problems resulting from writing the letter in ambiguous case as in "Y"-"ى" and "y"-"ي" or unification of some forms of a letter as in "A"-"ا", "O"-"أ", "I"-"إ" and so on.
- Encliticization of a word ending with "P"-"ة":

  "jmEthm"-"جمعتهم" (collect them) → جمعت + هم

  "jmEthm"-"جمعتهم" (their Friday) → جمعة + هم

- Encliticization of a word ending with "Y"-"ى":

  "mstwY"-"مستوى" (level) + "k"-"ك" (your)

  → "mstwAk"-"مستواك" (your level)

- "nA"-"نا" and "y"-"ي" are ambiguous and can be either enclitics or suffixes. (see section 6.1).
- Normalization adds ambiguity, for example normalizing "P"-"ة" to "h"-"ه" will create false enclitics: the word "Amp"-"امة" (nation) after normalization will become "Amh"-"امه", then if we apply the tokenization to the last word, it will become "Am+h"-"ام + ه" (him mother) but the right tokenization is "امة" "nation".
- Ambiguity results from decliticization of "l"-"ل", "A"-"ا" and "Al"-"ال" (the).

All these and other ambiguities are solved during tokenization stage in our system.

Another class of problems resulting from morphology is solved in this stage. For example the word "HmlwnA"-"حملونا" (they raise us) after tokenization will

be "HmlwA+nA"-"حملوا+نا" where the tokenizer adds the removed letter resulting from morphological rules.

There are other encoding problems where the same letter is written in different shape with different code. It is solved in this stage, as well. For example "zmlA}y"-"زملائي" (my colleagues) after tokenization will be "zmlA'+y"-"زملاء+ي" and so on.

Some of the ambiguities in POS tagging are solved already during tokenization. For example the words "bktbnA"-"بكتبنا" (by our books) after tokenization will be "b+ktb+nA"-"ب+كتب+نا" because it has preposition "b"-"ب" (by). The other tokenization is "b+ktbnA"-"ب+كتبنا" (by+we write) which is rejected by the tokenizer, because an inflected verb cannot appear after a preposition.

4.10 Our approach

We use a hybrid method for tokenization which is a combination of unsupervised method which depends on rules for getting segments, and statistical method for solving ambiguities. Our algorithm works as follows:

Task 1: As a preparation to the segmentation process, we first compute all verb, noun and pronoun proclitics and enclitics storing these combinations in lists. Then, the text is segmented into sentences and the sentences into words according to space and Arabic punctuations. Segmenting the words into clitics & bases is done by analyzer which produces all possible segments for each word. After this stage every word may have several segmentations.

Task 2: Now we remove noise introduced in the first task. We do so by deleting segmentations which produced one letter words with proclitics and enclitics (which is impossible in Arabic)[22] and duplicate segmentations (which may result from segmenting the same word treated once as a verb and once as a noun). We also remove segmentations whose inflected word is not in the dictionary (constructed separately from many resources). However, if all produced segmentations of a word should be removed, they are all passed to Task 3 for

[22] See section 5.8 more details for constructing the dictionary.

special treatment. Words whose segmentations are not all removed are passed to Task 4.

Task 3: Because the used dictionary does not cover all words in the language, there are many unknown words whose segmentations are passed from Task2 and must be processed here as out of vocabulary (OOV). We first choose the segmentations which give the largest number of letters in the proclitics and enclitics, and among these we choose ones that have the least number of proclitics and enclitics. If this does not yield a unique segmentation, the choice is not made and the possible segmentations are transferred to Task 4.

Task 4: Because the system may produce many segmentations for one word, in order to get one segmentation for each word, we select the segmentation with the least number of segments. If this still does not produce a unique segmentation, we use a method similar to that of Task 3. From the candidate segmentations we select the segmentations which give the longest possible sequences proclitics and enclitics, and among these we choose ones that have the least number of proclitics and enclitics. If this does not yield a unique segmentation, we choose the first one encountered.

Task 5: We eliminate, using statistical estimation, ambiguity of results of Task 1. This task is done in parallel with Tasks2, 3 and 4. This task is described below in Section 4.11.

Task 6: Smoothing or correction rules are used to reduce errors from the previous tasks.

For example, we add the following rule for distinguishing between a word ending with "t"-"ت" (normal Taa) or "p"-"ة" (Taa Marbuta):

IF *((the base word has Taa AND has enclitics) AND (has a proclitic of type preposition OR the previous base is a preposition))* ***THEN*** *Change Taa to Taa Marbuta.*

There are many other similar rules used in this task derived from Arabic grammars (AL-Bidhani) [3] (Al-Rajhi) [10] (Al-Hamlawy) [7] (Al-Galaiini) [6].

4.11 Applying statistical improvement

Our philosophy of using statistical support is the same as the one we use later in POS tagging system. Assume we have a sentence: w_1 w_2 … w_n with n words. Let the set of possible tokenizations of word w_i in this sentence be $\{s_1 \ldots s_j\}$, where j is the number of segmentation[23] of this word. Now we can apply any statistical method, like HMM used for tagging, for tokenization.

We have two facts: in our approach, first we used dictionary and rules for tokenization and solving ambiguities. Bigrams are used, and we do not consider *n*-grams for *n*>2. The bigrams equation which we used practically is:

$$\hat{s}_i = \arg\max_{s_i} \; p(w_i \mid s_i)\, p(s_i \mid s_{i-1})$$

$P(w_i \mid s_i)$ is probability of i^{th} word given the segmentation. $P(s_i \mid s_{i-1})$ is the probability of the segmentation given the previous segmentation.

4.12 Results

After applying all the previously described simple methods, we got the following results, in which we used bigrams on 45 files[24] with 29k words.

Without statistical support and without Task 4 the recall is 0.9877462, precision is 0.8617793 and F-measure is 0.920473. Without statistical support (one choice for each word using Task 4) the accuracy is 0. 9802977. With statistical support (one choice for each word) ten-fold cross-validate accuracy is 0.9883473.

In our tests, tokenizations "#Asrt#hA"-"#اسرت#ها"[25] and "#Asrp#hA"-"#اسرة#ها", "#nrA#hA"-"#نرا#ها" and "#nrY#hA"-"#نرى#ها" are assumed to be errors even though they are only orthographically wrong. In general, any change to the ending letter of the word resulting from morphology, if it is not compatible with the original letter, is assumed to be an error. Practical tokenized Arabic text

[23] s_1, …, s_j are segmentations, not segments. I.e., each one of these segmentations consist of one or more segments.

[24] The data was chosen randomly from Al-Watan 2004 corpus (available from http://sites.google.com/site/mouradabbas9/corpora). The sentences have been tokenized manually by ourself.

[25] Practically the tokenized text has format: proclitics#inflectedWord#enclitics. If there are more proclitics/enclitics, they are separated by +.

and its transliteration are shown in Figures 4-7 and 4-8, respectively[26]. Comparing with other works, the best known tokenization results have accuracy 99.12% and 99.2 % (Diab [32] and Habash [51], respectively) on the data of ATB. They did not solve the following problems: sometimes they take "AL"-"ال" as a part of a word, not as a clitic, which leads to a decreased level of ambiguity between "A+L"-"ا+ل" and "AL"-"ال" clitics (i.e., it increases accuracy). Next, in most of cases, they did not manipulate the letter changing due to morphology. I.e., the errors in the two examples in this section are considered to be correct in their approaches. Their algorithms are data dependent because they use statistical method. Our method without statistical improvement is only marginally worse, being data independent.

4.13 Discussion

We can see that we collect more than one method for solving ambiguity in tokenization. We introduce simple and effective methods for making decisions in tokenization, achieving high accuracy Arabic tokenization system. Our approach solves most ambiguities in tokenization. The tokenization is a separate task. It can be an efficient tool for annotating large corpora. If an extremely high accuracy is needed, wrong cases can be corrected manually. We do so measuring the accuracy of the next steps in our tagging system.

[26] The 45 tokenized files are freely available from the website: http://www.mimuw.edu.pl/~aliwy.

#مرة# #،# و#قبل# #سنتين# #،# #كتبت# #عن# ال#عراق# #الذي# #سوف# #يعمل# #على# #تغيير# ال#عالم# #،# #هل# #هذه# #كلمة# #كبيرة# و#مبالغ# #في#ها و#رب#ما #لم# #يسعف# ال#تعبير# #على# #وجه# ال#دقة# و+ال#وضوح# #من# #ان# ال#عراق# ال#قديم# ال#كامن# #تحت# ال#رمال# و+ال#ليشن# #،# #هو# #ذاك# #الذي# #سوف# #يغير# ال#عالم# #،# و#إذا# #ارتأينا# ال#فكرة# #في# ال#واقع# ال#فعلي# #،# ف#أن# ال#عالم# و#من# #خلال# #عشرة# #آلاف# #تل# #آثاري# #،# #لم# #يجر# ال#تنقيب# #في#ها ب+ال#عراق# #،# #سوف# #يمنح# #اكاديميات# ال#ارض# #فرصة# #علمية# ل#استعادة# و#من# #ثم# #تغيير# #تصورات#ها و#مفاهيم#ها #في# #مختلف# #قضايا# و#شؤون# ال#حياة# و+ال#تاريخ# #..# #اذن# ف+ال#عالم# س#يغير# #نفس#ه #من# #خلال# ال#عراق# #مثل#ما #تغير# #حين# #اعاد# ال#ماركسيون# ال#نظر# #في# #تصورات#هم #عن# #نمط# ال#انتاج# ال#اسيوي# و#فكرة# #نشوء# ال#طبقات# #حال#ما #اكتشف# ال#استشراق# #مدنا# #مثل# #سومر# و#بابل# و#آشور# #،# و#تحروا# #عند# #تفاصيل#ها #انظمة# #تسجيل# ال#عبيد# و+ال#اجراء# و+ال#موظفين# و#اشكال# #تنظيم# ال#عمل# و#ادارة# ال#دولة# #،# و#لو# #كان# ال#استشراق# #في# #زمن# #ماركس# و#انجلس# #قد# #توصل# #الى# #اكتشاف# #تلك# ال#مدن# و#دقائق#ها ال#يومية# ل#ما# #كتبا# #شيئا# #عن# ال#ارض# ال#مشاعة# و#مشكلة# ال#بزل# #اللذين# #حالا# #دون# #ارتقاء# ال#ملكية# ال#فردية# و#منعا# #من# #قيام# ال#صراع# ال#طبقي# #،# و#رب#ما #كانت# ال#ماركسية# #غير#ها #في# ال#نظر# #الى# ال#شرق# و+ال#غرب# #لو# #كان# ال#استشراق# #في# ال#مستوى# ال#تفصيلي# ك#ما# #جاء# #بعد# #ماركس #.# #

Figure 4-7: Sample of Arabic tokenized text

#mrp# #,# w#qbl# #sntyn# #,# #ktbt# #En# Al#ErAq# #Al*y# #swf# #yEml# #ElY# #tgyyr# Al#EAlm# #,# #hl# #h*h# #klmp# #kbyrp# w#mbAlg# #fy#hA w#rb#mA #lm# #ysEf# Al#tEbyr# #ElY# #wjh# Al#dqp# w+Al#wDwH# #mn# #An# Al#ErAq# Al#qdym# Al#kAmn# #tHt# Al#rmAl# w+Al#ly$n# #,# #hw# #*Ak# #Al*y# #swf# #ygyr# Al#EAlm# #,# w#I*A# #ArtOynA# Al#fkrp# #fy# Al#wAqE# Al#fEly# #,# f#On# Al#EAlm# w#mn# #xlAl# #E$rp# #|lAf# #tl# #|vAry# #,# #lm# #yjr# Al#tnqyb# #fy#hA b+Al#ErAq# #,# #swf# #ymnH# #AkAdymyAt# Al#ArD# #frSp# #Elmyp# l#AstEAdp# w#mn# #vm# #tgyyr# #tSwrAt#hA w#mfAhym#hA #fy# #mxtlf# #qDAyA# w#$Wwn# Al#HyAp# w+Al#tAryx# #..# #A*n# f+Al#EAlm# s#ygyr# #nfs#h #mn# #xlAl# Al#ErAq# #mvl#mA #tgyr# #Hyn# #AEAd# Al#mArksywn# Al#nZr# #fy# #tSwrAt#hm #En# #nmT# Al#AntAj# Al#Asywy# w#fkrp# #n$w'# Al#TbqAt# #HAl#mA #Akt$f# Al#Ast$rAq# #mdnA# #mvl# #swmr# w#bAbl# w#|$wr# #,# w#tHrwA# #End# #tfASyl#hA #AnZmp# #tsjyl# Al#Ebyd# w+Al#AjrA'# w+Al#mwZfyn# w#A$kAl# #tnZym# Al#Eml# w#AdArp# Al#dwlp# #,# w#lw# #kAn# Al#Ast$rAq# #fy# #zmn# ArtqA'# Al#mlkyp# Al#frdyp# w#mnEA# #mn# #qyAm# Al#SrAE# Al#Tbqy# #,# w#rb#mA #kAnt# Al#mArksyp# #gyr#hA #fy# Al#nZr# #AlY# Al#$rq# w+Al#grb# #lw# #kAn# Al#Ast$rAq# #fy# Al#mstwY# Al#tfSyly# k#mA# #iA'# #bEd# #mArks# #.#

Figure 4-8: Transliteration of Arabic tokenized text

Chapter 5

5.1 Introduction

The Arabic language is based on inflection and derivation, and words have many different forms that result from these procedures. Therefore extracting lemma is a hard problem for Arabic language. As a consequence, many researchers chose to deal with the stem, which is easier to extract, rather than with lemma. For example, in broken (abnormal) plural of nouns the word changes completely. In lemmatization the original form must be found, in stemming it is not necessary and is therefore easier.

In this chapter we build an Arabic analyzer which has two goals: the first is extracting POS and features of the word. The second is extracting the lemma of the word. These two goals are implemented in parallel. We built a dictionary as a tool for achieving these two goals.

The proposed analyzer is not intended for independent use because it was designed and implemented as a preprocessing stage for Arabic tagging system and, using the context of the word, it will reject some analyses, saving tagger's work.

5.2 Lemma, stem and root

When we deal with the analyzer, we must differentiate among three terms: Lemma, Stem and Root. They have different meaning. The lemma is the **canonical form**, **dictionary form**, or **citation form** of a set of words. The stem is the part of the word that never changes even when morphologically inflected[27]. The root is the original letters[28] of the word. Moreover, the term "root" is ambiguous in Arabic language: some researchers consider it to be the original letters, while others to be the imperative verb in 3rd masculine.

When we deal with the root, then the derivational and inflectional morphology is taken into account. When we deal with lemma, then only inflectional morphology will be taken into account. When we deal with stem, then part of inflectional morphology with part of derivational morphology will be taken into account. For example: changing the whole word will not be taken into account as broken plural. Figure 5-1 shows the difference between them with adding "**number**" feature to the word "kitAb"-"كتاب" (book).

Word	kitAb كتاب (book)	kitAbAn كتابان, kitAbYn كتابين (two books)	kutub كُتُبْ (books)
Root	ktb ك ت ب	ktb ك ت ب	ktb ك ت ب
Stem	kitAb كتاب	kitAb كتاب	kutub كتب
Lemma	kitAb كتاب	kitAb كتاب	kitAb كتاب

Figure 5-1: Lemma, stem and root of the word "book" with adding number feature[29].

We can summarize the difference between stem, root and lemma in the following points:

1. Stemming reduces word-forms to (pseudo) stems, whereas lemmatization reduces the word-forms to linguistically valid lemmas. Getting the root is done by reducing word-forms to original letters (root).

[27] In Arabic the changes of vowels will be taken into account in stemming.
[28] See Section 2.4 for more details about original letters.
[29] The plural is broken for this noun.

2. Extracting stem and root is relatively simple and can be done by deleting affixes. Extracting Lemma is more sophisticated and must refer to dictionary in some cases.
3. The root and stem are not valid words but lemma is.
4. More than one lemma can have the same stem; more than one stem can have the same root.

In our work, for verbal classes the lemma is 3rd masculine imperative verb. Lemma for the noun classes is the singular masculine, and if it does not exist, the singular feminine. For particles, strictly speaking, there is no lemma, so for unification we define it to be the particle itself.

5.3 Morphological analysis with lemma extraction for Arabic

Morphology is the branch of linguistics that deals with the internal structure of words (Al-Sughaiyer & Al-Kharashi) [12]. Then morphological analysis is the task to discover the possible structures of a given word and represent them in a desired format. Morphological analysis for Arabic was intensively studied by the researchers; some of those works are listed in section "Related work". From the computational point of view we talk about possible algorithms and automated techniques of performing morphological analysis.

Morphological analysis for Arabic can be done in two stages according to the word structure:

1. Dealing with clitics: splitting the words to its morphemes which can be done by tokenization.
2. Dealing with affixes and internal structure (inflected word): One or both of the following:
    a. Extracting the origin of the word (root, stem ...).
    b. Extracting the attributes of the word (POS, gender, number...).

Morphological analyzer, depending on the form of the extracted origin of the word, can be:

1. Root-based
2. Stem-based
3. Lemma-based

According to the approaches used, Morphological analyzer algorithms can be classified as follows (Al-Sughaiyer & Al-Kharashi) [12]:

1. **Table lookup approaches (simple method)**: all valid natural Arabic words along with their morphological decompositions are stored in a huge table. A given word is analyzed simply by accessing the table and retrieving information associated with that entry.
2. **Linguistic approaches (sophisticated rule-based)**: utilize linguistic rules that have been derived through deep analysis of Arabic morphological systems.
3. **Combinatorial approaches (brute force)**: all combinations of letters of a given word are tested and compared against a list of roots.
4. **Pattern-based approaches (less sophisticated rule-based)**: utilizes the apparent symmetry of generated natural Arabic words.

Table lookup approaches are typically not sufficient alone, because it is practically impossible to collect all forms of all words of Arabic. But it can be the best approach for the irregular forms. The second type of approach requires deep knowledge of linguistics, especially of the word construction rules, and any omission reduces the quality of the results. The third class of approaches does not need so deep linguistic knowledge, but it can give unwanted analyses. The fourth one is similar to the second approach, but it needs less knowledge. On the other hand, it requires collecting all possible patterns including the very rare ones, which can in turn produce wrong analyses in some cases.

There are many other classifications of morphological analysis algorithms for Arabic (see (Al-Sughaiyer & Al-Kharashi) [12]), but we chose the above one as the most useful for us.

It is clear that there is no single ideal approach to Arabic morphological analysis, but it is also clear that the application which will use the analyzed text is an important factor to consider when choosing the approach to adopt.

In our work the analysis is used to extract the word attributes, such as POS, gender, number, etc.

Lemmatization is the process of relating a given textual item to the actual lexical or grammatical morpheme (Dichy) [35]. It is the process of mapping from a word form to a lemma (Jurafsky & Martin) [54]. From the definition of morphological analysis, lemmatization can be a part of it. In our work it is limited to extracting the lemma from the word. Without using lexicon, lemmatization cannot be done with sufficiently high accuracy for many reasons which will be listed in the next section. We do not attempt word sense disambiguation (WSD) in our system hence, we accept more than one lemma for a word.

5.4 Challenges for lemmatization and analyzing

Due to the morphological complexity of the Arabic language, morphological analysis with lemma extraction is a very challenging task. Arabic language is regular in most cases of inflection and derivation, which leads to a relatively easy generation process. However, for irregular forms, it is more complicated. This difficulty grows rapidly also when a nonvowelized text is used[30]. Then the analysis process has to consider all possible vowelizations and produce all possible correct analyses for them. This huge number of analyses for each nonvowelized word leads to much increased probability of producing some wrong analyses among them.

The main challenges are:

1. A nonvowelized word can correspond to many vowelized words and therefore to many possible lemmas: for example the lemmas for the word “ktb”-“كتب” can be “kataba”-“كَتَبَ” (write), “ktAb”-“كتاب” (book) and “kat~aba”-“كَتَّبَ” (dedicated to write).
2. A normalized word can correspond to many unnormalized words and therefore to many possible lemmas: for example the lemmas for the word “An”-“ان” can be “On”-“أن”, “In”-“إن” and “|n”-“آن” in unvowelized case.

[30] Traditionally Qur’ān is vowelized, and so are children’s books. The rest of present day texts are nonvowelized.

3. Deleting or changing some letters, even in regular forms. For example the lemma for the word "yqwl"-"يقول" (he say) is "qAl"-"قال" (said).

4. Words whose grammatical lemma ends or begins with a sequence of letters identical to an affix. The mistake may occur when the attributes are extracted from the affixes. For example the letters "wn"-"ون" could be falsely interpreted as a suffix and deleted from the word "mrhwn"-"مرهون" (pawned). Similarly, the letters "An"-"ان" in the proper noun "EdnAn"-"عدنان" could be interpreted as a suffix. Similarly, the letter "t"-"ت" in the common noun "tEAwn"-"تعاون" (cooperation) could be interpreted as a prefix.

5. Complete change of the word in regular and irregular cases: broken plural is often an example of this phenomenon. The best solution in this case is to use a dictionary.

6. Transliterations of foreign words. Many foreign words, for instance foreign proper nouns, have more than one form of Arabic transliteration, which affects the analyzing process.

In our complete system clitics are dealt with during tokenization stage, and hence are not listed here.

5.5 Analyzing as preprocessing

Arabic analyzing is the second preprocessing step, after tokenization step, of the whole tagging system which we propose. Therefore we suppose that the input word to analyzing is an inflected word or clitics as in Figure 5-2. The output of this stage will be lemma, POS and features in case of nouns and verbs, meaning and working in case of particles[31].

[31] See chapter 3 for more details on our tagset.

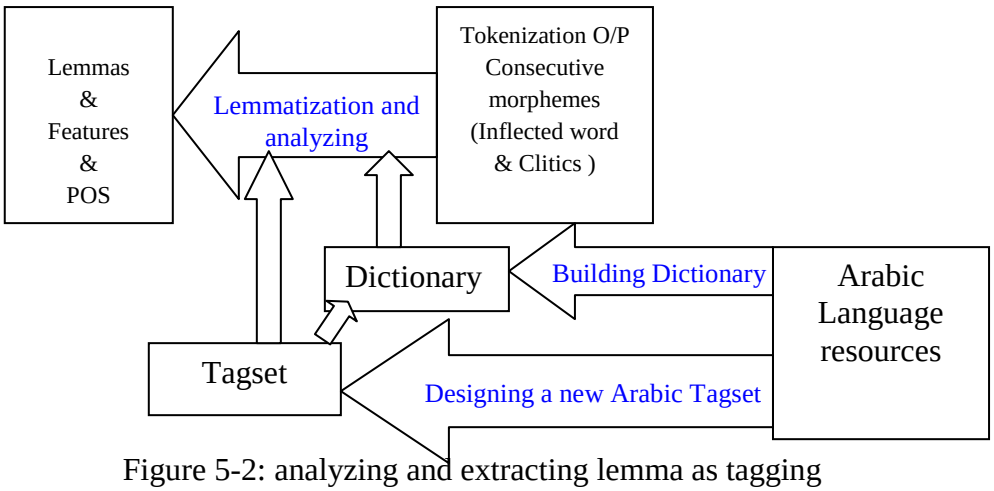


Figure 5-2: analyzing and extracting lemma as tagging preprocessing

Most of researchers depend basically on patterns for extracting root or stem but the pattern, in most cases, is not an efficient way for extracting lemma from the word. There is no any standardization for producing lemma from the word form in most cases.

We use our own lemmatizer and analyzer and rather than existing morphological analyzers for the following reasons:

1. We proposed a new Arabic tagset and existing morphological analyzers will not extract all the POSs and the features consistent with this tagset.
2. We deal with lemma instead of the root and stem.
3. We want to implement a complete tagging system.
4. Most analyzers mix segmentation and analyzing in one stage but we separate them into different tasks.

5.6 The proposed analyzing Approach

The item to be analyzed is a word without clitics (inflected word alone) or clitics alone, but all of them are known to the analyzer, because we assume that we are processing text and therefore the context of the present word is known to the analyzer.

Known words processing: no processing is needed because the lemma and features are in the dictionary.

Unknown words processing: we have more things to do. Unknown words are more likely to be nouns, because we use a large and fairly complete database of inflected verbs in the dictionary. As we mentioned previously there are many classes of nouns which are closed sets (like, e.g., relative nouns). The open classes of nouns are: proper, common, adjectives including genealogical and reduced nouns[32].

We will explain the construction of the dictionary in the next section. Now, we will focus on processing of unknown words.

5.6.1 Unknown words processing

Our approach to processing of unknown words is to do the most likely analyzing, without exhausting all possibilities. The main steps for unknown words processing are:

1. Extracting POS possibilities.
2. Extracting lemma and features.

The rules in the next sections are in general of four types: (i) strict positive rules, where if the condition is satisfied, then there is only one possibility for POS and features. (ii) Non-strict positive rules ("seems to be"), where if the condition is satisfied, then the POS and features are added to the list of possible ones, but other possibilities are not ruled out. (iii) Strict negative rules, where if the condition is satisfied, then some combinations of POS and features are ruled out, even if they were or will be added to the list of possible ones by non-strict positive rules. (iv) Non-strict negative rules ("seems not to be"), where if the condition is satisfied, then this combination of POS and features is an unlikely one.

It is a very interesting problem to find a good decision in a case of a word for which these rules produce a number of, perhaps contradictory, non-strict positive and non-strict negative indications. However, it seems that the method of resolving this problem has indeed little impact on the final accuracy of the analyzer, and therefore a very simple method is used, which treats non-strict negative rules as strict ones, in the absence of any strict positive indication. This

[32] Our tagset has 15 subclasses of noun.

reduces the number of possible analyzes, which is beneficial. We did not encounter any case of a conflict of a strict positive and a strict negative rule.

5.6.1.1 Extracting the POS possibilities

First we need to know the main POS to the word (noun, verb or particle); however, particle can be eliminated because the particles form a closed set. So really we have only two possibilities: noun and verb. Then we will extract the POSs according to our tagset.

1. **Extracting the main POS**: We must decide: verb or noun in this step. It can be done by applying the following classes of rules:
    a. Clitics rules. For example the definition particle "Al"-"ال" "Al" (the) appears with noun only.
    b. Affixes and word structure rules. For example, the letter "p"-"ة" appears in nouns only.
    c. Context rules. For example: verb cannot follow another verb.
    d. If none of the above rules is applicable, we assume by default that the word is a noun.
5. **2. Extracting the POSs according to our tagset:** it can be done separately for verbs and noun subclasses:
    a. Identification of past, present, and imperative forms of verbs is achieved by:
    i. Clitics: for example if the word has prociltic "s"-"س" (will) it must be a present tense verb.
    ii. Affixes: for example if the word has one of the prefixes "y, A, n, t"-"ي ,ا ,ن ,ت", it seems to be a present tense verb.
    iii. Preceding word: for example, the word after "ln"-"لن" (not) must be a verb in present tense. Similarly, an imperative verb after "qd"-"قد" (may be) is not possible.

b. Induction of noun subclasses: proper, common, reduced and adjective (including genealogical)[33] nouns is achieved as follows:

i. By the pattern: for example reduced nouns can be identified by their pattern because there are exactly three patterns for reduced nouns.

ii. By the word structure: for example the genealogical can be induced by the word ending, because it always ends with "y"-"ي".

iii. By the affixes: for example if a word ends with "p"-"ة", it seems not to be a proper noun.

iv. By the context: for example, if the previous word is a verb then the current one seems to be a common or proper noun.

At this stage we do not resolve the ambiguities; instead we find the most important analyzing for the word. We may overlook some possibilities, but they are very infrequent.

5.6.1.2 Extracting lemma and features

After differentiation between classes of words now we do the second phase of extracting lemma and features. Verbs and noun subclasses will be processed separately, but by the same methodology:

6. 1. Extracting the features from the affixes.

7. 2. Extracting the lemma by:

a. Deleting the affixes.
b. Retrieving the deleted and (or) the changed letter which resulted from the inflection.

For verbs classes, the above steps will be:

1. From affixes: for example the verb ending with "yn"-"ين" seems to be (i) plural masculine or (ii) singular feminine or (iii) dual masculine or (iv) dual feminine. If a verb begins with "t"-"ت" it seems to be (i) masculine 2nd person or (ii) feminine 2nd person. If we combine these two rules on the verb "tqwlyn"-"تقولين" (you (feminine) say), we simply induce its features to be singular feminine 2nd person.

2.

[33] We take only these classes of nouns because other noun subclasses are closed.

a. Deleting the affixes. “tqwlyn”-“تقولين” will give “qwly”-“قول”

b.“قول” “qwly” will become “قال”-“qAl” (he say). Let us note that this affects only the vowels “y, A, w”“ي , ا , و ”.

An example for nouns:

1. By affixes: for example a word ending with “p”-“ة” seems to be singular feminine.

2.

a. Deleting affixes (with exceptions). For example the word “ftAp”-“فتاة” (girl) will become “ftA”-“فتا”. The word “jrvwmy”-“جرثومي” (bacterial) is a genealogical noun[34] and, by exception, the affix “y”-“ي” will not be deleted.
b. Extracting the lemma by retrieving the deleted or changed letters (if necessary). “ftA”-“فتا” will become “ftY”-“فتى” “boy”.

We must remember that in most cases the word exists in the dictionary, which is quite large, especially for verbs, and the above heuristic analysis is done only for words which are not in the database.

5.7 Building Dictionary

Now we describe the construction of dictionaries, which are used in preprocessing. These dictionaries play a similar role to the dictionaries used in Buckwalter analyzer, with lemma added to POS and Features.

For verbs: This dictionary consists of slightly more than 6000 verbs inflected in all possible forms according to the templates used by Al-Dahdah [37] with adding certainty and jussive case. Then all these inflections are sorted and encoded in a way such that we can find them efficiently. The input to dictionary is an inflected verb in any tense or case and the output are its lemma and features. We used this large dictionary for one reason which is to get rid the problem of the changing which may happen in the inflected verb. The second reason is that verbs seem to be an almost closed set, and using about 6000 inflected verbs gives us more information than a corpus having 10 Mega words. The reason is that each verb has approximately 164 inflections. It means that we have approximately 984000 inflections, many of which will be missing in a corpus of size 10 Mega words.

[34] The ambiguity between “ktAfy”-“كتابي” as (my book) or (genealogical noun) was solved by tokenization preprocessing.

At present the software does binary search in a full dictionary of about 984000 inflections and is reasonably fast.

However, it is possible to encode the dictionary in a smaller and slightly more effective data structure, which has separate dictionaries of prefixes, suffixes and pairs (stem*, lemma), similarly as in Buckwalter analyzer. Stem* is created exactly as a stem, but is some cases can be an illegal word, and therefore not a stem in the strict sense. Our task is to induce the possible stems* from the inflected form of the verb. For example when the verb "qAl"-"قال" (he say) is inflected, we get as possible stems*: "قال" "qAl", "قول" "qwl", "قيل" "qyl" and "قل" "ql", all of which point to the same lemma, which is the output. Indeed, only the first one of the above words is legal and is therefore a true lemma. In other words, the stored stems* of inflected verbs are the forms which appear at least once in an inflection of a verb.

In case of particles we have a list of all particles, each one with its working and meaning, and therefore the analyzing process is again a simple search problem, like in the in case of verbs.

In the case of nouns, adjectives and so on, we collected them from the Internet. We added inflections and derivations as feminine (if applicable), numbers, genealogically (Yaa Alnasabi) and reduced nouns. The object, subject nouns, broken plural and so on are not derived by this method; instead they are collected from texts which reduces the cost of the code (time of writing code) and applying this generation on them if applicable. There are many classes of nouns which are closed sets, for example question nouns, numeral nouns and so on. The resulting dictionary is updatable.

5.8 Results

The proposed analyzer was built as a preprocessing stage of an Arabic tagging system. It is therefore not a general purpose analyzer. It produces all possible analyses for a given inflected word or clitics. These analyses are POS, features and lemma. Because it is used for subsequent tagging, the evaluation of it should measure how well it satisfies its function, i.e., generates true combinations of tag (POS & features) and lemma. Therefore we will not evaluate it according to recall, precision and F-measure.

The first important thing is to have the true tag and lemma produced.

The test dataset was a small corpus of 16 k words, manually annotated by a single analysis for each word, correct for this particular use of that word. In the test, for 99.67% of words, this correct analysis was among those produced by the analyzer.

The second important thing is that the analyzer almost never produces grammatically incorrect analyses.

In a manual verification of the output of the analyzer, only 0.1% of all analyses were grammatically incorrect. Appendix C shows practical analysis for a simple sentence.

5.9 Related work

Extracting lemma was much less studied than stem in the analysis stage. Many researchers dealt with lemma in Arabic language, but they did not explain details of the procedure of extracting lemma from the word. Some other researchers did not distinguish between lemma and stem and they dealt with them as if they were the same. The other researchers dealt with the root, especially in morphological analyses. It should be noted that root induction is relatively simpler than stem and lemma.

(El-Shishtawy & El-Ghannam) [39] do lemmatization in three phases: analyzing, POS tagging and then lemma generation. The first phase implementation is done with the open source Khoja stemmer (Khoja) [56], i.e., no private analyzer. The second phase is POS tagging which depends basically on patterns. The third phase is lemma generation which is related to our approach. They depend on patterns and rules for generating the lemma from verb without any explanation or examples of these rules. The noun is manipulated in similar manner. Our approach at the first glance may appear to be similar to this work, but there are many differences: first, they take the output of POS tagging to lemma generation and in our work the output of lemmatization and analyzing stage will be fed to POS tagging. I.e., our lemma generation is done by the analyzer alone and does not depend on tagging. Second, in our work we use our own analyzer, while the authors of (El-Shishtawy & El-Ghannam) [39] use a third-party analyzer. Third, in our system at least one lemma is produced for each analysis,

while in the other system the lemma is produced only for the previously selected POS. Fourth, we use a dictionary of fully inflected forms of the known words and templates for unknown verbs. In case of nouns we use rules and a dictionary of irregular cases. El-Shishtawy & El-Ghannam [39] do not explain their approach in detail, except that they mention a dictionary of irregular forms. Fifth, their approach is limited to IR, and our approach is quite general.

Concerning morphological analyzers, there are many works in this field.

MAGEAD (Habash et al.) [50] provides an analysis for a root+pattern representation, it has separate phonological and orthographic representations, and it allows for combining morphemes from different dialects.

Darwish analyzer [31] was only concerned with generating the possible roots of a given Arabic word. It is based on automatically derived rules and statistics.

(Gridach and Chenfour) [44] Their approach is based on Arabic morphological automaton technology. They take a special representation of Arabic morphology (root and scheme) to construct a few morphological automata which were used directly in developing a system for Arabic morphological analysis and generation.

Elixir-FM (Smrz) [88] is a functional morphology system which models templatic morphology and orthographic rules.

BAMA (Buckwalter) [26] is based on a lexicon, which has morphotactic and orthographic rules encoded inside it.

5.10 Discussion and feature work

We have built, implemented and evaluated an Arabic analyzer which extracts lemma. The analyzer produces POS, features and lemma of the inflected word or clitics. The produced POS and features are described according to our new, very rich tagset. Many problems, which can be solved by a tagging system, were solved by the analyzer using the context. The context is taken in account only for unknown words. According to the previous results, it is suitable to use it in tagging. Lemma extraction offers many benefits when compared to extracting stem or root. For example, it can be used in word sense disambiguation.

Our suggestion is that expanding (i) the number of the inflected verbs used in the analyzer and (ii) expanding the database of abnormal inflections of the noun subclasses, can lead to still more accurate analyses.

It would be very beneficial to test the analyzer on a larger corpus. However, it is very time-consuming to produce, since it must be done by hand using a new, rich tagset.

Chapter 6

Survey of General and Arabic Tagging systems

6.1 Introduction

POS tagging is one of the most important natural language problems studied by researchers. The significance of POS for language processing is the large amount of information they give about a word and its neighbors (Jurafsky & Martin) [54]. POS tagging is the process of assigning a part-of-speech or other syntactic class marker to each word in the corpus (Jurafsky & Martin) [54]. It is, in other words, the process of assigning a tag from limited set of tags (tagset) to a word. The number of tags in a tagset depends on the language and the intended application. If we talk about tagging then we always mean some tagset, perhaps implicitly. See Chapter 2 for more details about tagset.

There are many methods applied to POS tagging. Most of the modern methods use some form of machine learning.

This chapter will focus on methods used for tagging regardless of the language. Then we will list the most important approaches applied to Arabic language.

There are many classifications of POS tagging methods, like the distinction between supervised and unsupervised methods, or into rule-based, stochastic and hybrid. We do not use these classifications in our presentation below.

6.2 Tagging by manually created rules

It is the oldest morphosyntactic disambiguation method, claimed to be the best, but very costly. It requires manual work of experts. Modern and earliest rule-based approaches to POS tagging are based on two stages architecture (Jurafsky & Martin) [54]. They are dictionary and rule sets. The dictionary is used to assign each word a list of possible POS tags. The rule-sets (mostly manually written) are used for solving the tagging problem, i.e., choosing the right POS for each word. In some cases, these rules can even correctly tag unknown words.

A rule-based tagger tries to apply some linguistic knowledge to exclude sequences of tags that are syntactically incorrect. They can be of the form of contextual rules such as: *if an unknown term is preceded by a determiner and followed by a noun, then label it as an adjective* (Jackson & Moulinier) [53]. The main drawback of those early systems are the laborious work of manually coding the rules and the requirement of linguistic background (Nitin & Fred) [73]. Probably the first rule-based tagging system was given by Klein and Simpson [61], which was based on a large set of handcrafted rules and a small lexicon to handle the exceptions (Nitin & Fred) [73].

Constraint grammar approach is another example of this method and EngCG is a tagger based on this approach. It applies a large set of constraints (as many as 3,744 constraints) to the input sentence to rule out incorrect POS tags (Karlsson et al.) [55].

6.3 *n*-grams Model

n-grams are crucial in many NLP tasks, tagging is one of these tasks. *n*-gram is a contiguous sequence of *n* items from a given text.

Initially *n*-grams were used for predicting the next word in a text, and the chain rule of probability of words (in a sentence of length *n*) was used for that purpose:

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)...P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

Here $P(w_3 \mid w_1^2)$ is the probability of 3rd word given the sequence of 1st word and 2nd word, etc.

The *n*-gram approximation of the above is:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$$

The probabilities are taken from counted frequencies in the training corpus:

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

n-grams model is sometimes referred to as the Language Model. The previous formula is good for predicting words but how is it used for tagging. If we want to apply it to tagging, we may do the following (in the general case):

$$\hat{t}_i = \arg\max_{t_i} \quad p(w_i|t_i) P(t_i|t_{i-N+1}^{i-1})$$

There are special cases of *n*-grams which are unigram, bigram and trigram, where *n* is 1, 2 and 3, respectively. Unigram is very simple, does not take any context information into account (no tag sequence information). Unigram simply selects the most probable tag for each specific word. Bigram uses more (but still little) information by taking the previous tag into account. Trigram adds even more by taking two previous tags into account.

The following formulas represent unigram, bigram and trigram tagging respectively:

Unigram simplification $\hat{t}_i = \arg\max_{t_j} \quad p(w_i \mid t_i)$

Bigram simplification $\hat{t}_i = \arg\max_{t_j} \quad p(w_i \mid t_i)\, p(t_i \mid t_{i-1})$

Trigram simplification $\hat{t}_i = \arg\max_{t_j} \quad p(w_i \mid t_i)\, p(t_i \mid t_{i-2} t_{i-1})$

n-grams simplification is very important in HMM tagger. Some problems arise by using *n*-grams and many stochastic tagging methods, when some *n*-grams have frequency zero in the training corpus. They can be solved by using Laplace or Good-Turing smoothing (Jurafsky & Martin) [54]. When we use *n*-grams and we have no example of a particular *n*-gram we can use shorter sequences. We can do

also weighted interpolation of trigram, bigram and unigram count (Jurafsky & Martin) [54].

6.4 Transformation-Based tagging (Brill) [24]

It is an approach based on machine learning, and is sometimes called Brill tagging. Instead of trying to acquire the linguistic rules manually, Brill describes a system that learns a set of correction rules by a methodology called transformation-based learning (TBL) (Nitin & Fred) [73]. It behaves like a method with manually written rules, because rules are used to specify tags, and at the same time like a stochastic tagging, because machine learning is used, based on a manually tagged corpus.

The algorithm has two main phases (Nitin & Fred) [73]:

1. Initial phase.
2. Learning phase.

Initial phase is accomplished by labeling every word with its most likely tag, for example, by assuming that each word is a noun (which is the most common tag) or taking the output of another tagger.

Learning phase is accomplished by two stages repeated in a loop until there is no improvement any more. The first is the examination of every possible transformation and selecting one which gives the maximal improvement of the tagging. The second is re-tagging corpus applying the rules from the first stage. The rules are limited to predefined templates. See Figure 6-1 for an example of these templates where a, b, z, and w are POS tags.

Change tag a to b when the preceding (following) word is tagged z.
Change tag a to b when the word two before (after) is tagged z.
Change tag a to b when one of the two preceding (following) words is tagged Z.
Change tag a to b when one of the three preceding (following) words is tagged z.
Change tag a to b when the preceding word is tagged z and the following word is tagged w.
Change tag a to b when the preceding (following) word is tagged z and the word two before (after) is tagged w.
Change tag a to b when the current word is (is not) capitalized.
Change tag a to b when the previous word is (is not) capitalized.

Figure 6-1: **Examples of Brill Templates.**

These templates are used for inducing rules in the same form with different data. An example of a rule learned by Brill's tagger is "*Change tag NN to VB when the previous word is tagged TO*".

The space of transformation sequences we have to search is huge. A naive implementation of transformation-based learning will therefore be quite inefficient (Manning & Schütze) [68].

6.5 HMM tagger

Hidden Markov Model (HMM) is the most frequently used technique for POS tagging. It is used for tagging one complete sentence at a time, by selecting the most likely sequence of tags for its words.

HMMs allow us to estimate probabilities of unobserved events where observed events are the words and the hidden events are part-of-speech tags. It uses the formula:

$$t_1^n = \arg\max_{t_1^n} \quad p(t_1^n \mid w_1^n)$$

We cannot compute it directly, therefore by using Bayes' rule with simplification the previous formula will be:

$$t_1^n = \arg\max_{t_1^n} \quad p(w_1^n \mid t_1^n)\, p(t_1^n)$$

HMM tagger simplifies this formula by two assumptions. The first assumption is that the probability of a word depends on its part-of-speech tag and is independent of other words around it, and of the other tags around it:

$$p(w_1^n \mid t_1^n) \approx \prod_{i=1}^{n} p(w_i \mid t_i)$$

The second assumption is that the probability of a tag appearing depends only on the previous tag, the bigram assumption (Jurafsky & Martin) [54]:

$$p(t_1^n) \approx \prod_{i=1}^{n} p(t_i \mid t_{i-1})$$

Together they yield the third equation:

$$t_1^n = \arg\max_{t_1^n} \quad p(t_1^n \mid w_1^n) \approx \arg\max_{t_1^n} \prod_{i=1}^{n} p(w_i \mid t_i)\, p(t_i \mid t_{i-1})$$

6.6 Decision trees [83]

DT tagger was presented in (Schmid) [83], as an improvement to HMM method, avoiding problems of estimating transition probabilities from sparse data. In this tagger transition probabilities are estimated using decision tree. The decision tree automatically determines the appropriate size of the context which is used to estimate the transition probabilities. The most important criterion for the success of the learning algorithms based on DTs is the construction of a set of questions to be used in the decision procedure (Nitin & Fred) [73]. DT tagger is a Markov model using DT for estimating transition probabilities ($p(t_n \mid t_{n-2}t_{n-1})$).

6.7 Maximum Entropy

It was proposed by (Ratnaparkhi) [77][78]. Maximum entropy (ME) models provide us more flexibility in dealing with the context and are used as an alternative to HMMs in the domain of POS tagging (Nitin & Fred) [73]. The flexibility comes from the ability to include any template that we consider useful: it may be simple (target tag t_i depends on t_{i-1}) or complex (t_i depends on t_{i-1} and/or t_{i-2} and/or w_{i+1}) (Nitin & Fred) [73]:

$$t_1^n = \arg\max_{t_1^n} \quad p(t_1^n \mid w_1^n)$$

$$\approx \arg\max_{t_1^n} \prod_{i=1}^{n} p(t_i \mid h_i)$$

We can express the conditional probability in terms of a log-linear (exponential) model (Nitin & Fred) [73]:

$$p(t \mid h) = \frac{1}{Z(h)} \quad \prod_{j=1}^{k} \alpha_j^{\,f_j(t,h)}$$

$$Z(h) = \sum_t \prod_{j=1}^{k} \alpha_j^{\,f_j(t,h)}$$

$Z(h)$ is to ensure true probability distribution and f_j is a feature with binary value (see Figure 6-2 and Figure 6-3 for a whole template of features and an

example, respectively) and α_j is the weight of f_j with positive value. t is a tag from a tagset T and h is a history from the possible contexts (histories) H.

Condition	Features	
w_i is not rare	$w_i = X$	& $t_i = T$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$	& $t_i = T$
	X is suffix of w_i, $\|X\| \leq 4$	& $t_i = T$
	w_i contains number	& $t_i = T$
	w_i contains uppercase character	& $t_i = T$
	w_i contains hyphen	& $t_i = T$
$\forall\ w_i$	$t_{i-1} = X$	& $t_i = T$
	$t_{i-2}t_{i-1} = XY$	& $t_i = T$
	$w_{i-1} = X$	& $t_i = T$
	$w_{i-2} = X$	& $t_i = T$
	$w_{i+1} = X$	& $t_i = T$
	$w_{i+2} = X$	& $t_i = T$

Figure 6-2: template in (Ratnaparkhi) [77].

$$f_j(h_i,t_i) = \begin{cases} 1 & \text{if } w_i = \textit{like and } t_i = VB \\ 0 & \text{otherwise} \end{cases}$$

$$f_j(h_i,t_i) = \begin{cases} 1 & \text{if } \textit{suffix}(w_i) = \text{"ing" and } t_i = VBG \\ 0 & \text{otherwise} \end{cases}$$

$$f_j(h_i,t_i) = \begin{cases} 1 & \text{if } w_i = \textit{about and } t_{i-2} = DET \textit{ and } t_{i-1} = NNS \textit{ and } t_i = IN \\ 0 & \text{otherwise} \end{cases}$$

Figure 6-3: Practical features in ME approach. In a maximum entropy model, the feature can be simple: this word has this tag, consider morphology or consider tag sequences.

The probability distribution P we seek is the one that maximizes the entropy of the distribution under some constraints:

$$\arg\max_p - \left(\sum_{\substack{h \in H \\ t \in T}} \overline{P}(h) P(t \mid h) \log(t \mid h) \right)$$

subject to

$$E(f_j) = \overline{E}(f_j) \qquad 1 \leq j \leq k$$

$$\sum_{i=1}^{n} \overline{P}(h_i)P(t_i \mid h_i)\, f_j(h_i,t_i) = \sum_{i=1}^{n} \overline{P}(h_i,t_i)\, f_j(h_i,t_i)$$

$E(f_j)$ and $\overline{E}(f_j)$ denote, respectively, the model's expectation and the observed expectation of feature f_j . $\overline{P}(h_i)$ and $\overline{P}(h_i,t_i)$ are the relative frequencies, respectively, of context h_i and the context-tag pair (h_i , t_i) in the training data. The intuition behind maximizing the entropy is that it gives us the most uncertain distribution. In other words, we do not include any information in the distribution that is not justified by the empirical evidence available to us. The parameters of the distribution P can be obtained using the generalized iterative scaling algorithm (Nitin & Fred) [73].

6.8 Neural networks

Neural network is information processing paradigm inspired by biological nervous systems, such as our brain. Structurally, it is a large number of highly interconnected processing elements (neurons) working together. Like people, they learn from experience (by example). Neural networks are configured for a specific application, such as pattern recognition or data classification, through a learning process.

In multilayer perceptron networks (MLP-networks), the processing units are arranged vertically in several layers. Connections exist only between units in adjacent layers. There are three classes of layers which are input layer, hidden layer (activations are not visible externally) and output layer. The goal is to find the best network to predict, based on the input nodes, the correct output nodes.

(Schmid) [84] introduced neural networks for POS tagging. The Net-Tagger consists of an MLP-network and a lexicon. In the output layer of the MLP network each unit corresponds to one of the tags in the tagset. The network learns during training to activate that output unit that represents the correct tag and to deactivate all other output units. Hence, in the trained network, the output unit with the highest activation indicates, which tag should be attached to the word that is currently being processed.

The input of the network comprises all the information that the system has about the POS's of the current word, the p preceding words and the f following words. More specifically, for each POS tag t_j and each of the $p + 1 + f$ words in the context, there is an input unit whose activation in_{ij} represents the probability that w_i has part of speech t_i. So, if there are n possible tags, there are $n * (p + 1 + f)$ input nodes.

For the input word being tagged and its following words, the lexical POS probability $p(t_j|w_i)$ is all we know about the POS. This probability does not take any contextual influences into account. For the preceding words, there is more information available, because they have already been tagged.

An artificial neural network gave 96.22% accuracy for English (Schmid) [84].

Although Neural Network (NN) taggers do not seem to outperform the HMM taggers in general, they have some attractive properties. First, ambiguous tagging can be handled easily without additional computation. When the output nodes of a network correspond to the tags in the tagset, normally, given an input word and its context during the tagging phase, the output node with the highest activation is selected as the tag of the word. However, if there are several output nodes with close enough activation values, all of them can be given as candidate tags (Nitin & Fred) [73].

Neural network taggers converge to top performances with small amounts of training data and they are suitable for languages for which large corpora are not available (Nitin & Fred) [73].

6.9 Memory based learning [30]

Memory-based learning is a form of supervised learning based on similarity-based reasoning. The part of speech tag of a word in a particular context is extrapolated from the most similar cases held in memory (Daelemans et, al.) [30].

In AI, the concept has appeared in several disciplines (from computer vision to robotics), using terminology such as similarity-based, example-based, memory-based, exemplar-based, case-based, analogical, lazy, nearest-neighbor, and instance-based (Daelemans et, al.) [30].

In a memory-based approach, a set of cases is kept in memory. Each case consists of a word with preceding and following context, and the corresponding category for that word in that context. A new sentence is tagged by selecting for each word in the sentence and its context the most similar case(s) in memory, and extrapolating the category of the word from these 'nearest neighbors'. A memory-based approach has features of both learning rule-based taggers (each case as a specific rule) and of stochastic taggers (form of *k*-nearest neighbors modeling). The approach in its basic form is computationally expensive, however; each new word in context that has to be tagged, has to be compared to each pattern kept in memory (Daelemans et, al.) [30].

Memory-based learning is a form of supervised, inductive learning from examples. Examples are represented as vectors of feature values with an associated category label (Daelemans et, al.) [30].

6.10 Boosting [1]

Boosting is a machine learning algorithm that was introduced to POS tagging by (Abney et al.) [1].

The idea of boosting is to combine many simple "rules of thumb" called "weak hypotheses", such as "*the current word is a noun if the previous word is* ***the***". The main idea of boosting is to combine many such rules in a principled manner to produce a single highly accurate classification rule (Abney et, al.) [1].

The boosting algorithm is an iterative one of R rounds, where a new rule of thumb is derived from the training data at each round, using a weak learner (Jackson & Moulinier) [53].

Boosting is similar to transformation-based learning (Brill), both build classifiers by combining simple rules, and both are noted for their resistance to overfitting, but they differ in theoretical foundation (Abney et al.) [1].

6.11 Relaxation labeling (Padró) [75]

Relaxation is a well-known technique used to solve consistent labeling problems.

A consistent labeling problem, given a set of variables, is to assign to each variable a label compatible with the labels of the other ones, according to a set of compatibility constraints.

The main idea of using relaxation labeling in POS tagging is to represent POS tagging as a constraint satisfaction problem. Then it can be addressed with the usual techniques of that field, such as relaxation labeling.

It seems reasonable to consider POS tagging as a combinatorial problem, in which we have a set of variables (words in a sentence), a set of possible labels for each one (POS tags), and a set of constraints.

One can consider weighted labeling, in which a weight is assigned to each possible label of each variable, and the task is to maximize the "global consistency" by relaxation. The constraints can be gathered automatically from the training corpus, too.

6.12 Cyclic Dependency Network [90]

The conditional probability of tag dependency is assumed unidirectional (depending on previous tags) in *n*-gram based methods, including HMM tagging. In CDN this conditional probability of tag dependency is bidirectional (depending on previous and following tags). (Toutanova et, al.) [90] proposes to make an explicit use of both preceding and following tag contexts via a dependency network representation, using priors in conditional loglinear models. The resulting tagger gives 97.24% accuracy on the Penn Treebank WSJ.

6.13 Finite-State Transducers [79]

Finite-state transducers have important applications in many areas of natural language processing.

A finite-state transducer is a finite-state automaton whose transitions are labeled by pairs of symbols. The first symbol is the input and the second is the output. Applying a finite-state transducer to an input consists of following a path according to the input symbols, and the result is the sequence of output symbols encountered on that path.

(Roche & Schabes) [79] used FST for speeding up processing the Brill tagger. It is constructed in four steps.

The first step consists of turning each contextual rule found in Brill's tagger into a finite-state transducer. Each contextual rule is defined locally; that is, the transformation it describes must be applied at each position of the input sequence.

The second step consists of turning the transducers produced by the preceding step into transducers that operate globally on the input in one pass. This transformation is performed for each transducer associated with each rule.

The third step combines all transducers into a single transducer.

The fourth and final step consists of transforming the finite-state transducer obtained in the previous step into an equivalent deterministic transducer.

(Silfverberg & Lindén) [86] used parallel weighted finite-state transducers to implement a part-of-speech tagger. Their system consists of a weighted lexicon and a guesser combined with a bigram model turned into two weighted transducers. They reported 98.29% of accuracy on English Europarl corpus.

6.14 Genetic Algorithm [2]

Genetic algorithms are a group of very general algorithms to find approximate solutions to optimization and search problems.

(Nitin & Fred) [73] Although genetic algorithms have accuracies worse than those offered by HMM and rule-based approaches, they can be seen as an efficient alternative in POS tagging. They reach performances near their top performances with small populations and a few iterations.

(Alba et, al.) [2] report a genetic algorithm able to solve the tagging problem with accuracy no worse than a specific method which was designed for this

problem. In addition, GAs can perform the search of the best sequence of tags for any context-based model, even if it does not fulfill the Markov assumption.

6.15 SVM

Support Vector Machines (SVMs) are supervised machine learning algorithms for binary classification (Nakagawa et, al.) [72]. They can handle a large number of (overlapping) features with good generalization performance (Diab et, al.) [33]. SVMs can easily handle high-dimensional spaces, with a large number of features (Nitin & Fred) [73] (Mayfield et, al.) [70].

SVMs are known to be resistant to overtraining, because only the training vectors that are closest to the hyperplane (called support vectors) determine itse parameters (Nitin & Fred) [73] (Mayfield et, al.) [70].

(Mayfield et, al.) [70] report tagging accuracy 92.95 %. The data set was the Penn Treebank Wall Street Journal collection, which contains about 1.5 million tokens annotated with a part of speech for each token.

6.16 Fuzzy set theory [60]

The taggers formed using the fuzzy set theory are similar to HMM taggers, except that probabilities used in the latter are replaced by fuzzy membership functions in the former (Nitin & Fred) [73]. Neural networks are used for estimating the transition probabilities and some transformations of lexical probabilities for the observation possibilities (Kim & Kim) [60]. One advantage of these taggers is their high performance with small data sizes (Nitin & Fred) [73].

(Kim & Kim) [60] showed, using fuzzy set theory of second order, the accuracy around 95.81 % on 800,000 words from the Brown corpus which is included in the Penn Treebank Corpus; the tagset size was 49 tags.

6.17 Best match

(Stomp) [87] “matches the text to be tagged to long continuous strings from the training data (as long as possible) and assigns each match the same tags as the matching part of the training data”. Back-off, as smoothing, is used with this method. The accuracy achieved by this method is 94.5 %. The Stockholm-Umea

Corpus (SUC) a manually corrected tagged corpus of Swedish, was used for training and testing. This method is described in more detail in Chapter 7 below.

6.18 Combining different taggers

Most of the methods and papers quoted above used only one method for tagging. However, there are methods to combine them in a way such that the accuracy will be improved. Combined taggers can be classified into:

Voting (Henrich et, al) [52]: a few taggers are run independently and the final result is selected by voting among these taggers.

Stacking (Wu et, al.) [91]: the output of one tagger is fed to another one in a serial sequence.

co-training (Clark) [29]: two taggers are trained on the output of the other one.

Fusion: taggers are combined internally.

Hybrid: combination of two or more of the previous methods.

For more details see Chapter 7, where they are explained in more detail. We describe a new method, called **master-slaves**, to combine taggers.

6.19 POS tagging approaches used for Arabic

SVM: (Diab) (Diab et, al.) [33][32] applied SVM to Arabic POS tagging and tokenization. The SVM-POS tagger achieved accuracy of 95.49%. The Arabic TreeBank consisting of 4519 sentences was used in training and testing. She used the LDC's POS tagset, which consists of 24 tags[35].

SVM + morphological analyzer: (Habash and Rambow) [46] applied SVM with support of a morphological analyzer for producing all possible analyses. The data used came from the Penn Arabic Treebank (Maamouri et al.). Their POS evaluation shows accuracy of 97.6% on ATB1 and accuracy of 95.7% on ATB2, both based on gold standard tokenization.

Statistical and rule-based: In (Khoja) [57], a system is developed, using a combination of both statistical and rule-based techniques. It uses a simple tagset. A corpus of 50,000 words in Modern Standard Arabic (an extract from the Saudi

[35] See chapter 3 for more details

Al-Jazirah newspaper, dated 03/03/1999) was tagged using this tagset[36]. She achieved accuracy of around 90 %.

HMM: (AL-Shamsi & Guessoum) [11] The proposed HMM POS tagger has been tested and has achieved performance of 97%. It used a very simple POS tagset of 55 tags. The training was done on a special small corpus consisting of 9.15 MB corpus of native Arabic articles. The authors used a stemmer for segmenting and separating affixes from the stem to produce prefix, stem, and suffix parts.

Brill (Transformation) tagging: first, Freeman [42] presented an Arabic tagger based on the Brill tagger. He was using this environment as a tool to semi-automatically tag text. With every new text he added rules to the tagger's rule files by hand, as well as new items to the tagger's lexicon file.

Brill (Transformation) + morphological analyzer: (AlGahtani et al.) [5] used transformation-based learning as implemented in the Brill tagger (Brill, 1994) for POS tagging of Arabic, with segment-based tags. They used the Buckwalter morphological analyzer (Buckwalter) [25]. (AlGahtani et al.) evaluated their approach on the whole ATB as well as on ATB1. For ATB1, they achieved POS tagging accuracy of 96.9%. Using the whole ATB the accuracy was 96.1%, even though large parts of the treebank are duplicated between parts, so that it is likely that parts of their test set were actually present in the training set (AlGahtani et al.) [5].

Rules-based and memory-based: (Tlili-Guiassa) [89] used a hybrid of rule-based and a memory-based learning methods. His method is based firstly on rules automatically learned from the training corpus (that consider the post-position, ending of a word and patterns) and then the anomalies were corrected by adopting a memory-based learning method (MBL). Secondly, by checking the exceptional cases of rules, more information was made available to the learner for treating those exceptional cases. The accuracy was 85 %. The tagset was derived from that of Khoja.

Classifier + regular expressions: (Seth Kulick**)** [64] described an approach to simultaneous tokenization and part-of-speech tagging that is based on separating

[36] See chapter 3 for more details

the closed- and open-class items, and focusing on the likelihood of the possible stems of the open class words. He used regular expressions with a reduced tagset. The data set was ATB3-v3.2 and the accuracy of tagging was 95.147%.

Memory-based learning: (Van den Bosch et al.) [23] used memory-based learning for both morphological analysis and POS tagging of Arabic. They reported an overall accuracy of 91.5%.

Statistical [71]: (Mohamed & Kübler) [71] used two approaches. Their first approach used complex tags that described full words and did not require any word segmentation. The second approach was segmentation-based, using a segmenter based on machine learning. They showed that word-based POS tagging can yield better results than segment-based tagging (93.93% vs. 93.41%). Combining both methods resulted in a word accuracy of 94.37%. POS tagset of the Penn Arabic Treebank was used and two sections of the ATB (P1V3 and P3V1), since those two sets do not contain duplicate sentences. This data set contained approximately 500 000 words.

HMM tagger without morphological analyzer or lexicon: In (Köprü) [62] the accuracy was 95.51% with a very small Arabic tagset of 17 tags. The data set was Penn Arabic Treebank ATB (parts 1, 2 and 3) which consisted of 629,866 words.

HMM tagger with morphological analyzer: In (El Hadj et, al.) [36] the data set was 21882 words with a very small, custom tagset of 13 tags. The accuracy was 96%.

HMM tagger with morphological analyzer with lexicon: In (Mansour) [69] the morphological analyzer was Buckwalter's analyzer. This approach was applied to Hebrew and Arabic. The data set was ATB (parts 1, 2 and 3). The accuracy was 96.12%.

6.20 Arabic POS tagging as a part of toolkits and applications

There are many toolkits for specific tasks in Arabic language processing. The best known ones which do POS tagging are MADA+TOKEN and AMIRA.

MADA (Morphological Analysis and Disambiguation for Arabic) (Habash) [45] is a utility that, given raw Arabic text, adds as much lexical and morphological information as possible by disambiguating, in one operation, part-of-speech tags, lexemes, diacritizations and full morphological analyses (Habash) [45]. TOKEN is a general tokenizer for Arabic.

AMIRA [32] is a successor suite of the ASVMTools (Diab et al., 2007). The AMIRA toolkit includes a clitic tokenizer (TOK), part of speech tagger (POS) and base phrase chunker (BPC) – a shallow syntactic parser. The accuracy of Amira using ERTS tagset was reported to be 96.13% and the accuracy using RTS tagset to be 96.15%.

Chapter 7

Combining Taggers in Master-slaves technique

7.1 Introduction

There are many methods used for POS tagging. Most of modern methods are corpus-based and are based on machine learning. HMM is the most studied and probably the most frequently used tagging method. We propose a new method to combine taggers, which we call master-slave technique. In our approach, HMM tagger is used as the master tagger and Brill and MaxMatch (MM) taggers as

slaves. The main property of our method is that the master tagger will process each sentence with different probabilities (different knowledge), as we will see in next sections.

7.2 Related work

There are many approaches and works in POS tagging therefore we will mention only those used in our approach and some of the combined approaches.

Stomp [87] in his MaxMatch tagger "matches the text to be tagged to long continuous strings from the training data (as long as possible) and assigns each match the same tags as the matching part of the training data". The same idea but in a different context is used as a part of our research. In the paper (Glass & Bangay) [43] first the performance of each used tagger is verified experimentally. The taggers are then grouped to form a voting system, but in no cases the combined results improve on the individual accuracies. In (Yonghui et, al.) [92] the authors, after studying four corpus-based approaches to part of speech (POS) tagging: tranform-based error driven, the decision tree, hidden Markov model and maximum entropy, present a novel data fusion strategy in POS tagging – called correlation voting. They proved that the correlative voting is better than other fusion methods, with an average decrease of 27.85% of the initial tagging error rate. In the paper (Henrich et, al.) [52] combiTagger combines automatically the outputs of several taggers. The system, which is open source, provides algorithms for simple and weighted voting. It improved the accuracy by 1.26 – 1.58 % over the best method of its individual component taggers. The authors of (Loftsson) [67] used many combinations of several taggers in a simple voting approach. The combination of TBL, TNT and Ice taggers wins 0.81% over the best individual method which was Ice tagger (with accuracy 91.80%).

The book (Nitin & Fred) [73] presents many other combinations of taggers by using voting or stacking methods. It can be useful for further reading about combined taggers.

7.3 Techniques for combining taggers

Most of modern taggers, for annotation, are constructed by combining two or more approaches in a way such that the accuracy will be increased. Tagger

combination methods can be divided into voting, stacking, co-training, fusion and hybrid.

In **Voting**, several taggers run independently and the final result is selected by voting among these tagger outputs. Voting can be simple or weighted. In a simple voting all taggers have the same weight. Weighted voting is done by adding more weight to the tagger which has higher accuracy (Henrich et, al.) [52]. The biggest problem in voting is when the used taggers are similar in methodology, i.e., they make similar errors in similar situations.

Stacking: The basic concept behind stacking is to train two or more taggers sequentially, with each successive tagger incorporating the results of the previous ones in some fashion (Wu et, al.) [91]. The biggest problem in stacking is that the errors made by the taggers tend to accumulate.

Co-training (Clark) [29] is a method in which two taggers are iteratively retrained on each other's output. The taggers should be sufficiently different (e.g., based on different models) for co-training to be effective (Nitin & Fred) [73].

Fusion tagger is a tagger which combines several tagging approaches internally. The final tagger will somehow collect the features of its components. It is really not a method to combine arbitrary taggers, because there is no uniform way to do it and each such fusion is essentially unique. The tagger in Section 7-7 is an example of this type.

Hybrid: where several of the previous combinations are used collectively. For example voting and stacking can be used when we use a rule-based tagger for eliminating unwanted analyses and the output is fed to other many taggers for voting.

In this chapter we present a new master-slave technique using HMM tagger as a master, and Brill and MM taggers as slaves.

7.4 Maximum match (MM) Tagger

Maximum match (best match in (Sjobergh) [87]) tagger finds the longest *n*-gram (i.e., with maximal possible *n*) in the text to be tagged, which is also present in the training data, and tags the *n*-gram in the text copying the tags from its counterpart in the training data. This pair of identical *n*-grams is called a match. If

there are several equally long matching *n*-grams, the most common matching tag (in these matches) is chosen. If it is still a tie, the one first encountered is chosen. There is also a back-off method for short matches and special treatment of unknown words.

In our work we deal with maximum match in two different contexts. In the first we take it as independent tagger, implemented using a very simple version of best match (Stomp tagger). The back-off method, in this case, was not used and the unknown words get the "None" tag. It is explained in Section 7.7 how MM can be combined with the HMM tagger.

In the second context for any word *w* in the input sentence, we record the length of the longest match, which contains *w*. This length is called the maximum match for *w*. For example, if we have sentences "w_6, w_5, w_1, w_2" and "w_2, w_3, w_4, w_6" in the corpus and the input sentence is "w_1, w_2, w_3, w_4", then the maximum match is 2, 3, 3 and 3 for the words w_1, w_2, w_3 and w_4, respectively.

7.5 HMM tagger

HMM is the most frequently used technique for POS tagging. It used for tagging one complete sentence at a time by selecting the most likely sequence of tags for specific sequence of words. See Chapter 6 for more details.

7.6 First experiment of combining of MM & HMM taggers

Before going further, let us consider what happens if the input sentence completely matches a sentence from the training corpus: what is the probability of tagging the input sentence same as the one from the training corpus? The answer, theoretically, should be one, but practically there is no guarantee for this. The same problem arises when a long phrase in the input sentence is also found in the training corpus (we call it again a match). In order to increase the chance of tagging this phrase in the same way as in the training corpus, we modify the HMM tagger. We do this by using MM tagger explained in Section 7.5. The easiest way is by multiplying the HMM probabilities by a factor reflecting the number of matched tokens to the number of all tokens in the input sentence,

thereby privileging the tags which agree with the tags used for words of the match found in the training corpus.

Suppose the length of the input sentence is *n*. First we want to assign to each tag *t* in our tagset a value *mm*(*t*), resulting from processing the input sentence. It is done as follows:

mm(*t*) is the length of the longest match between the input sentence and a sentence in the training corpus, such that tag *t* is assigned to at least one word in the corpus part of the matching, minus *n.*

It is clear that for a tag *t* which never appears in a matching, *mm*(*t*)= –*n*.

Then we process the input sentence using HMM tagger whose probabilities are modified in the following way:

$$p(w_i \mid t_i)\, p(t_i \mid t_{i-1}) := p(w_i \mid t_i)\, p(t_i \mid t_{i-1}).e^{mm(t_i)}$$

What happens exactly in the previous formula is that we relatively decrease the chance of selecting the tags which do not appear in long matches.

The result of applying this change to the HMM tagger is the following equation, which defines the augmented HMM tagger.

$$t_1^n = \arg\max_{t_1^n} p(t_1^n \mid w_1^n) \approx \arg\max_{t_1^n} \prod_{i=1}^{n} p(w_i \mid t_i)\, p(t_i \mid t_{i-1}).e^{mm(t_i)}$$

A tagger using this simple idea has been implemented and tested practically, just to see if it works. The accuracy increased from 95.28%, achieved by the unmodified HMM to 95.55%, in a test using the Brown corpus of English and 10-fold cross-validation. This result has encouraged us to generalize this method, in particular to more than two taggers.

We should note that Viterbi algorithm has not been affected, because our modification is reflected by the word likelihood probabilities. And Viterbi algorithm selects the maximum input to each state (tag) depending on the transition probabilities and the word likelihood probabilities. I.e., selecting the maximum input to state still works as before.

7.7 Modification for general use

We have used MM as a source of additional information supplied to HMM, for modifying its probabilities. Indeed our formula incorporates into HMM tagger more than a single sequence of tags, because it changes the factors by which the probabilities are multiplied, depending on the length of the local maximal matching fragments. While generalizing our method to taggers other than MM, we assume that the tagger produces a single sequence of tags and nothing more. Indeed, it would be extremely difficult to incorporate with HMM anything beyond it, since the internal information produced by each tagger is different.

Therefore we modify our method of combining taggers relying on the sequences of tags produced by the taggers, only. The benefit of it is that we can use any tagger now in combination with HMM tagger.

Let's work on the same example of MM and HMM. The role of MM was to modify the HMM probabilities, i.e., each sentence was processed using HMM with different probabilities. We want to use the same idea using another tagger in place of MM, say Brill tagger. Using Brill tagger, we can modify the emission probabilities of HMM tagger, multiplying them by a constant factor *f* smaller than 1, except the tags produced by the Brill tagger on the same sentence. In general the output of a tagger is fed to HMM tagger which then re-estimates its internal probabilities (knowledge) according to previous tagger's output. The first tagger can be seen as a slave (property) and the second, which we call a master. Before explaining the details, let us note that the power of the slave depends on *f*, which can and should be selected experimentally. Our goal here is not selecting the best *f*. Definitely, it should be investigated in the future work, in particular investigating if *f* should remain a constant, or perhaps depend on the tagged sentence, the tagset used, the kind of tagger used as a slave, and many other factors. At present, we report the first experiments, using a fixed *f*.

Any number of slave taggers can be used with one master. Assume that we have m+1 taggers ($T_1 \ldots T_{m+1}$). T_{m+1} is HMM tagger and will be used as a master, the other will be used as slave taggers. The master tagger is trained for estimating its probabilities. Then the input sentence is tagged by each of the slave taggers $T_1 \ldots T_m$. The outputs of all slave taggers are fed to master in parallel for each sentence. Then the master changes its probabilities according to the outputs of the

slaves for this sentence. Then master does the tagging for this sentence according to the new probabilities. The important thing, in this method, is that using different probabilities for estimating each sentence. Figure 7-1 shows a block diagram for the proposed master and slaves tagger.

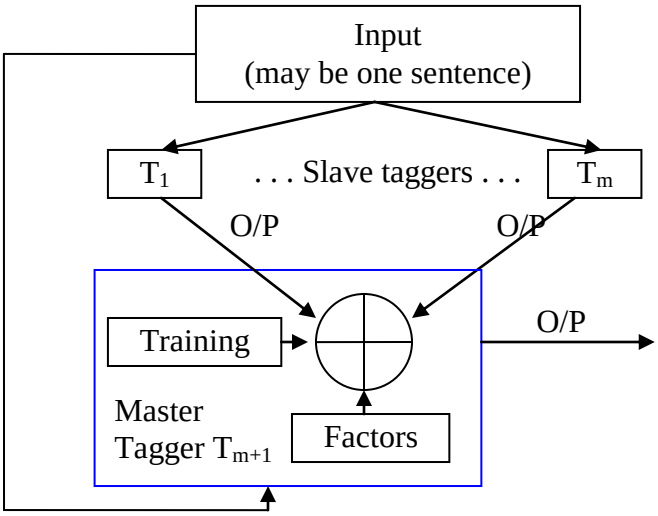


Figure 7-1: Combining taggers into a master-slaves tagger.

7.8 Difference between the new and other methods

There are many differences between our approach and the existing approaches in general. First, each sentence is tagged by the master tagger according to a different knowledge, affected by the results of the slaves taggers. We can say that we have a new tagger for each sentence.

There is no limitation for the number of slave taggers, as opposed to voting which needs an odd number of them to avoid ties.

It differs from stacking by using more than one tagger (as slaves) which feed their outputs in parallel to the master tagger.

7.9 Experiments

We have taken three taggers: HMM, MM (Stomp) and Brill tagger. Each one has been tested alone, using 10-fold cross-validation. Then we have done two tests where HMM has been the master tagger. In the first test the Brill tagger has been the only slave, and in the second we have added MM as the second slave. The factor has been constant 0.29 for all tests. It was selected in a few other tests, not reported here, as the most effective one. Our goal here was neither selecting the

best value of the factor, nor selecting the best way to use the factor. Therefore we fixed the value and the approach of using this value which was as follows: when the sentence is already tagged by the slaves then the probabilities of all tags a specific word w_n are multiplied by that factor except that the tag(s) which is/are output from the slaves for this word. The data set was Brown corpus which is freely available as a part of the NLTK package under Python environment [22]. Also Brill tagger is a built-in tagger in Python. We built very simple implementations of MM and HMM taggers. The unknown words are processed, in Brill and MM taggers, by giving them "None" tag, and in HMM by giving all tags in the tagset equal probabilities. It is not a good method in general, but we wanted to test how using of HMM, MM and Brill taggers as master and slaves changes the performance, if compared with traditional HMM tagger, under the same simple specifications. Figure 7-2 shows the results of these tests. We can see that we gain 0.26 % by using Brill tagger as the only slave and 0.42 % by using both Brill and MM as slaves. When annotating a corpus of 2 million words, it means correcting the tagging of about 8400 words. The other data set for Arabic consists of 45 files (29k words) annotated by hand with our new tagset[37].

Master or Original tagger	Slave tagger	Total words	Correctly tagged words	Accuracy	Accuracy on the Arabic corpus
Brill	----------	1161192	1096687	94.44 %	86.43%
Maxmatch	---------	1161192	1061635	91.5 %	83.26%
HMM	---------	1161192	1106482	95.28 %	88.81%
HMM	Brill	1161192	1109411	95.54 %	89.40%
HMM	Brill+maxmatch	1161192	1111281	95.70 %	90.05%

Figure 7-2: Results of Master-slaves tagging.

7.10 Discussion and Further work

In the previous section we have proposed a new method for combining taggers, the master and slaves method. We implemented this method by using three taggers which are HMM, MM and Brill tagger. We focused in our implementation on proving practically that this method works, not on selecting the best value of

[37] See chapter 3 for more details on tagsets.

the factors or selecting a different factor for each slave tagger. We would like to mention that the factors can be (i) constant for all slaves (very simple) (ii) different for each slave tagger (iii) weighted factors depending on the accuracy of each slave tagger (iv) variable factors where for each slave tagger the factor will be changed according to some conditions. Any other type of factor can be used with the same methodology where the internal probabilities of the master tagger will be changed. The gain of accuracy was quite considerable, given the simplicity of the approach and very limited tuning of the method. We hope that by using weighted or variable factors the gain of accuracy can be increased. The interesting thing in the results is that the accuracy of MM was 91.5%, much less than the first slave Brill and the master HMM, and still by adding it as the second slave we improved the accuracy. Actually we expected the accuracy to drop because of the huge difference in the accuracy between HMM and MM. But what happened is the reverse: the master tagger still has the control for selecting the best tag among the tags suggested by the slaves. It was the main reason for selecting the name of the method master and slaves. A successful application of this method to a highly a inflected language, such as Arabic, proves its generality. The low accuracies of all taggers for Arabic are mainly due to (i) using very small data set (ii) using very rich tagset.

Chapter 8

Combining Rules-based and Master-Slaves Taggers

8.1 Introduction

In this chapter we will describe an implementation of Arabic POS combined tagger. The first tagging technique we use is by using manually written rules. The tagger consists of a few hundred of hand-written rules. Most of these rules were taken from Arabic traditional grammar books (AL-Bidhani) [3] (Al-Rajhi) [10] (Al-Hamlawy) [7] (Al-Galaiini) [6]. The task of the rule-based tagger is to eliminate unwanted tags from the context. It simplifies the work of the next tagger. The second tagger is a master-slaves tagger which was constructed in the previous chapter. The master is HMM tagger and the slaves are Brill tagger and maxmatch tagger (MM). The rules-based tagger is added to the master-slaves tagger as a third, special slave. It can alternatively be seen as a separate tagger combined with master-slaves using stacking.

The main reason for adding a rules-based Arabic tagger is that we do not yet have a large corpus annotated by our rich tagset. The second reason is that we would like to annotate a new, larger Arabic corpus with our rich tagset. We do not

focus on the speed of processing because our work is intended to be a tool for producing large annotated Arabic corpus. I.e., our tagger will be used offline[38].

8.2 Related work

There are many papers that combine rules-based and statistical taggers. Almost all these works use the stacking technique. All the works mentioned in the previous Chapter can be mentioned here, e.g., (Yonghui et, al.) [92] (Henrich et, al.) [52] (Loftsson) [67]. Book (Nitin & Fred) [73] presents other combinations of taggers by using voting or stacking methods.

For Arabic, if we consider a morphological analyzer as a light tagger, (Khoja) [57] is an example of a stacking combination. All possible tags for each word with its stem are fed from the analyzer to a statistical tagger trained on a corpus, to get the best tag for that word. She achieved 90% accuracy on a data set of 50 k words, using a simple tagset[39].

Our work here is different from the above mentioned works: (i) we have an analyzer (ii) we have manually written rules for eliminating unwanted tags (iii) the output of a rules-based tagger is fed to the master-slaves tagger with two slave taggers. None of the mentioned papers had all those elements at the same time.

The earliest POS tagging systems were rule-based systems, in which a set of rules was manually constructed and then applied to tag a given text (Nitin & Fred) [73]. Theoretically such taggers should have high accuracy, but constructing such a tagger is a very difficult task. Therefore most of the researchers did not construct rule-based taggers containing rules for all possible features of the language, because it was practically impossible. Then the researchers tried to collect the rules from the experts. The main drawback of those early systems was the laborious work of manually coding the rules and the requirement of strong linguistic background.

There are also corpus-based rule taggers. The rules, in a corpus-based rule tagger, are extracted automatically from the corpus – the Brill tagger is the best example of this type.

[38] See Chapter 1 for definition of offline and online tagger.
[39] See Chapter 3 for more details on this tagset.

8.3 Comparing between manually created rule-based taggers and other taggers

In order to compare taggers meaningfully one must take into account the training data sets they use (if any), the test data sets and tagsets they use. However, one can name a few distinctions between Manually written rule-based taggers and statistical taggers, used in our work.

1. Manually written rule-based taggers do not use (and depend on) a corpus, and therefore are more general.
2. Manually written rule-based taggers are more stable in performance, when the test data changes.
3. Manually written rule-based taggers require human expertise in linguistics, which is not necessary to construct statistical taggers.
4. Manually written rule-based taggers require much more human work and are therefore slower to construct.
5. Manually written rule-based taggers have less problems with unknown words than statistical ones, especially those without analyzer.
6. There are only a few rules (no matter if manually written or generated automatically) without any exceptions.
7. Rule-based taggers have cyclic dependency problems. For example take the rule: *there are no two consecutive verbs*. If there are two words, each one can have verb and noun POSs tags, then we cannot get the decision from that rule, if there are no other rules to break the cyclic dependency: the tag for the first word depends on the tag for the second, and vice versa.

8.4 Implementation of an Arabic manually written rule-based tagger

There are many difficulties when we implement manually written rule-based tagger. The first is that in most cases, the tagger cannot select only one tag for each word. This restricts the possibility to combine this kind of tagger with other taggers. Another problem that is that the rules written by experts may have complicated forms and programming them in one form is difficult.

For simplification of the previous problems, we use a unified, restricted form of rules we implement. Complicated rules are first split into (perhaps several) simple rules, and only then implemented. All rules are used for eliminating unwanted tags for specific words in the context, so our goal will not be selecting the best tag. The unified form of our rules is:

"***if*** *conditions* ***then*** *eliminate (list of tags)*"

This form can be implemented in a simple way.

Here are some randomly selected samples from the rules used in the implementation and extracted from (AL-Bidhani) [3] (Al-Rajhi) [10] (Al-Hamlawy) [7] (Al-Galaiini) [6]:

- *"**if** the word is preceded by a reduction particle **then** eliminate (tags with POS<>noun and tags with case<>genitive)".*
- *"**if** the word preceded by Def particle **then** eliminate (tags with POS<>noun)"*
- *"**if** the word is at the beginning of a sentence and (POS=noun or (POS=verb & mood=present)) **then** eliminate (tags with case or mood<> nominative)*
- *"**if** the word follows a verb without 'Al' 'ال' **then** eliminate (tags with POS=adjective)*
- *"**if** the word is preceded by 'س' or 'سوف' particles **then** eliminate (tags with POS & Mood <> verb & present)*
- *"**if** 'ك' is a proclitic **then** eliminate (tags with POS & working <>particle & reduction)*
- *"**if** all the analyses of the preceding word have verb class **then** eliminate(tags with POS= verb)*
- *"**if** all the analyses of the following word have verb class **then** eliminate (tags with POS= verb)*
- *"**if** the preceding word tag has genitive case and the current word has 'ال' as a proclitic **then** eliminate (tags with case<> genitive)*

Many such rules are used for building our rule-based tagger. Building this tagger, collecting rules, building dictionary and the analyzer were the most time consuming tasks in this dissertation.

8.5 Combining manually written rule-based taggers

As we know, the first step in tagging is to assign all the possible tags to each word. Most of these tags may be eliminated almost immediately, it is a task for the rule-based tagger, which assists this way statistical taggers.

There are many methods for combining more than one tagger into one tagging system[40]. The most frequently used and easiest is voting. But it cannot be used according to our specification for rule-based tagger, which may leave several possible tags for a single word. Therefore, by using stacking technique, we can combine a rule-based tagger with a tagger constructed by master-slaves technique. Figure 8-1 shows this combination. An important note here is that stacking can be seen as a special case of master-slaves technique.

Using slaves, in master-slaves technique for a simple (fixed) factor, we modify the emission probabilities of a HMM tagger, multiplying them by a constant factor smaller than 1, except the tags produced by slaves. The operation we use now, eliminating tags, can be described as using the rule-based tagger as a slave with factor 0.

There is another reason to use rule-based tagger with master-slaves. In a rule-based tagger, the rules are used to eliminate unwanted tags, which in turn simplifies the task of the master tagger, since the eliminated tags need not be taken into account. This benefit arises when such a tag is selected by another slave tagger.

Figure 8-1 can be understood in two ways: that it presents a rule-based tagger attached as a slave with factor zero, or as a tagger combined using stacking technique. We prefer the first meaning because it is a part of our general technique of master-slaves. Of course, with factor zero the rule-based tagger can eliminate any tag completely, which causes the master tagger not to take it into account. Therefore it is a special, very powerful slave.

[40] See Chapter 7 for more details about taggers combination.

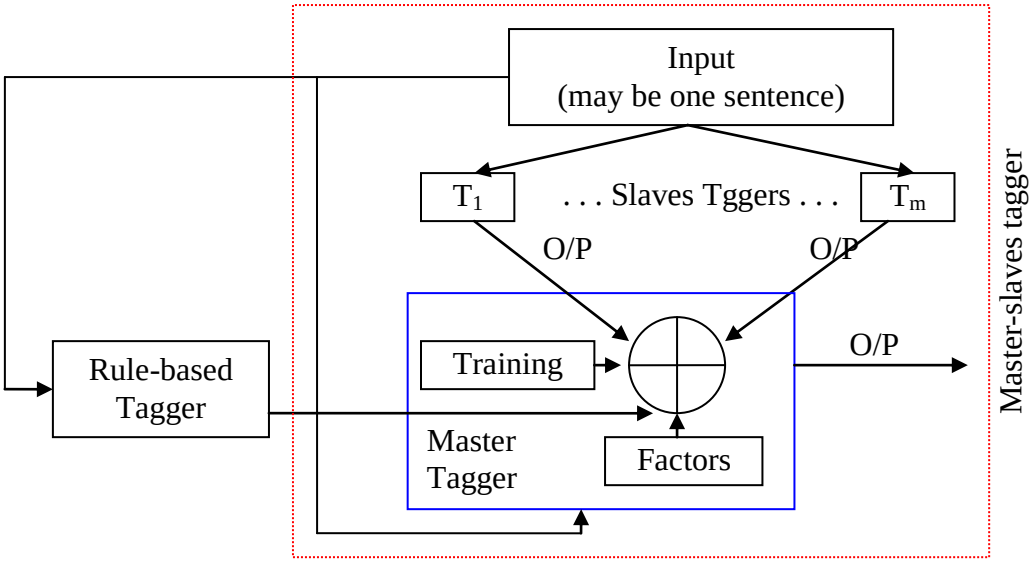


Figure 8-1: The overview of the tagging system.

8.6 Results and discussion

We applied the tagger described above to a data set of 45 files (29k words in total), annotated manually with our tagset. The result of using HMM, Brill and MM taggers combined as master-slaves was accuracy of 90.05 %. The accuracy after adding the rule-based tagger increased to 92.86 %.

We can see that the accuracy increased by using the rule-based tagger. The large increase of accuracy is most likely due to the fact that we use a small corpus, which leads to low accuracy of statistical methods. Using the rule-based tagger, which is independent of the type and size of corpus, increases the accuracy. We expect that when the size of corpus will increase, the gain of accuracy due to rule-based tagger will diminish.

Chapter 9

Results, Related Work and Future Work

The main goal of our dissertation was to construct a comprehensive tagging system, which can be used for annotating Arabic corpora.

In our dissertation, we did analytical study, implementation and evaluation of Arabic tagging system, starting from raw text to tag disambiguation. The system was implemented under a new very rich tagset, which was designed and developed by us. We split the tagging process to stand alone stages which simplified building the whole system.

Our corpus consists of 45 files with 29k word in total, annotated by our tagset. It was used as a training corpus for the statistical methods used in our tagger. The accuracy of the tagging was calculated assuming 100% correctness of tokenization, which required 1.2% of manual corrections.

9.1 Implementation

Our implementation, for all system stages, was done in the C# environment. There is one exception: the software testing the master-slaves technique on the Brown corpus was written in Python, because the Python taggers are freely available. Otherwise we used C# even though it does not contain any special library for NLP, for many reasons: the basic goal was to build the whole application for Arabic, using input and output without any transliteration. It is an

easy language comparing to other languages; it combines the power of C++ and simplicity of VB. An application with a rich and comfortable user interface, important for the annotator, can be created quite easily. The problem of Unicode when dealing with Arabic language does not exist. C# is also a relatively fast language. The last reason which caused us to select a language rarely used for NLP is that we built all the parts of the system: tokenizer, analyzer and tagger ourselves, and no parts of them were taken from existing resources.

It seems to us that manual correction of the tokenization output before tagging is desired. This work does not take much time, comparable to the time of just reading the text. This operation increases the accuracy of tagging, while manual correction of tagging results is definitely more time-consuming.

The most labour-intensive parts of our dissertation were its practical parts: building the dictionaries, the analyzer and collecting the rules for tagging. But the result seems worth the effort.

9.2 Results and discussion

Tagset: Designing a new Arabic tagset, suitable for Classical Arabic (CA) and Modern Standard Arabic (MSA), is a hard problem. In addition to the classical constructions in tag systems, we introduced interleaving of tags. Interleaving is a relation between tags which, in certain situations, can be attached to the same occurrence of a word, but each of them can also appear alone. Our tagset makes this relation explicit.

Tokenization: It is an initial task for almost all Arabic language processing applications. This task was achieved, in our system, by rule-based and statistical methods. We separated the tokenization process in order to simplify the tagging process. The accuracy of this stage was 98.8%. It is comparable to other similar works. Because it is an independent task, it can be modified without affecting the whole system. In order to increase the accuracy of the subsequent stages, the output of tokenization can be corrected manually which should take relatively little time.

Analysis and lemma extraction: as was mentioned in Chapter 5, the goals of this task are extracting all the analyses of the word and extracting the lemma. These analyses provide POS and features according to our tagset. Our analyzer

cannot be used independently, because it is specialized for the needs of our complete system. Because we use it for tagging, we evaluated its accuracy measuring how often the true analysis is among all analyses produced. For doing this evaluation we used a small corpus of 16 k words, manually annotated by a single analysis for each word, correct for this particular use of that word. In the test, for 99.67% of words, the correct analysis was among those produced by the analyzer. On the other hand, in a manual verification of the output of the analyzer, only 0.1% of all analyses were grammatically incorrect.

Tagging: We used two techniques of combining taggers, which are stacking and master-slaves techniques. The taggers used by these techniques are manually created rule-based tagger, HMM, Brill and MM taggers. HMM, Brill and MM taggers are combined with master-slaves technique, with HMM as the master and the other as slaves. Rules-based tagger is combined using stacking or, equivalently, as a special slave, with the master-slave tagger.

Master-slaves technique: Independently of the construction of the whole system, we have devised a new method for combining taggers, which is master-slaves technique. The HMM master tagger chooses the best tag according to its knowledge, which is modified by the results obtained by the slave taggers. This increases the accuracy when compared with normal HMM tagger, even above the level of the best accuracy achieved by the component taggers alone. The accuracies of using this technique are shown in Figure 9-1. The reader should remember that our tagest with several thousand tags is used, and the training corpus was relatively small, therefore the accuracy cannot be as high as in the cases of taggers using small tagsets and large corpuses.

We used a rules-based tagger for increasing the accuracy and eliminating unwanted tags. Relatively few rules were used in our tagger, and not all features of Arabic language were taken into account. Constructing the rules for this tagger was one of the most time consuming tasks. Implementation of these rules was not an easy task, if compared to the implementation of the statistical methods. The accuracy was increased to 92.86 % by adding the rule-based tagger to the master-slaves one.

Master or original	Slave	Brown corpus			Accuracy of Private Arabic corpus
		Total words	Matched words	Accuracy	
Brill	----------	1161192	1096687	94.44 %	86.43 %
Maxmatch	---------	1161192	1061635	91.5 %	83.26 %
HMM	---------	1161192	1106482	95.28 %	88.81 %
HMM	Brill	1161192	1109411	95.54 %	89.40 %
HMM	Brill+maxmatch	1161192	1111281	95.70 %	90.05 %

Figure 9-1: Accuracy of using HMM, Brill and MM in master-slaves combination.

9.3 Future work

The rules which used in tagging are of one form: "*If this tag not applicable to the present word for some reason, then delete it".* This form makes updating them easier. Surely, we did not use all rules known in Arabic, because not all of them can be represented in this form, and we did not have time and specialized knowledge to create the optimal set of rules. One direction of improving the system is to extend it by adding more rules written by experts.

The second obvious way to improve the performance of the system is to use much larger corpus for training. This large corpus can be updated in each cycle of running the system, as in Figure 9-2, where the output of the tagger, corrected by a human annotator, is added to the corpus.

We also have plans of using other methods of tagging Arabic, such as maximum entropy based tagger. In our opinion it is suitable for a highly inflected language, such as Arabic, and quite different in methodology, which gives a possibility of different results. It can be then used as a yet another slave in the master-slaves hybrid tagger. Using more slaves will affect the time of processing, but according to our plan of building an offline tagger, speed of processing is not a crucial factor.

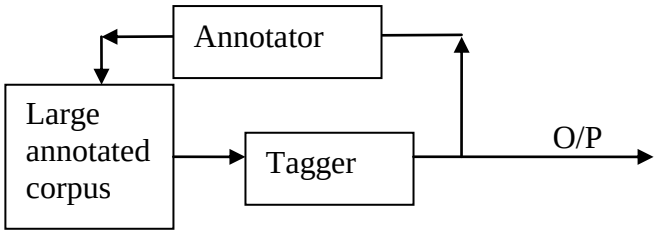


Figure 9-2: Corpus feedback.

We also think about building a tagger to use the third, syntactical level in our tagset. It will require knowledge of the Arabic syntax. The output of our system can be used as its input. The good news for this tagger is that in Arabic there are strong relations between the case of the class, and the syntactic class itself.

Finally, we will use our system as an application for annotating of Arabic texts taken from Iraqi media. We believe, in next two years, it will see the light for free availability.

APPENDIX A1

Arabic letters family Unicode

Appendix A2:

Arabic verb patterns

Table 1: Trilateral (merely and extra) verb pattern.

Verb Type	Verb form	Pattern Transliteration	Arabic Script
Merely	I	faEala – yafoEulu	فَـعَلَ – يَفْعُلُ
	I	faEala – yafoEilu	فَعَلَ – يَفْعِلُ
	I	faEala – yafoEalu	فَعَلَ – يَفْعَلُ
	I	faEila – yafoEalu	فَعِلَ – يَفْعَلُ
	I	faEula – yafoEulu	فَعُلَ – يَفْعُلُ
	I	faEila – yafoEilu	فَعِلَ – يَفْعِلُ
Merely +Extra one letter	II	faE~ala – yufaE~ilu	فَعَّلَ – يُفَعِّلُ
	III	faAEala – yufaAEilu	فَاعَلَ – يُفَاعِلُ
	IV	OafoEala – yufoEilu	أَفْعَلَ – يُفْعِلُ
Merely +Extra two letters	V	tafaE~ala – yatafaE~alu	تَفَعَّلَ – يَتَفَعَّلُ
	VI	tafaAEala – yatafaAEalu	تَفَاعَلَ – يَتَفَاعَلُ
	VII	AnofaEala – yanofaEilu	انْفَعَلَ – يَنْفَعِلُ
	VIII	AfotaEala – yafotaEilu	افْتَعَلَ – يَفْتَعِلُ
	IX	AfoEal~a – yafoEal~u	افْعَلَّ – يَفْعَلُّ
Merely +Extra three letters	X	AsotafoEala – yasotafoEilu	اسْتَفْعَلَ – يَسْتَفْعِلُ
	XI	AfoEaAl~a – yafoEaAl~u	افْعَالَّ – يَفْعَالُّ
	XII	AfoEawoEala – yafoEawoEalu	افْعَوْعَلَ – يَفْعَوْعَلُ
	XIII	AfoEaw~ala – yafoEaw~alu	افْعَوَّلَ – يَفْعَوَّلُ
	XIV	AfoEanolala - yafoEanolalu	افْعَنْلَلَ – يَفْعَنْلَلُ
	XV	AfoEanolaY - yafoEanolaY	افْعَنْلَى – يَفْعَنْلَى

Table 2: quadrilateral (merely and extra) verb pattern and the appendix to it from trilateral

Verb Type	Verb form	Pattern- Transliteration	Arabic script	appendix to it from trilateral Arabic script
Merely	I	**faEolala – yufaEolilu**	**فَعْلَلَ – يُفَعْلِلُ**	
		fawoEala – yufawoEilu		**فَوْعَلَ – يُفَوْعِلُ**
		fayoEala – yufayoEilu		**فَيْعَلَ – يُفَيْعِلُ**
		faEowala – yufaEowilu		**فَعْوَلَ – يُفَعْوِلُ**
		faEoyala – yufaEoyulu		**فَعْيَلَ – يُفَعْيُلُ**
		faEolala – yufaEolilu		**فَعْلَلَ – يُفَعْلِلُ**
		faEolaY – yufaEolaY		**فَعْلَى – يُفَعْلَى**
Extra one letter to merely	II	**tafaEolal – yatafaEolal**	**تَفَعْلَل – يَتَفَعْلَل**	
		tafaEolala – yatafaEolalu		**تَفَعْلَلَ – يَتَفَعْلَلُ**
		tafawoEala – yatafawoEalu		**تَفَوْعَلَ – يَتَفَوْعَلُ**
		tafayoEala – yatafayoEalu		**تَفَيْعَلَ – يَتَفَيْعَلُ**
		tafaEowala – yatafaEowalu		**تَفَعْوَلَ – يَتَفَعْوَلُ**
		tafaEolaY – yatafaEolaY		**تَفَعْلَى – يَتَفَعْلَى**
Extra two letter to merely	III	**AfoEanolal – yafoEanolalu**	**افْعَنْلَل – يَفْعَنْلَلُ**	
		AfoEanolal – yafoEanolalu		**افْعَنْلَل – يَفْعَنْلَلُ**
		AfoEanolaY – yafoEanolaY		**افْعَنْلَى – يَفْعَنْلَى**
		AftEolY – yaftaEolY		**افتعْلى – يَفْتَعْلى**
	IV	AfoEalal~a – yfoEalil~u	**افْعَلَلَّ – يفْعَلِلُّ**	

Appendix B:

Practical Text tagged by the proposed tagset

WEtagged practical text by the proposed tagset. The text was taken from Assabah journal (formal journal in Iraq). Date of publishing 19-03-2012. The title is "Ur Chaldeans"

مرة، وقبل سنتين، كتبت عن العراق الذي سوف يعمل على تغيير العالم، هل هذه كلمة كبيرة ومبالغ فيها وربما لم يسعف التعبير على وجه الدقة والوضوح من ان العراق القديم الكامن تحت الرمال والليشن، هو ذاك الذي سوف يغير العالم، واذا ارتأينا الفكرة في الواقع الفعلي، فأن العالم ومن خلال عشرة آلاف تل آثاري، لم يجر التنقيب فيها بالعراق، سوف يمنح اكاديميات الارض فرصة علمية لاستعادة ومن ثم تغيير تصوراتها ومفاهيمها في مختلف قضايا وشؤون الحياة والتاريخ.. اذن فالعالم سيغير نفسه من خلال العراق مثلما تغير حين اعاد الماركسيون النظر في تصوراتهم عن نمط الانتاج الاسيوي وفكرة نشوء الطبقات حالما اكتشف الاستشراق مدنا مثل سومر وبابل وآشور، وتحروا عند تفاصيلها انظمة تسجيل العبيد والاجراء والموظفين واشكال تنظيم العمل وادارة الدولة، ولو كان الاستشراق في زمن ماركس وانجلس قد توصل الى اكتشاف تلك المدن ودقائقها اليومية لما كتبا شيئا عن الارض المشاعة ومشكلة البزل اللذين حالا دون ارتقاء الملكية الفردية ومنعا من قيام الصراع الطبقي، وربما كانت الماركسية غيرها فى النظر الى الشرق والغرب لو كان الاستشراق فى المستوى التفصيلى كما جاء بعد ماركس.

Arabic word	Clitics and word base			Tag	Explanation
	Token	Transliteration	Translation		
مرة	مرة	mrp	Once ,Time	NNou_SFNN	Noun Common Singular Feminine Nominative Not Structured
،	،	,	,	CPnc	Punctuation
	و	w	And	PNon_Non	Particle Not_have_working have_No_meaning
وقبل	قبل	qbl	before	NAdv_SMAN	Noun Adverb Singular Masculine Accusative Not Structured
سنتين	سنتين	sntyn	Two years	NNou_DFGN	Noun Common Dual Feminine Genative Not Structured
،	،	,	,	CPnc	Punctuation
كتبت	كتبت	ktbt	I wrote	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
عن	عن	En	About	PRed_Adv	Particle For_Reduction have_meaning_of_Adverbial
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
العراق	عراق	ErAq	Iraq	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured
الذي	الذي	Al*y	Which	NRel_SMGY	Noun Relative Singular Masculine Genative Structured
سوف	سوف	swf	Will	PNon_Fut	Particle Not_have_working have_meaning_of_Future
يعمل	يعمل	yEml	Works	VPrt_3SMNNNA	Verb Present Third Singular Masculine Nominative Not Structured Not Certainty Active
على	على	ElY	At	PRed_Adv	Particle For_Reduction have_meaning_of_Adverbial

تغيير	تغيير	tgyyr	Changing	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
العالم	عالم	EAlm	world	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
،	،	,	,	CPnc	Punctuation
هل	هل	hl	Is, Are	PNon_Int	Particle Not_have_working have_meaning_of_Interrogative
هذه	هذه	h*h	This	NDem_SMGY	Noun Demostrative Singular Masculine Genative Structured
كلمة	كلمة	klmp	Word	NNou_SFNN	Noun Common Singular Feminine Nominative Not Structured
كبيرة	كبيرة	kbyrp	Large	NAdo_SFNN	Noun Adjective(Other) Singular Feminine Nominative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
ومبالغ	مبالغ	mbAlg	exaggerate	NAdo_SMNN	Noun Adjective(Other) Singular Masculine Nominative Not Structured
	في	fy	In	PRed_Adv	PRed_Adv
فيها	ها	hA	her	NPrn_SFGY	Noun Pronoum Singular Feminine Genative Structured
	و	w	And	PNon_Non	Particle Not_have_working have_No_meaning
	رب	rb	May	PNon_Crd	Particle Not_have_working have_meaning_of_increasing_decreasing
وربما	ما	mA	be	PPrv_Non	Particle For_Preventing have_No_meaning
لم	لم	lm	Not	PJus_Neg	Particle For_jusive have_meaning_of_Negative
يسعف	يسعف	ysEf	Ministering	VPrt_3SMJNNA	Verb Present Third Singular Masculine JussiveNot Structured Not Certainty Active
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
التعبير	تعبير	tEbyr	expression	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
على	على	ElY	At	PRed_Adv	Particle For_Reduction have_meaning_of_Adverbial
وجه	وجه	wjh	Face	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الدقة	دقة	dqp	accuracy	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
والوضوح	وضوح	wDwH	clarity	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
من	من	mn	Of	PRed_Non	Particle For_Reduction have_No_meaning
أن	أن	On	That	PCop_Cer	Particle For_copying have_meaning_of_Certainty
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
العراق	عراق	ErAq	Iraq	NPrp_SMAN	Noun Proper Singular Masculine Accusative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
القديم	قديم	qdym	Old	NAdo_SMAN	Noun Adjective(Other) Singular Masculine Accusative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الكامن	كامن	kAmn	latent	NAdo_SMAN	Noun Adjective(Other) Singular Masculine Accusative Not Structured
تحت	تحت	tHt	Under	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
	ال		The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الرمال	رمال	Al rmAl	sands	NNou_PMGN	Noun Common Plural Masculine Genative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
والليشن	ليشن	ly$n	launch	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
،	،	,	,	CPnc	Punctuation
هو	هو	hw	He	NPrn_SMNY	Noun Pronoum Singular Masculine Nominative Structured
ذاك	ذاك	*Ak	That	NDem_SMNY	Noun Demostrative Singular Masculine Nominative Structured
الذي	الذي	Al*y	Which	NRel_SMNY	Noun Relative Singular Masculine Nominative Structured
سوف	سوف	swf	Will	PNon_Fut	Particle Not_have_working have_meaning_of_Future
يغير	يغير	ygyr	Change	VPrt_3SMNNNA	Verb Present Third Singular Masculine Nominative Not Structured Not Certainty Active
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
العالم	عالم	EAlm	world	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured

،	،	,	,	CPnc	Punctuation
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
وإذا	إذا	I*A	if	PNon_Adv	Particle Not_have_working have_meaning_of_Adverbial
ارتأينا	ارتأينا	ArtOynA	We decided	VPst_1PCOYNA	Verb Past First Plural Common NonMood Structured Not Certainty Active
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الفكرة	فكرة	fkrp	idea	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
في	في	fy	In	PRed_Adv	Particle For_Reduction have_meaning_of_Adverbial
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الواقع	واقع	wAqE	reality	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الفعلي	فعلي	fEly	actual	NAdg_SMGN	Noun Adjective(Genealogical) Singular Masculine Genative Not Structured
،	،	,	,	CPnc	Punctuation
	ف	f	then	PNon_Lnk	Particle Not_have_working have_meaning_of_Linking
فأن	أن	On	that	PCop_Cer	Particle For_copying have_meaning_of_Certainty
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
العالم	عالم	EAlm	world	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
ومن	من	mn	from	PRed_Non	Particle For_Reduction have_No_meaning
خلال	خلال	xlAl	Through	NAdv_SMGN	Noun Adverb Singular Masculine Genative Not Structured
عشرة	عشرة	E$rp	Ten	NNod_SFGN	Noun Number(Ordinal) Singular Feminine Genative Not Structured
آلاف	آلاف	|lAf	Thousands	NNod_PMGN	Noun Number(Ordinal) Plural Masculine Genative Not Structured
تل	تل	tl	Hill	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
آثاري	آثاري	|vAry	Archaeologist	NAdg_SMGN	Noun Adjective(Genealogical) Singular Masculine Genative Not Structured
،	،	,	,	CPnc	Punctuation
لم	لم	lm	Not	PJus_Neg	Particle For_jusive have_meaning_of_Negative
يجر	يجر	yjr	happen	VPrt_3SMJNNA	Verb Present Third Singular Masculine JussiveNot Structured Not Certainty Active
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
التنقيب	تنقيب	tnqyb	exploration	NNou_SMNN	Noun Common Singular Masculine Nominative Not Structured
	في	fy	In	PRed_Adv	PRed_Adv
فيها	ها	hA	her	NPrn_SFGY	Noun Pronoum Singular Feminine Genative Structured
	ب	b	in	PRed_Adv	Particle For_Reduction have_meaning_of_Adverbial
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
بالعراق	عراق	ErAq	Iraq	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured
،	،	,	,	CPnc	Punctuation
سوف	سوف	swf	Will	PNon_Fut	Particle Not_have_working have_meaning_of_Future
يمنح	يمنح	ymnH	Gives	VPrt_3SMNNNA	Verb Present Third Singular Masculine Nominative Not Structured Not Certainty Active
اكاديميات	اكاديميات	AkAdymyAt	Academies	NNou_PFNN	Noun Common Plural Feminine Nominative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الارض	ارض	ArD	land	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
فرصة	فرصة	frSp	Opportunity	NAdo_SFAN	Noun Adjective(Other) Singular Feminine Accusative Not Structured
علمية	علمية	Elmyp	Scientific	NAdg_SFAN	Noun Adjective(Genealogical) Singular Feminine Accusative Not Structured
	ل	l	To	PRed_Cau	Particle For_Reduction have_meaning_of_Caution
لاستعادة	استعادة	AstEAdp	restore	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
	و	w	And	PCnj_Non	Particle For_Conjection have_No_meaning
ومن	من	mn	from	PRed_Non	Particle For_Reduction have_No_meaning
ثم	ثم	vm	Then	PCnj_Non	Particle For_Conjection have_No_meaning
تغيير	تغيير	tgyyr	Change	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
تصوراتها	تصورات	tSwrAt	her	NNou_PFGN	Noun Common Plural Feminine Genative Not Structured

	ها	hA	Perceptions	NPrn_SFGY	Noun Pronoum Singular Feminine Genative Structured
	و	w	and	PCnj_Non	Particle For_Conjection have_No_meaning
	مفاهيم	mfAhym	concepts	NNou_PFGN	Noun Common Plural Feminine Genative Not Structured
ومفاهيمها	ها	hA	her	NPrn_SFGY	Noun Pronoum Singular Feminine Genative Structured
في	في	fy	In	PRed_Non	Particle For_Reduction have_No_meaning
مختلف	مختلف	mxtlf	Different	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
قضايا	قضايا	qDAyA	Issues	NNou_PFGN	Noun Common Plural Feminine Genative Not Structured
	w	w	and	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
وشؤون	و شؤون	$Wwn	Affairs	NNou_PFGN	Noun Common Plural Feminine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الحياة	حياة	HyAp	life	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
	و	w	And	PCnj_Non	Particle For_Conjection have_No_meaning
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
والتاريخ	تاريخ	tAryx	date	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
..	..	..	..	CPnc	Punctuation
اذن	اذن	A*n	So	PNon_Ans	Particle Not_have_working have_meaning_of_Answer
	ف	f	then	PNon_Non	Particle Not_have_working have_No_meaning
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
فالعالم	عالم	EAlm	world	NNou_SMNN	Noun Common Singular Masculine Nominative Not Structured
	س	s	will	PNon_Fut	Particle Not_have_working have_meaning_of_Future
سيغير	يغير	ygyr	change	VPrt_3SMNNNA	Verb Present Third Singular Masculine Nominative Not Structured Not Certainty Active
	نفس	nfs	self	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
نفسه	ه	h	him	NPrn_SMGY	Noun Pronoum Singular Masculine Genative Structured
من	من	mn	from	PRed_Non	Particle For_Reduction have_No_meaning
خلال	خلال	xlAl	Through	NAdv_SMGN	Noun Adverb Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
العراق	عراق	ErAq	Iraq	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured
	مثل	mvl	Like	NNou_SMNN	Noun Common Singular Masculine Nominative Not Structured
مثلما	ما	mA	what	NRel_SMGY	Noun Relative Singular Masculine Genative Structured
تغير	تغير	tgyr	Changed	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
حين	حين	Hyn	When	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
اعاد	اعاد	AEAd	Re-	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الماركسيون	ماركسيون	mArksywn	Marxists	NAdg_PMNN	Noun Adjective(Genealogical) Plural Masculine Nominative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
النظر	نظر	nZr	view	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
في	في	fy	In	PRed_Non	Particle For_Reduction have_No_meaning
	تصورات	tSwrAt	Perceptions	NNou_PMGN	Noun Common Plural Masculine Genative Not Structured
تصوراتهم	هم	hm	them	NPrn_PMGY	Noun Pronoum Plural Masculine Genative Structured
عن	عن	En	About	PRed_Non	Particle For_Reduction have_No_meaning
نمط	نمط	nmT	Pattern	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الانتاج	انتاج	AntAj	production	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الاسيوي	اسيوي	Asywy	Asian	NAdg_SMGN	Noun Adjective(Genealogical) Singular Masculine Genative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
وفكرة	فكرة	fkrp	idea	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
نشوء	نشوء	n$w’	Emergence	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured

Appendix B

	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الطبقات	طبقات	TbqAt	layers	NNou_PFGN	Noun Common Plural Feminine Genative Not Structured
	حال	HAl	event	NAdv_SMAN	Noun Adverb Singular Masculine Accusative Not Structured
حالما	ما	mA	that	NRel_SMGY	Noun Relative Singular Masculine Genative Structured
اكتشف	اكتشف	Akt$f	Discover	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الاستشراق	استشراق	Ast$rAq	Orientalism	NAdg_SMNN	Noun Adjective(Other) Singular Masculine Nominative Not Structured
مدنا	مدنا	mdnA	Cities	NNou_PMAN	Noun Common Plural Masculine Accusative Not Structured
مثل	مثل	mvl	Such as	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
سومر	سومر	swmr	Sumer	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured
	و	w	and	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
وبابل	بابل	bAbl	Babylon	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
وآشور	آشور	|$wr	Assyria	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured
،	،	,	,	CPnc	Punctuation
	و	w	And	PCnj_Non	Particle For_Conjection have_No_meaning
وتحروا	تحروا	tHrwA	made inquiries	VPst_3PMOYNA	Verb Past Third Plural Masculine NonMood Structured Not Certainty Active
عند	عند	End	At	NAdv_SMAN	Noun Adverb Singular Masculine Accusative Not Structured
	تفاصيل	tfASyl	Details	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
تفاصيلها	ها	hA	here	NPrn_SFGY	Noun Pronoum Singular Feminine Genative Structured
انظمة	انظمة	AnZmp	Systems	NNou_PMAN	Noun Common Plural Masculine Accusative Not Structured
تسجيل	تسجيل	tsjyl	Registration	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
العبيد	عبيد	Ebyd	slaves	NNou_PMGN	Noun Common Plural Masculine Genative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
والاجراء	اجراء	AjrA’	action	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
والموظفين	موظفين	mwZfyn	staff	NNou_PMAN	Noun Common Plural Masculine Accusative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
واشكال	اشكال	A$kAl	forms	NNou_PMAN	Noun Common Plural Masculine Accusative Not Structured
تنظيم	تنظيم	tnZym	Organization	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
العمل	عمل	Eml	work	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
وادارة	ادارة	AdArp	management	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الدولة	دولة	dwlp	state	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
،	،	,	,	CPnc	Punctuation
	و	w	And	PNon_Non	Particle Not_have_working have_No_meaning
ولو	لو	lw	if	PNon_Con	Particle Not_have_working have_meaning_of_Conditional
كان	كان	kAn	Was	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الاستشراق	استشراق	Ast$rAq	Orientalism	NAdo_SMNN	Noun Adjective(Other) Singular Masculine Nominative Not Structured
في	في	fy	In	PRed_Adv	Particle For_Reduction have_meaning_of_Adverbial
زمن	زمن	zmn	Time	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
ماركس	ماركس	mArks	Marx	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured

	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
وانجلس	انجلس	Anjls	Angeles	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured
قد	قد	qd	May	PNon_Rlz	Particle Not_have_working have_meaning_of_Realization
توصل	توصل	twSl	Reach	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
الى	الى	AlY	To	PRed_Non	Particle For_Reduction have_No_meaning
اكتشاف	اكتشاف	Akt$Af	Discovery	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
تلك	تلك	tlk	That	NDem_SFGY	Noun Demostrative Singular Feminine Genative Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
المدن	مدن	mdn	cities	NNou_PFGN	Noun Common Plural Feminine Genative Not Structured
	و	w	and	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
	دقائق	dqA}q	minutes	NNou_PFGN	Noun Common Plural Feminine Genative Not Structured
ودقائقها	ها	hA	her	NPrn_SFGY	Noun Pronoum Singular Feminine Genative Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
اليومية	يومية	ywmyp	day	NAdg_SFGN	Noun Adjective(Genealogical) Singular Feminine Genative Not Structured
	ل	l	For	PNon_Non	Particle Not_have_working have_No_meaning
لما	ما	mA	what	PNon_Neg	Particle Not_have_working have_meaning_of_Negative
كتبا	كتبا	ktbA	they wrote	VPst_3DMOYNA	Verb Past Third Dual Masculine NonMood Structured Not Certainty Active
شيئا	شيئا	$y}A	Something	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
عن	عن	En	About	PRed_Non	Particle For_Reduction have_No_meaning
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الارض	ارض	ArD	land	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
المشاعة	مشاعة	m$AEp	Commons	NAdo_SFGN	Noun Adjective(Other) Singular Feminine Genative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
ومشكلة	مشكلة	m$klp	problem	NNou_SFGN	Noun Common Singular Feminine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
البزل	بزل	bzl	puncture	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
اللذين	اللذين	All*yn	Who	NRel_DMGY	Noun Relative Dual Masculine Genative Structured
حالا	حالا	HAlA	prevented	VPst_3DMOYNA	Verb Past Third Dual Masculine NonMood Structured Not Certainty Active
دون	دون	dwn	Below	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
ارتقاء	ارتقاء	ArtqA’	Upgrade	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الملكية	ملكية	mlkyp	Royal	NAdg_SFGN	Noun Adjective(Genealogical) Singular Feminine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الفردية	فردية	frdyp	individual	NAdg_SFGN	Noun Adjective(Genealogical) Singular Feminine Genative Not Structured
	و	w	And	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
ومنعا	منعا	mnEA	prevented	VPst_3DMOYNA	Verb Past Third Dual Masculine NonMood Structured Not Certainty Active
من	من	mn	from	PRed_Cau	Particle For_Reduction have_meaning_of_Caution
قيام	قيام	qyAm	standin up	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الصراع	صراع	SrAE	conflict	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الطبقي	طبقي	Tbqy	class	NAdg_SMGN	Noun Adjective(Genealogical) Singular Masculine Genative Not Structured
،	،	,	,	CPnc	Punctuation
	و	w	And	PNon_Non	Particle Not_have_working have_No_meaning
	رب	rb	may	PNon_Crd	Particle Not_have_working have_meaning_of_increasing_decreasing
وربما	ما	mA	be	PPrv_Non	Particle For_Preventing have_No_meaning

كانت	كانت	kAnt	Was	VPst_3SFOYNA	Verb Past Third Singular Feminine NonMood Structured Not Certainty Active
	ال	Al	The	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الماركسية	ماركسية	mArksyp	Marxist	NAdg_SFNN	Noun Adjective(Genealogical) Singular Feminine Nominative Not Structured
	غير	gyr	changed	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
غيرها	ها	hA	it	NPrn_SFAY	Noun Pronoum Singular Feminine Accusative Structured
في	في	fy	in	PRed_Non	Particle For_Reduction have_No_meaning
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
النظر	نظر	nZr	view	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
الى	الى	AlY	to	PRed_Non	Particle For_Reduction have_No_meaning
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الشرق	شرق	$rq	East	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	و	w	and	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
والغرب	غرب	grb	west	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
لو	لو	lw	If	PNon_Con	Particle Not_have_working have_meaning_of_Conditional
كان	كان	kAn	was	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
الاستشراق	استشراق	Ast$rAq	Orientalism	NAdo_SMNN	Noun Adjective(Other) Singular Masculine Nominative Not Structured
في	في	fy	in	PRed_Non	Particle For_Reduction have_No_meaning
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
المستوى	مستوى	mstwY	level	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured
	ال	Al	the	PNon_Def	Particle Not_have_working have_meaning_of_Definition
التفصيلي	تفصيلي	tfSyly	detailed	NAdo_SMGN	Noun Adjective(Other) Singular Masculine Genative Not Structured
	ك	k	as	PRed_Sim	Particle For_Reduction have_meaning_of_Simile
كما	ما	mA	what	NRel_SMGY	Noun Relative Singular Masculine Genative Structured
جاء	جاء	jA’	came	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMood Structured Not Certainty Active
بعد	بعد	bEd	after	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured
ماركس	ماركس	mArks	Marx	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured
.	.	.	.	CPnc	Punctuation

Appendix C:

Output of our analyzer for simple sentence

word			Tokens	Lemma	Analyzing / Tag	
Arabic	Transli-teration	Trans-lation				
اجتمع	AjtmE	Met	اجتمع	اِجتَمَعَ	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMoodStructured Not Certainty Active #
				اِجتَمَعَ	VImv_2SMOYNA	Verb Imperative Second Singular Masculine NonMoodStructured Not Certainty Active #
				اِجتَمَعَ	VPrt_1SMNNNA	Verb Present First Singular Masculine Nominative Not Structured Not Certainty Active #
				اِجتَمَعَ	VPrt_1SMJNNA	Verb Present First Singular Masculine JussiveNot Structured Not Certainty Active #
				اِجتَمَعَ	VPrt_1SMANNA	Verb Present First Singular Masculine Accusative Not Structured Not Certainty Active #
				اِجتَمَعَ	VPst_3SMOYNP	Verb Past Third Singular Masculine NonMoodStructured Not Certainty Passive #
				اِجتَمَعَ	VPrt_1SMNNNP	Verb Present First Singular Masculine Nominative Not Structured Not Certainty Passive #
				اِجتَمَعَ	VPrt_1SMJNNP	Verb Present First Singular Masculine JussiveNot Structured Not Certainty Passive #
				اِجتَمَعَ	VPrt_1SMANNP	Verb Present First Singular Masculine Accusative Not Structured Not Certainty Passive #
زعماء	zEmA’	Leaders	زعماء	زَعيمٌ	NAdo_PMNN	Noun Adjective(Other) Plural Masculine Nominative Not Structured #
				زَعيمٌ	NAdo_PMAN	Noun Adjective(Other) Plural Masculine Accusative Not Structured #
				زَعيمٌ	NAdo_PMGN	Noun Adjective(Other) Plural Masculine Genative Not Structured #
الدول	Aldwl	States	ال	ال	PNon_Def	Particle Not_have_working have_meaning_of_Definition #
			دول	دَوْلَةٌ	NNou_PFNN	Noun Common Plural Feminine Nominative Not Structured #
				دَوْلَةٌ	NNou_PFAN	Noun Common Plural Feminine Accusative Not Structured #
				دَوْلَة	NNou_PFGN	Noun Common Plural Feminine Genative Not Structured #
العربية	AlErbyp	The Arabic	ال	ال	PNon_Def	Particle Not_have_working have_meaning_of_Definition #
			عربية	عربي	NAdg_SFNN	Noun Adjective(Genealogical) Singular Feminine Nominative Not Structured #
				عربي	NAdg_SFAN	Noun Adjective(Genealogical) Singular Feminine Accusative Not Structured #
				عربي	NAdg_SFGN	Noun Adjective(Genealogical) Singular Feminine Genative Not Structured #
في	fy	In	في	في	PRed_Non	Particle For_Reduction have_No_meaning #
				في	PRed_Cau	Particle For_Reduction have_meaning_of_Caution #
				في	PRed_Adv	Particle For_Reduction have_meaning_of_Adverbial #
				في	NFiv_SMGN	Noun Five_Noun Singular Masculine Genative Not Structured #
				وَفى	VImv_2SFOYNA	Verb Imperative Second Singular Feminine NonMoodStructured Not Certainty Active #
بغداد	bgdAd	Baghdad	بغداد	بغداد	NPrp_SMNN	Noun Proper Singular Masculine Nominative Not Structured #
				بغداد	NPrp_SMAN	Noun Proper Singular Masculine Accusative Not Structured #
				بغداد	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured #
				بَغْدادُ	NNou_SMNN	Noun Common Singular Masculine Nominative Not Structured #
				بَغْدادَ	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured #
				بَغْدادِ	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured #
واجمعوا	wAjmEwA	And gather	و	و	PCnj_Lnk	Particle For_Conjection have_meaning_of_Linking #
			اجمعوا	و	PRed_Cer	Particle For_Reduction have_meaning_of_Certainty #
				و	PRed_Non	Particle For_Reduction have_No_meaning #
				أجمَعَ	VPst_3PMOYNA	Verb Past Third Plural Masculine NonMoodStructured Not Certainty Active #
				أجمَعَ	VImv_2PMOYNA	Verb Imperative Second Plural Masculine NonMoodStructured Not Certainty Active #
				أجمَعَ	VPst_3PMOYNP	Verb Past Third Plural Masculine NonMoodStructured Not Certainty Passive #
				جَمَعَ	VImv_2PMOYNA	Verb Imperative Second Plural Masculine NonMoodStructured Not Certainty Active #
على	ElY	To/on	على	على	PRed_Lnk	Particle For_Reduction have_meaning_of_Linking #
				على	PRed_Non	Particle For_Reduction have_No_meaning #
				على	PRed_Adv	Particle For_Reduction have_meaning_of_Adverbial #
				على	PRed_Cnd	Particle For_Reduction have_meaning_of_Conditional #
				على	PRed_Cau	Particle For_Reduction have_meaning_of_Caution #
				عَلَى	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMoodStructured Not Certainty Active #
ان	An	That	ان	إنَّ	PCop_Cer	Particle For_copying have_meaning_of_Certainty #
				أنَّ	PCop_Cer	Particle For_copying have_meaning_of_Certainty #
				إنْ	PNon_Non	Particle Not_have_working have_No_meaning #
				أنْ	PNon_Non	Particle Not_have_working have_No_meaning #
				إنْ	PNon_Neg	Particle Not_have_working have_meaning_of_Negative #
				إنْ	PNon_Non	Particle Not_have_working have_No_meaning #
				أنْ	PNon_Non	Particle Not_have_working have_No_meaning #

				أنْ	PAcu_Sub	Particle For_Accusative have_meaning_of_Subordinating #
				أنَّ	VPst_3SMOYNA	Verb Past Third Singular Masculine NonMoodStructured Not Certainty Active #
				أنّ	VPst_3SMOYNP	Verb Past Third Singular Masculine NonMoodStructured Not Certainty Passive #
				وَنَى	VPrt_1SMJNNA	Verb Present First Singular Masculine JussiveNot Structured Not Certainty Active #
يساندوا	ysAndwA	They support	يساندوا	سَانَدَ	VPrt_3PMJNNA	Verb Present Third Plural Masculine JussiveNot Structured Not Certainty Active #
				سَانَدَ	VPrt_3PMANNA	Verb Present Third Plural Masculine Accusative Not Structured Not Certainty Active #
				سَانَدَ	VPrt_3PMJNNP	Verb Present Third Plural Masculine JussiveNot Structured Not Certainty Passive #
				سَانَدَ	VPrt_3PMANNP	Verb Present Third Plural Masculine Accusative Not Structured Not Certainty Passive #
الربيع	AlrbyE	The spring	ال	ال	PNon_Def	Particle Not_have_working have_meaning_of_Definition #
			ربيع	ربيع	NPrp_SMNN	Noun Proper Singular Masculine Nominative Not Structured #
				ربيع	NPrp_SMAN	Noun Proper Singular Masculine Accusative Not Structured #
				ربيع	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured #
				ربيعٌ	NPrp_SMNN	Noun Proper Singular Masculine Nominative Not Structured #
				ربيعً	NPrp_SMAN	Noun Proper Singular Masculine Accusative Not Structured #
				ربيعٍ	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured #
				ربيعٌ	NNou_SMNN	Noun Common Singular Masculine Nominative Not Structured #
				ربيعً	NNou_SMAN	Noun Common Singular Masculine Accusative Not Structured #
				ربيعٍ	NNou_SMGN	Noun Common Singular Masculine Genative Not Structured #
العربي	AlErby	The Arabic	ال	ال	PNon_Def	Particle Not_have_working have_meaning_of_Definition #
			عربي	عربي	NPrp_SMNN	Noun Proper Singular Masculine Nominative Not Structured #
				عربي	NPrp_SMAN	Noun Proper Singular Masculine Accusative Not Structured #
				عربي	NPrp_SMGN	Noun Proper Singular Masculine Genative Not Structured #
				عُرْبِي	NAdg_SMNN	Noun Adjective(Genealogical) Singular Masculine Nominative Not Structured #
				عُرْبِي	NAdg_SMAN	Noun Adjective(Genealogical) Singular Masculine Accusative Not Structured #
				عُرْبِي	NAdg_SMGN	Noun Adjective(Genealogical) Singular Masculine Genative Not Structured #
				عَرَبِي	NAdg_SMNN	Noun Adjective(Genealogical) Singular Masculine Nominative Not Structured #
				عَرَبِي	NAdg_SMAN	Noun Adjective(Genealogical) Singular Masculine Accusative Not Structured #
				عَرَبِي	NAdg_SMGN	Noun Adjective(Genealogical) Singular Masculine Genative Not Structured #
				عَرَبِيٌ	NAdg_SMNN	Noun Adjective(Genealogical) Singular Masculine Nominative Not Structured #
				عَرَبِيً	NAdg_SMAN	Noun Adjective(Genealogical) Singular Masculine Accusative Not Structured #
				عَرَبِيٍ	NAdg_SMGN	Noun Adjective(Genealogical) Singular Masculine Genative Not Structured #
				عَرَبِي	NAdg_SMNN	Noun Adjective(Genealogical) Singular Masculine Nominative Not Structured #
				عَرَبِي	NAdg_SMAN	Noun Adjective(Genealogical) Singular Masculine Accusative Not Structured #
				عَرَبِي	NAdg_SMGN	Noun Adjective(Genealogical) Singular Masculine Genative Not Structured #
				عَرَّبَ	VImv_2SFOYNA	Verb Imperative Second Singular Feminine NonMoodStructured Not Certainty Active #

References

[1]. Abney S., Schapire R. and Singer Y. (1999). ***Boosting Applied to Tagging and PP Attachment***. Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, USA.

[2]. Alba E., Luque G., Araujo L. (2006). ***Natural language tagging with genetic algorithms***. Information Processing Letters journal, Volume 100 (No 5).

[3]. AL-Bidhani S. (2000). ***Nuzhat Altarf for explanation of verb construction in the science of morphology***. (by) Al-Ain, UAE, (Arabic Book).

[4]. Al-Dahdah A. (1989). ***Lexicon of Arabic language Grammar in tables and tablets***. 4th edition. Maktabat-Lebnan-Nashiroon, Beirut, Lebanon, (Arabic Book).

[5]. AlGahtani S., Black W., and McNaught J. (2009). ***Arabic part-of-speech-tagging using transformation-based learning***. Proceeedings of the 2nd International Conference on Arabic Language Resources and Tools, Cairo, Egypt.

[6]. Al-Galaiini M. (1990). ***Jamia Al-drooss Al-Arabia*** 1st edition, (by) Dar Al-Karkh, (Arabic book).

[7]. Al-Hamlawy A.(1957). ***Shaza Al-Orf in the art of morphology***. (by) Dar Al-Kiaan, Riadh, KSA, (Arabic book).

[8]. Al-Moradi I. (1992). ***Al-Juna Al-Dani in particles of meaning***. 1st edition, (by) Dar al-kotob al-ilmiyah, Beirut, Lebanon, (Arabic Book).

[9]. AlQrainy S. and Ayeshi A. (2006). ***Developing a tagset for automated POS tagging in Arabic***. WSEAS Transactions on Computers Vol 5.

[10]. Al-Rajhi A. (1979). ***The application of morphology***. (by) Dar Al-Nahdha Al-Arabia Beirut, (Arabic book).

[11]. AL-Shamsi F. and Guessoum A. (2006). ***A Hidden Markov Model-Based POS Tagger for Arabic***. *8es Journees Internationales d'Analyse statistique des Donnees Textuelles.*

[12]. Al-Sughaiyer I. and Al-Kharashi I. (2004). ***Arabic morphological analysis techniques: A comprehensive survey***. Journal of the American Society for Information Science and Technology Vol. 55 (No. 3).

[13]. Attia M. (2007). ***Arabic tokenization system***. Proceedings of the Workshop on Computational Approaches to Semitic Languages (Semitic '07): Common Issues and Resources. Stroudsburg, PA, USA.

[14]. Atwell E. (2008). ***Development of tag sets for part-of-speech tagging***. Ludeling A, Kyto M (ed.) Corpus Linguistics: An International Handbook, Vol 1**,** Mouton de Gruyter.

[15]. Atwell E., Al.Sulaiti L., Al.Osaimi S. and Abu.Shawar B. (2004). ***A Review of Arabic Corpus Analysis Tools***. Proceedings of JEP.TALN'04 Arabic Language Processing, Fes, Morocco.

[16]. Badr I., Zbib R. and Glass J. (2008). ***Segmentation for English-to-Arabic Statistical Machine Translation***. Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies. Columbus, Ohio, USA.

[17]. Beesley K. (1996). ***Arabic finite-state morphological analysis and generation***. In Proceedings of the 16th International Conference on Computational Linguistics (COLING-96), Copenhagen, Denmark.

[18]. Beesley K. (2001). ***Finite-State Morphological Analysis and Generation of Arabic at Xerox Research***: Proceedings of the Arabic Language Processing: Status and Prospect: 39th Annual Meeting of the Association for Computational Linguistics. Toulouse, France.

[19]. Benajiba Y. and Zitouni I. (2010). ***Arabic Mention Detection: toward better unit of analysis.*** Proceeding of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics. Stroudsburg, PA, USA.

[20]. Benajiba Y., Zitouni I. (2010): Arabic ***Word Segmentation for Better Unit of Analysis***. Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association (ELRA), Malta.

[21]. Bin-Muqbil M. (2006). ***Phonetic and Phonological Aspects of Arabic Emphatics and Gutturals***. PhD dissertation, University of Wisconsin-Madison, USA.

[22]. Bird S., Klein E. and Loper E. (2009). ***Natural Language Processing with Python***. (by) Published by O'Reilly Media, USA.

[23]. Bosch A., Marsi E., and Soudi A. (2007). ***Memory-based morphological analysis and part-of-speech tagging of Arabic***. Arabic Computational Morphology: knowledge-based and empirical methods, Kluwer / Springer Publications.

[24]. Brill E. (1995). ***Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging***. Computational Lingustics, Vol 21(No. 4).

[25]. Buckwalter T. (2002). ***Buckwalter Arabic Morphological Analyzer Version 1.0***. Linguistic Data Consortium, University of Pennsylvania.

[26]. Buckwalter T. (2004). ***Isues in Arabic Orthography and morphology Analysis***. Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages, USA.

[27]. Cavalli-Sforza V., Soudi A. and Mitamura T. (2000). ***Arabic Morphology Generation Using a Concatenative Strategy***. Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (NAACL 2000). Seattle, Washington, USA.

[28]. Chanod J. and Tapanainen P. (1996): ***A Non-deterministic Tokeniser for Finite-State Parsing***. Proceeding of 12th European Conference on Artificial Intelligence, Budabest, Hungary.

[29]. Clark A. (2003). ***Combining distributional and morphological information for part of speech induction***. Proceeding of 10th of Annual meeting of EACL, Budapest, Hungary.

[30]. Daelemans W., Zavrel J., Berck P. and Gillis S. (1996). ***MBT: A memory-based part of speech tagger generator***. Proceedings of the Fourth Workshop on Very Large Corpora / ACL SIGDAT, Cobenhagen, Denmark.

[31]. Darwish K. (2002): ***Building a Shallow Arabic Morphological Analyzer in One Day***. Proceedings of the ACL-02 workshop on Computational approaches to Semitic languages. PA, USA.

[32]. Diab M. (2009). ***Second Generation AMIRA Tools for Arabic Processing: Fast and Robust Tokenization, POS tagging, and Base Phrase Chunking***. Proceedings of the Second International Conference on Arabic Language Resources and Tools, Cairo, Egypt.

[33]. Diab M., Hacioglu K. and Jurafsky D. (2004). ***Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks.*** Proceedings of Human Language Technology Conference (HLT-NAACL), Boston, Massachusetts, USA.

[34]. Diab M., Hacioglu K. and Jurafsky D. (2007). ***Automated Methods for Processing Arabic Text: From Tokenization to Base Phrase Chunking***. Arabic Computational Morphology: Knowledge-based and Empirical Methods. Kluwer / Springer Publications.

[35]. Dichy J. (2001). ***On lemmatization in Arabic, A formal definition of the Arabic entries of multilingual lexical databases***. Proceding in Arabic NLP Workshop at ACL/EACL, Toulouse, France.

[36]. El Hadj Y., Al-Sughayeir I. and Al-Ansari A. (2009). ***Arabic Part-Of-Speech Tagging using the Sentence Structure***. Proceedings of the Second International Conference on Arabic Language Resources and Tools, Cairo, Egypt.

[37]. El-Dahdah A. (1994): ***An Intermediate Dictionary of Verb Conjugation***. 1st edition, Libaririe Du Liban Publisher, Beirut, Lebnan. (Aabic book).

[38]. Elhadj Y. (2009). ***Statistical Part-of-Speech Tagger for Traditional Arabic Texts***. Journal of Computer Science Vol5 (No.11).

[39]. El-Shishtawy T. and El-Ghannam F. (2012). ***An Accurate Arabic Root-Based Lemmatizer for Information Retrieval Purposes***. International Journal of Computer Science Issues(IJCSI), Vol. 9, (No. 1).

[40]. Elworthy D. (1995). ***Tagset Design and Inflected Languages***. *In: Proceedings of the ACL SIGDAT Workshop*, Dublin.

[41]. Feldman A. (2008) .**Tagset Design, Inflected Languages, and N-gram Tagging**. The Linguistics Journal Vol. 3 (No. 1).

[42]. Freeman A. (2001): ***Brill's POS tagger and a morphology parser for Arabic***. Proceding of ACL/EACL-Workshop on Arabic Language Processing: Status and Prospects, Toulouse, France.

[43]. Glass K. and Bangay S. (2005). ***Evaluating parts-of-speech taggers for use in a text-to-scene conversion system***. Proceeding of SAICSIT, White River, South Africa.

[44]. Gridach M. and Chenfour N. (2011). ***Developing a New System for Arabic Morphological Analysis and Generation***. Proceedings of the 2nd Workshop on South and Southeast Asian Natural Language Processing (WSSANLP)-IJCNLP, Chiang Mai, Thailand.

[45]. Habash N. (2010). ***Introduction to Arabic Natural Language Processing***. Synthesis Lecture on Human Langauge Technologies. A Publication in the Morgan & Claypool Publishers series, UAS.

[46]. Habash N. and Rambow O. (2005). ***Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop***. Proceedings of the 43rd Annual Meeting of the ACL, Michigan, USA.

[47]. Habash N. and Roth R. (2009). ***CATiB: The Columbia Arabic Treebank.*** Proceedings of the ACL-IJCNLP, Suntec, Singapore.

[48]. Habash N. and Sadat F. (2006). ***Arabic Preprocessing Schemes for Statistical Machine Translation***. published in the Proceedings of Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT/NAACL), New York, USA.

[49]. Habash N., Faraj R., and Roth R. (2009). ***Syntactic Annotation in the Columbia Arabic Treebank***. In Proceedings of MEDAR International Conference on Arabic Language Resources and Tools, Cairo, Egypt.

[50]. Habash N., Rambow O. and Kiraz G. (2005). ***Morphological Analysis and Generation for Arabic Dialects***. Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, Michigan, USA.

[51]. Habash N., Rambow O. and Roth R. (2009). ***MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological, Disambiguation, POS Tagging, Stemming and Lemmatization*****.** Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt.

[52]. Henrich V., Reuter T. and Loftsson H. (2009): ***CombiTagger: A System for Developing Combined Taggers***. Proceedings of the Twenty-Second International FLAIRS Conference, Sanibel Island, Florida, USA**.**

[53]. Jackson P. and Moulinier I. (2002). ***Natural Language Processing for Online Applications Text Retrieval Extraction and Categorization***. John Benjamins Publishing Company, Amsterdam, Philadelphia.

[54]. Jurafsky D. and Martin J. (2008). ***Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition***. (by) Prentice Hall, USA.

[55]. Karlsson F., Voutilainen A., Heikkila J. and Anttila A. (1995). ***Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text***. Mouton de Gruyter, berlin, Germany.

[56]. Khoja S. (1999): ***Stemming Arabic Text***. Computing Department, Lancaster University, Lancaster, U.K.

[57]. Khoja S. (2001). ***APT: Arabic Part-of-Speech Tagger***. In Proceedings *Student* Workshop at the Second Meeting of (NAACL2001), Pittsburgh, Pennsylvania.

[58]. Khoja S. and Garside R. (1999). ***Stemming Arabic Text***. Lancaster, UK, Computing Department, Lancaster University. http://www.*comp.lancs.ac.uk*/*computing*/*users*/*khoja*/*stemmer*.*ps*.

[59]. Khoja S., Garside R., and Knowles G. (2001). **A tagset for themorphosyntactic tagging of Arabic**. In Proceedings of Corpus Linguistics, Lancaster, UK.

[60]. Kim J., Kim G. (1996): ***Fuzzy Network Model for Part-of-Speech Tagging under Small Training Data***. Natural Language Engineering, Vol. 2 (No 2).

[61]. Klein S. and Simpson R. (1963). ***A computational approach to grammatical coding of English words***. Journal of ACM Vol. 10(No. 3).

[62]. Köprü S. (2011): ***An efficient part-of-speech Tagger for Arabic.*** Proceedings of the 12th international conference on Computational linguistics and intelligent text processing (CICLing'11), Tokyo, Japan.

[63]. Kuba A., Felföldi L. and Kocsor A. (2005). ***POS tag-ger combinations on Hungarian text***. Proceedings of the 2nd International Joint Conference on Natural LanguageProcessing (IJCNLP-05), Heidelberg, Germany.

[64]. Kulick S. (2010). ***Simultaneous Tokenization and Part-of-Speech Tagging for Arabic without a Morphological Analyzer***. Proceedings of the Association for Computational Linguistics (ACL) Conference Short Papers, Uppsala, Sweden.

[65]. Kulick S., Gabbard R., and Marcus M. (2006). ***Parsing the Arabic Treebank: Analysis and Improvements***. Proceedings of theTreebanks and LinguisticTheories Conference, Prague, Czech Republic.

[66]. Lee Y., Papineni K. and Roukos S. (Emam O. and Hassan H.) (2003). ***Language Model Based Arabic Word Segmentation***. Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics. Sapporo, Japan.

[67]. Loftsson H. (2006): ***Tagging Icelandic text: An experiment with integrations and combinations of taggers***. Language Resources and Evaluation Vol. 40 (No. 2).

[68]. Manning C. and Schütze H. (1999). ***Foundations of Statistical Natural Language Processing***. (by) MIT Press. Cambridge, London, UK.

[69]. Mansour S., Sima'an K. and Winter Y. (2007). ***Smoothing a Lexicon-based POS Tagger for Arabic and Hebrew***. Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages (Semitic '07): Common Issues and Resources, Prague, Czech Republic.

[70]. Mayfield J., McNamee P., Piatko C. and Pearce C. (2003): ***Lattice-based Tagging using Support Vector Machines***. Proceedings of the twelfth international conference on Information and knowledge management (CIKM '03), Louisiana, USA.

[71]. Mohamed E. and Kübler S. (2010). ***Arabic Part of Speech Tagging***. Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta.

[72]. Nakagawa T., Kudoh T. and Matsumoto Y. (2001). ***Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines***. Proceedings of the 6th Natural Language Processing Pacific Rim Symposium, Tokyo, Japan.

[73]. Nitin I. and Fred J. (2010). ***Handbook of Natural Language Processing, Second Edition***. Chapman & Hall/CRC Machine Learning & Pattern Recognition, USA.

[74]. Nugues P. (2006). ***An Introduction to Language Processing with Perl and Prolog***. (by) Springer-Verlag, Berlin Heidelberg, Germany.

[75]. Padró L. (1996): ***POS tagging using relaxation labeling***. Proceedings of the 16th conference on Computational linguistics (COLING), Copenhagen, Denmark.

[76]. Peng F., Feng F. and McCallum A. (2004). ***Chinese segmentation and new word detection using conditional random fields***. In Proceedings of the 20th international conference on Computational Linguistics (COLING '04), University of Geneva, Switzerland.

[77]. Ratnaparkhi A. (1996). ***A Maximum Entropy Model for Part-Of-Speech Tagging***. Proceedings of the Empirical Methods in Natural Language Processing Conference (EMNLP), University of Pennsylvania, USA.

[78]. Ratnaparkhi A. (1998). ***Maximum entropy models for natural language ambiguity resolution***. PhD dissertation, University of Pennsylvania, Philadelphia.

[79]. Roche E. and Schabes Y. (1995). ***Deterministic part-of-speech tagging with finite-state transducers***. In: Computational Linguistics Journal http://dl.acm.org/citation.cfm?id=211200, Vol. 21 (No. 2).

[80]. Ryding K. (2005). ***A Reference Grammar of Modern Standard Arabic***. (by) Unversity Press, Cambridge, UK.
[81]. Sakhr Software, **Arabic Morphological Analyzer** http://www.sakhr.com.
[82]. Sawalha M. (2011). ***Open-source Resources and Standards for Arabic Word Structure Analysis:*** Fine Grained Morphological Analysis of Arabic Text Corpora TAGGING. PhD dissertation, School of Computing, University of Leeds, UK.
[83]. Schmid H. (1994). ***A Probabilistic Part-of-Speech Tagging Using Decision Trees***. *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.
[84]. Schmid H. (1994). ***Part-of-Speech Tagging with Neural Networks***. *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, Kyoto, Japan.
[85]. Shaalan K. (2010). ***Rule-based Approach in Arabic Natural Language Processing***. International Journal on Information and Communication Technologies, Vol. 3(No. 3).
[86]. Silfverberg M. and Lindén K. (2010). ***Part-of-Speech Tagging Using Parallel Weighted Finite-State Transducers***. Proceeding of 7th International Conference on Natural Language Processing (IceTAL), Reykjavik, Iceland.
[87]. Sjobergh J. (2003). ***Stomp, a POS-tagger with a Different View***. Proceeding of Recent Advances in Natural Language Processing (RANLP-2003). Borovets, Bulgaria.
[88]. Smrz O. (2007). ***Functional Arabic Morphology, Formal System and Implementation***. Ph.D. thesis, institute of formal and applied Linguistics, Faculty of mathematics and physics, Charles University in Prague.
[89]. Tlili-Guiassa Y. (2006): ***Hybrid Method for Tagging Arabic Text***. Journal of Computer Science Vol 2 (No 3).
[90]. Toutanova K., Klein D., Manning C. and Singer Y. (2003). ***Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network***. Proceeding of Human Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada.
[91]. Wu D., Ngai G. and Carpuat M. (2003). ***A Stacked, Voted, Stacked Model for Named Entity Recognition***. precceding of 7th Conference on Natural Language Learning (CoNLL-2003), Edmonton, Canada.
[92]. Yonghui G., Baomin W., Changyuan L., and Bingxi W. (2006). ***Correlation voting fusion strategy for part of speech tagging***. Proceeding of 8th International Conference on Signal Processing (ICSP), IEEE conference, Guilin, China.