

AUTOMATIC DIACRITIZER FOR ARABIC TEXTS

By

Mohammad Ahmed Sayed Ahmed Ahmed Al Badrashiny

A Thesis Submitted to the
Faculty of Engineering, Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
ELECTRONICS & ELECTRICAL COMMUNICATIONS

Faculty of Engineering, Cairo University

Giza, Egypt

June 2009

AUTOMATIC DIACRITIZER FOR ARABIC TEXTS

By

Mohammad Ahmed Sayed Ahmed Ahmed Al Badrashiny

A Thesis Submitted to the
Faculty of Engineering, Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
ELECTRONICS & ELECTRICAL COMMUNICATIONS

UNDER THE SUPERVISION OF

Mohsen Abdul Raziq Ali Rashwan

Professor
Faculty of Engineering
Cairo University

Faculty of Engineering, Cairo University

Giza, Egypt

June 2009

AUTOMATIC DIACRITIZER FOR ARABIC TEXTS

By
Mohammad Ahmed Sayed Ahmed Ahmed Al Badrashiny

A Thesis Submitted to the
Faculty of Engineering, Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
ELECTRONICS & ELECTRICAL COMMUNICATIONS

Approved by the Examining Committee

Prof. Dr. Mohsen Abdul Raziq Ali Rashwan

Main Supervisor

Prof. Dr. Aly Aly Fahmy

Prof. Dr. Mohamed Waleed Talaat Fakhr

Faculty of Engineering, Cairo University

Giza, Egypt

June 2009

In the name of Allah, Most Gracious, Most Merciful

“Praise be to Allah, Who hath guided us to this (felicity): never could we have found guidance, had it not been for the guidance of Allah”

THE HOLY QURAN: AL A'ARAF (43)

Acknowledgements

All praise is due to Allah, who guided me to this.

I would like to express my sincere gratitude to my supervisor; Dr. Mohsen Rashwan. I'm greatly indebted to his assistance, guidance and support.

Thanks to RDI company for their permission to use their Arabic NLP tools. I would like to thank also Dr. Mohamed Attia and Eng. Ibrahim Sobh for their generous help.

I would like to thank the Arabic language experts Hussein Al Bassomy, Youssef Al Tehamy and Amr Al jendy for their help in collecting and annotating the data and system evaluation experiments.

I am very grateful for my dear parents, wife, daughter and my friends whom I consider as my brothers. Thank you all for being always there when I needed you most. Thank you for believing in me and supporting me. I believe that without your support and your prayers, none of this work would be accomplished. Finally, I hope this thesis be a useful addition to the research activities of Arabic natural language processing.

*For my parents,
My wife “Shimaa” and my daughter “Judy”*

List of Contents

Acknowledgements.....	V
List of Figures.....	IIX
List of Tables.....	X
List of Abbreviations and Acronyms.....	XI
Abstract.....	XIII
Chapter 1:	
Introduction.....	1
1.1 Automatic Arabic Text Diacritization Problems and Importance.....	2
1.2 Linguistic and Historical Background.....	4
1.3 Survey.....	5
1.4 Challenges and Points of Innovation.....	7
1.4.1 Challenges.....	7
1.4.2 Innovation Points in Our Work.....	8
1.5 The Presented System.....	8
1.6 Thesis Outline.....	9
Chapter 2:	
Background On The Arabic Factorization Model Used In Our System.....	11
2.1 Arabic Morphological Analysis.....	12
2.2 Arabic POS Tagging.....	14
2.3 Morphological and Syntactical Diacritization According to <i>ArabMorpho</i> [®] ver.4 View point.....	17
2.4 Morphological and Syntactical Diacritization According to our View point using <i>ArabMorpho</i> [®] ver.4.....	19
Chapter 3:	
Statistical Disambiguation Methods.....	22
3.1 Maximum a Posteriori Probability Estimation.....	23
3.2 Probability Estimation Via Smoothing Techniques.....	24
3.3 Disambiguation Technique.....	26

Chapter 4:	
Disambiguating a Hybrid of Unfactorized and Factorized Words' Sequences.....	30
4.1 The Offline Phase	32
4.1.1 Dictionary Building Process	33
4.1.2 Language Model Building Process	38
4.2 The Runtime Phase.....	38
Chapter 5:	
Experimental Results	40
5.1 A Comparison with the Recent Related Work	41
5.2 Experimental Setup	43
5.3 The effect of the corpus size on the system behavior.....	45
5.4 Experiments Design and Results Analysis	47
5.4.1 Experiment no. 1:.....	47
5.4.2 Experiment no. 2:.....	48
5.4.3 Experiment no. 3:.....	48
5.4.4 Experiment no. 4:.....	49
5.4.5 Experiment no. 5:.....	50
5.5 Errors Analysis	51
Chapter 6:	
Conclusion and Future Work	55
6.1 Conclusion.....	56
6.2 Future Work	57
References.....	58
The list of References in English.....	59
قائمة المراجع العربيّة.....	62

List of Figures

Chapter 1:

Figure 1.1: “Un-analyzable Segments” in input text.	9
--	---

Chapter 2:

Figure 2.1: Classifying the 9 types of morphemes in the Arabic lexicon of ArabMorpho [®] ver.4.	13
Figure 2.2: The Arabic lexical disambiguation trellis.	17
Figure 2.3: The Arabic POS Tags–Syntactic Diacritics search trellis.	18
Figure 2.4: Disambiguation lattice for morphological disambiguation, syntactic diacritization.	19
Figure 2.5: The Arabic morphological analyses–Syntactic Diacritics search trellis	21
Figure 2.6: The architecture of Arabic diacritizer statistically disambiguating factorized Arabic text.	21

Chapter 3:

Figure 3.1: The ambiguity of multiple solutions of each word in the input text W leading to a solution trellis of possible analyses $(a_1 \times a_2 \times \dots \times a_L)$	23
---	----

Chapter 4:

Figure 4.1: The hybrid Arabic text diacritization architecture disambiguating factorized and full-form words.	31
Figure 4.2: The hybrid Arabic text diacritization architecture disambiguating factorized and full-form words.	32
Figure 4.3: Dictionary structure.	35
Figure 4.4: Dictionary searching criterion.	37
Figure 4.5: Phrase segmentation.	39

Chapter 5:

Figure 5.1: Corpus size versus the out of vocabulary OOV.	47
--	----

List of Tables

Chapter 1:

Table 1.1: Arabic diacritics set.	3
--	---

Chapter 2:

Table 2.1: exemplar arabic morphological analyses.....	14
Table 2.2: arabic pos tags set.....	15
Table 2.3: pos labels of sample arabic lexemes.....	16
Table 2.4: pos tags-vectors of sample arabic words.	17

Chapter 5:

Table 5.1: propose hybrid diacritizer morphological & syntactical (case ending) diacritization error rates versus other state-of-the-art systems; our best results are shown in boldface.....	43
Table 5.2: distribution of trn_db_i over diverse domains.....	44
Table 5.3: distribution of tst_db over diverse domains.	45
Table 5.4: the effect of the training corpus size on the dictionary size.	46
Table 5.5: the effect of the training corpus size on the number of analyses per word.	46
Table 5.6: morphological & syntactic diacritization accuracies of the factorizing diacritizer versus the hybrid one.	47
Table 5.7: morphological and syntactic diacritization error rate of the hybrid diacritizer at large training data.	48
Table 5.8: studying the effect of the training data size changing on different domains... ..	49
Table 5.9: shares of the factorizing & un-factorizing diacritization error rates in the hybrid diacritization error rate.	50
Table 5.10: studying the effect of the increase of the training data on the memory size.	51
Table 5.11: studying the time consumption by the factorizing and the hybrid systems... ..	51
Table 5.12: morphological errors analyses.	53
Table 5.13: syntactical errors analyses.	54

List of Abbreviations and Acronyms

1. **ASR**: Automatic Speech Recognition.
2. **DER**: Diacritics Error Rate.
3. **EM**: Expectation Maximization.
4. **L-Tagging model**: is an Arabic diacritizer system that uses the “Lexemes citation form” in the diacritization process and if there exists (OOV) in the stems.
5. **Morpho_WER_{fac}**: The morphological word error rate of the factorizing system.
6. **Morpho_WER_{un-fac}**: The morphological word error rate of the un-factorizing system.
7. **OCR**: Optical Character Recognition.
8. **OOV**: Out-Of-Vocabulary.
9. **OOV_Morpho_WER_{un-fac}**: The morphological word error rate of the un-factorizing system due to Out-Of-Vocabulary.
10. **OOV_Synta_WER_{un-fac}**: The syntactical word error rate of the un-factorizing system due to Out-Of-Vocabulary.
11. **POS - Tagging**: Part Of Speech – Tagging.
12. **Q-Tagging model**: is an Arabic diacritizer system that uses the “Quadruple citation form” in the diacritization.
13. **SLM**: Statistical Language Model.
14. **Synta_WER_{fac}**: The syntactical word error rate of the factorizing system.
15. **Synta_WER_{un-fac}**: The syntactical word error rate of the un-factorizing system.
16. **Stat_Morpho_WER_{un-fac}**: The morphological word error rate of the un-factorizing system due to statistical disambiguation.
17. **Stat_Synta_WER_{un-fac}**: The syntactical word error rate of the un-factorizing system due to statistical disambiguation.
18. **TRN_DB_I**: A standard Arabic text corpus with as size $\approx 750K$ words collected from numerous domains over diverse domains. This text corpus is morphologically analyzed and phonetically transcribed. All these kinds of annotations are manually revised and validated.
19. **TRN_DB_II**: A standard Arabic text corpus with as size $\approx 2500K$ words that are only phonetically transcribed in full without any extra annotation.
20. **TST_DB**: A standard Arabic text corpus. It consists of 11K words that are manually annotated for morphology. This test text covers diverse domains.
21. **WER**: Word Error Rate.

22. **WER_h**: The diacritization word error rate of the hybrid diacritizer.
23. **WER_{un-fac}**: The un-factorizing component of the word error rate of the hybrid diacritizer.
24. **WER_{fac}**: The factorizing component of the word error rate of the hybrid diacritizer.
25. **WL-Tagging model**: is an Arabic diacritizer that uses the Full word citation form in the diacritization process and if there exists (OOV) in the words' dictionary, it backs-off to the “L-Tagging model”.

Abstract

The problem of entity factorizing versus unfactorizing is one of the main problems that face peoples who work in the human languages technology (HLT) field. As a case study for this problem; this thesis studies the problem of automatic Arabic text diacritization. The thesis compares the diacritization through words factorization using the morphological analyses versus the diacritization through the words unfactorization using the full-form words. Hence, the thesis introduces a two-layer stochastic system to diacritize raw Arabic text automatically. The first layer determines the most likely diacritics by choosing the sequence of unfactorized full-form Arabic word diacritizations with maximum marginal probability via A^* lattice search algorithm and n-gram probability estimation. When full-form words are out-of-vocabulary (OOV), the system utilizes a second layer, which factorizes each Arabic word into its possible morphological constituents (prefix, root, pattern and suffix), then uses n-gram probability estimation and A^* lattice search algorithm to select among the possible factorizations to get the most likely diacritization sequence. While the second layer has better coverage of possible Arabic forms, the first layer yields better disambiguation results for the same size of training corpora, especially for inferring syntactical (case-based) diacritics. The presented hybrid system possesses the advantages of both layers. After a background on Arabic morphology and part-of-speech tagging, the thesis details the workings of both layers and the architecture of the hybrid system. The experimental results show word error rates of 7.5% for the morphological diacritization and 24.6% for the syntactic diacritization by the second factorizing layers alone, and only 3.6% for the morphological diacritization and 12.7% for the syntactic diacritization by our hybrid system. By comparing our presented system with the other best performing systems - to our knowledge - of Habash and Rambow [14] using their training and testing corpus; it is found that the word error rates of 5.5% for the morphological diacritization and 9.4% for the syntactic diacritization by Habash and

Rambow [14], and only 3.1% for the morphological diacritization and 9.4% for the syntactic diacritization by our system.

From the experimental results we can conclude that; for the small training corpus size the unfactorizing system is better since it can reach to the saturation state faster than the factorizing one but it may suffer from the OOV problem, but for the very large training corpus size; the two systems are almost the same, except that the cost of the unfactorizing systems is lower. So, the best strategy is using a hybrid of the two systems to enjoy the fast learning and the low cost of the factorizing system and the wide coverage of the factorizing one.

Chapter 1

Introduction

The main theme of this thesis is to study the problem of entity factorizing versus unfactorizing, and to answer the question of “is it better to deal with the factorized entities or the unfactorized ones in the HLT area?” So, to answer this question; the problem of Automatic Arabic text diacritization is selected as a case study of the “factorizing versus unfactorizing” problem. This is done by comparing the results of a diacritizer system that is based on factorized entities versus another one that is based on unfactorized entities.

In this introductory chapter; the problem of automatic Arabic text diacritization is defined along with potential applications, a necessary linguistic and historical background is presented next, the background art is then introduced, the challenging points especially those that stimulated innovation are then manifested, afterwards the necessary components for realizing a solution are identified, and the rest of the thesis is finally outlined.

1.1 Automatic Arabic Text Diacritization Problems and Importance

Arabic is one of a class of languages where the intended pronunciation of a written word cannot be completely determined by its standard orthographic representation; rather, a set of special diacritics is needed to indicate the intended pronunciation. Different diacritics over for the same spelling produce different words with maybe different meanings (e.g. عِلْم → “science”, عَلَم → “flag”, عَلَّمَ → “taught”, عَلِمَ → “knew” ... etc.). These diacritics, however, are typically omitted in most genres of written Arabic, resulting in widespread ambiguities in pronunciation and (in some cases) meaning. While native speakers are able to disambiguate the intended meaning and pronunciation from the surrounding context with minimal difficulty, automatic processing of Arabic is often hampered by the lack of diacritics. Text-to-speech (TTS), Part-Of- Speech (POS) tagging, Word Sense Disambiguation (WSD), and Machine Translation can be enumerated among a longer list of applications that vitally benefit from automatic diacritization [6].

The Arabic alphabet consists of 28 letters; 25 letters represent the consonants such as ب (pronounced as /b/) and 3 letters represent the long vowels such as ا (both pronounced as /a:/), ي (pronounced as /i:/), and و (pronounced as /u:/). The Arabic diacritics consist of 8 different shapes and some combinations of them. These diacritics represent short vowels, doubled case endings (Tanween), and syllabification marks.

Table 1.1 below shows the complete set of Arabic diacritics.

Diacritic's type	Diacritic	Example on a letter	Pronunciation
Short vowel	Fatha	بَ	/b//a/
	Kasra	بِ	/b//i/
	Damma	بُ	/b//u/
Doubled case ending (Tanween)	Tanween Fatha	بَا	/b//an/
	Tanween Kasra	بِي	/b//in/
	Tanween Damma	بُو	/b//un/
Syllabification marks	Sukuun	بْ	No vowel: /b/
	Shadda	بّ	Consonant doubling: /b//b/

Table 1.1: Arabic diacritics set.

The diacritics shown in table 1.1 above are the basic set of Arabic diacritics, but another set of shapes may appear as a combination of Shadda-Short vowel pairs such as بّ (pronounced as /b//b//a/), and Shadda-Tanween pairs such as بّ (pronounced as /b//b//un/).

One major challenge with Arabic is its rich derivative and inflective nature, so it is very difficult to build a complete vocabulary that covers all (or even most of) the Arabic generable words [4], [6]. In fact, while Arabic has a very rich vocabulary with regard to full-form words, the resulting data sparseness is much more manageable when parts of words (morphemes) are considered separately, due to Arabic's very systematic and rich morphology [1], [4], [6], [7], [10], [26]. Hence, reliable Arabic morphological analysis is crucial for Arabic text diacritization. Thanks for RDI (www.rdi-eg.com) labs by supporting our experiments, by using their Arabic text diacritization system (*ArabMorpho*[®] ver.4) that factorizes the input Arabic text into all the possible lexemes and case diacritics then statistically disambiguates the most likely sequence of these entities via deep lattice search, hence infers the most likely diacritization and phonetic transcription of the input text [3], [6]. While the virtue of this methodology is its excellent coverage of the language; its drawback is that the search space for the correct diacritics using the factorized word components is much larger than the original search space of full-form words. This larger search space requires larger size of training data, which is expensive and time consuming to build and validate. [6], [26]. Furthermore, this approach requires much processing time due to the large size of the constructed search lattice.

So, we have started to try the same statistical language modeling and disambiguation methodologies over full-form Arabic words instead of factorized ones. While this approach proved to be faster and can produce more accurate diacritization, using a manageable size of training data, it apparently suffers from the problem of poor coverage. It has then been realized that a hybrid of the two approaches may gain the advantages of each of them.

1.2 Linguistic and Historical Background

The pronunciation of a word in some languages like English is always fully determined by its constituting characters. In these languages, the sequence of consonants and vowels determines the correct voice of the word. Such languages are called non diacritized languages [4].

On the other hand, there are languages, like Latin and Arabic, where the sequence of consonants and vowels does not determine the correct voice of the word. In these languages, we can find two or more words have the same spelling but each one of them has a different meaning and different voice. So to remove this ambiguity, special marks are put above or below the spelling characters to determine the correct pronunciation. These marks are called diacritics [4].

The automatic diacritization of Arabic text is turned into an R&D candidate problem due to the simple unfortunate fact that Arabic text is scarcely written with its full diacritical marks.

This fact is rooted into the long history of the Arabic language whose ancient Bedouin native speakers in the Arabic peninsula before Islam relied mainly on oral communication rather than written text. This situation resulted into an early orthographic system which resulted in a highly ambiguous script for even the experienced readers [5], [6], [28], [31], [32].

With the emergence of Islam and the revelation of Qur'aan, Muslims had to guard their holy book against all kinds of misconstruction. So they developed the Arabic orthography to extend and disambiguate its basic graphemes set via dotting and at a later stage adding extensive diacritical marks that clarify accurately its phonetic transcription [5], [6], [28], [31], [32].

Even for the skilled writers it was too slow to deliver at the realistic rates needed for official governmental documentation, especially during the grand empires like those of

Abbasids and Ottomans, who developed slimmer versions with simpler fonts, where spelling only is scripted in order to minimize the number of needed strokes and hence speed up the writing process [5], [6], [28], [31], [32].

1.3 Survey

The actually implemented systems of computational processing of Arabic are mostly Arabic diacritizer processors. These systems can be divided into two categories:

1. Systems implemented by individuals as part of their academic activities.
2. Systems implemented by commercial organizations for realizing market applications.

The power point of the systems of the first category was that they presented some good ideas as well as some formalization. The weak point was that these systems were mostly partial demo systems. They gave a higher priority to demonstrating the new ideas over producing complete mature engines capable of dealing automatically with real-life Arabic text [4].

On the other hand, driven by the market need for applications of Arabic morphological processing, the systems of the second category enhanced the theoretical arguments presented by the ones of the first category to produce usable products [4].

From the first category, we review four approaches that are directly relevant to us:

- a.** It is found that the state-of-the-art is for two systems that are produced by two academic groups; Zitouni et al. [27] and Habash and Rambow [14]. A complete discussion about these two systems and a comparison between them and the presented system in this thesis are found in chapter 5 below.
- b.** Vergyri and Kirchhoff (2004), they choose from the diacritizations proposed by the Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter, 2004). However, they train a single tagger using unannotated data and expectation maximization (EM), which necessarily leads to a lower performance. They were motivated by the goal of improving automatic speech recognition (ASR), and have an acoustic signal parallel to the undiacritized text. All their experiments use acoustic models. They show that word error rate (WER) for diacritization decreases by nearly 50% (from 50%) when BAMA is added to the acoustic information [14].
- c.** Ananthakrishnan et al. (2005) also work on diacritization with the goal of improving ASR. They use a word-based language model (using both diacritized

and undiacritized words in the context) but back-off to a character-based model for unseen words. They consult BAMA to narrow possible diacritizations for unseen words, but BAMA does not provide much improvement used in this manner.

From the second category, the most representative commercial Arabic morphological processors are Sakhr's, Xerox's, and RDI's [4].

- a.** Sakhr's: it is an Arabic diacritizer was achieved by native Arabic speakers. Nevertheless, it suffers from some shortcomings.
 - i. Although this system is a factorizing system (i.e. it should be there is no coverage problem), but it was based on the standard Arabic dictionaries (i.e. the morphologically possible Arabic words that are not registered in these dictionaries are not considered). The problem of this restriction is that even the most elongated Arabic dictionaries do not list all the used Arabic vocabulary at its time. Moreover, the language is a dynamic phenomenon, i.e. an unused possible word at some time may be indispensable at later time. Also, an unused word at some Arabic country may be famous at another Arabic country [4].
 - ii. The actual implementation of the statistical disambiguation at Sakhr is made by considering only the monograms of words (the frequency of single words) in the text corpus and does not count for the correlation among neighboring words. Considering correlation makes statistical correlation far more effective than overlooking it [4].
- b.** Xerox's: it is an Arabic diacritizer, it was the best system implemented by non-native Arabic speakers. Also, it suffers from the following shortcomings.
 - i. Although this system is a factorizing system (i.e. it should be there is no coverage problem), but it is based on the standard Arabic dictionaries (i.e. the morphologically possible Arabic words that are not registered in these dictionaries are not considered). This is the same corresponding shortcoming of Sakhr's system mentioned above [4].
 - ii. Xerox system has no mechanism of disambiguation [4].
 - iii. This system is made by non-native Arabic speakers. Moreover, they also selected dictionaries and references on the classical Arabic morphologically written by non-native Arabic speakers. Some concept as well as many fine points are misunderstood or simply overlooked. Also, a significant portion

of morphological entities (root, forms, prefixes, or suffixes) are absent or mistaken. So, the coverage of the final system is not excellent, especially when the system is tested against a literature-oriented or an old Arabic text [4].

- c. RDI's: it is a large scale Arabic diacritizer achieved by native Arabic speakers. It has the following advantages over the above competing systems:
 - i. It is a factorizing system; each regular derivative root is allowed to combine with any form as long as this combination is morphologically allowed. This allows dealing with all the possible Arabic words and removes the need to be tied to a fixed vocabulary [4], [6].
 - ii. This system uses a powerful n-grams statistical disambiguation technique. This means that the system considers the statistical correlation among the words and their neighbors [4], [6].

Although this system solved the shortcoming of the above two systems, but it suffers from the following shortcomings:

- i. This system is a factorizing system; its drawback is its relatively sluggish attenuation of the disambiguation error margin with increasing the annotated training corpora which are expensive and time consuming to build and validate [6], [26].

The existence of all the morphologically allowed combinations for a certain word led to a large amount of possibilities for that word, which in turn led to a higher processing time for the disambiguation process; also it increased the difficulty of disambiguation process.

1.4 Challenges and Points of Innovation

1.4.1 Challenges

Due to the following challenges; the task of building a reliable Arabic diacritizer is a hard one:

- 1- Arabic text is typically written without any diacritics [4], [5], [6], [11], [28], [32].
- 2- Arabic text is commonly written with many common spelling mistakes (أ-ا), (ة-ه), (ى-ي) [4], [6].

- 3- Due to the highly derivative and inflective nature of Arabic, it is very difficult to build a complete vocabulary that covers all (or even most of) the Arabic generable words [4], [6].
- 4- While the virtue of morphological analyzer to solve the problem of coverage instead of using dictionary, its drawback is its relatively sluggish attenuation of the disambiguation error margin with increasing the annotated training corpora which are expensive and time consuming to build and validate [6], [26].

About two thirds of Arabic text words have a syntactically dependent case-ending which invokes the need to a syntax analyzer which is a hard problem [4], [6].

1.4.2 Innovation Points in Our Work

- 1- It is a hybrid system of the unfactorizing system (i.e. it is a dictionary based systems) and the factorizing system (i.e. it depends on morphological analyzer). This achieves the advantages of the two systems (speed and accuracy from the unfactorizing system and the excellent coverage of the language from the factorizing ones).
- 2- The training cost is lower than the factorizing systems; since it depends mainly for training on diacritized data, which are available for free most of the times or at least with a low cost. This is better than the factorizing systems that depend mainly on a fully manually annotated training data (Part-Of-Speech tagged and morphologically analyzed data) which is very costly.

1.5 The Presented System

Aiming to enhance the performance of the Arabic diacritizer of factorized Arabic text; we developed a hybrid system that combines the morphology based diacritizer with another diacritizer that is based on full-form words. Figure 4.2 in chapter 4 shows the architecture of this hybrid Arabic diacritizer.

A large Arabic text corpus with a revised full (morphological & Syntactic) phonetic annotation is used to build a dictionary of full-form Arabic words vocabulary. In the offline phase also, this text corpus is indexed and used to build a statistical language model of full-word n-grams. In the runtime; each word in the input Arabic text is

searched for in this dictionary by the “Word Analyzer and Segmentor” module. If the word is found, the word is called “analyzable” and all its diacritization occurrences are retrieved from the dictionary. A consequent series of analyzable words in the input text is called “analyzable segment”. All the diacritization occurrences of the words in an analyzable segment constitute a lattice, as shown by figure 1.1 below, that is disambiguated via n-grams probability estimation and A^* lattice search to infer the most likely sequence of diacritizations [2], [16], [21].

The diacritized full-form words of the disambiguated analyzable segments are concatenated to the input words in the un-analyzable segments (if ever) to form a less ambiguous sequence of Arabic text words. The latter sequence is then handled by the “Factorizing Disambiguator” that is illustrated in chapter 2 below.

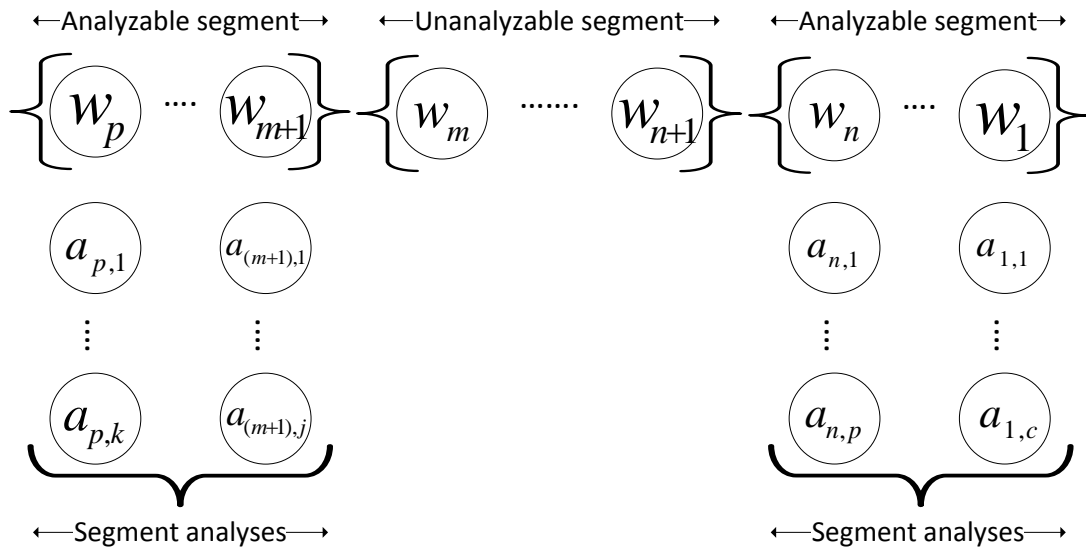


Figure 1.1: “Un-analyzable Segments” in input text.

1.6 Thesis Outline

After this introductory chapter; the factorizing system that is used as a back-off system in this thesis (*ArabMorpho*[®] ver.4) is discussed in some details in chapter 2.

The mathematical foundations of statistical language modeling as well as optimal trellis search as a basis for statistical disambiguation are established in chapter 3.

Chapter 4 introduces the hybrid system giving a clear illustration to the un-factorizing system and all its components, and how it is connected to the factorizing system in chapter 2 to create the hybrid system.

In chapter 5 a detailed discussion for the experiments and the results for the hybrid system with some comparisons with the factorizing system are mentioned, also a

comparison with recent related work is held. The last chapter presents overall conclusions, contemplations, and suggestions for future work building on what has been done throughout this thesis.

Chapter 2

Background On The Arabic Factorization Model Used In Our System

The diacritization of an Arabic word consists of two components; morphology-dependent and syntax-dependent ones. While the morphological diacritization distinguishes different words with the same spelling from one another; e.g. عِلْمٌ which means “science” and عَلَمٌ which means “flag”, the syntactic case of the word within a given sentence; i.e. its role in the parsing tree of that sentence, determine the syntax-dependent diacritic of the word. For example; الرياضياتِ عِلْمٌ درُسْتُ implies the syntactic diacritic of the target word - which is an “object” in the parsing tree - is “Fatha”, while يفيدُ عِلْمٌ الرياضياتِ جميعَ العلومِ implies the syntactic diacritic of the target word – which is a “subject” in the parsing tree - is “Damma”.

In this chapter, the factorizing part of our presented system - RDI (*ArabMorpho*[®] ver.4) - is introduced and the problems of diacritizing the morphology-dependent parts of the word and syntax-dependent ones according to its point of view are then discussed. At the end of this chapter, we discuss how *ArabMorpho*[®] ver.4 is used in the presented hybrid system.

2.1 Arabic Morphological Analysis

RDI’s Arabic morphological model assumes the canonical structure uniquely representing any given Arabic word w to be a quadruple of lexemes (or morphemes) so that $w \rightarrow q = (t: p, r, f, s)$ where p is prefix code, r is root code, f is pattern (or form) code, and s is suffix code. The type code t can signify words belonging to one of the following 4 classes: *Regular Derivative* (w_{rd}), *Irregular Derivative* (w_{id}), *Fixed* (w_f), or *Arabized* (w_a) [4], [6].

Prefixes and suffixes; P and S , the 4 classes applied on patterns giving F_{rd} , F_{id} , F_f , and F_a , plus only 3 classes applied on roots¹; R_d , R_f , and R_a constitute together the 9 categories of lexemes in this model. As shown in figure 2.1 below.

¹ The roots are common among both the regular and irregular derivative Arabic words.

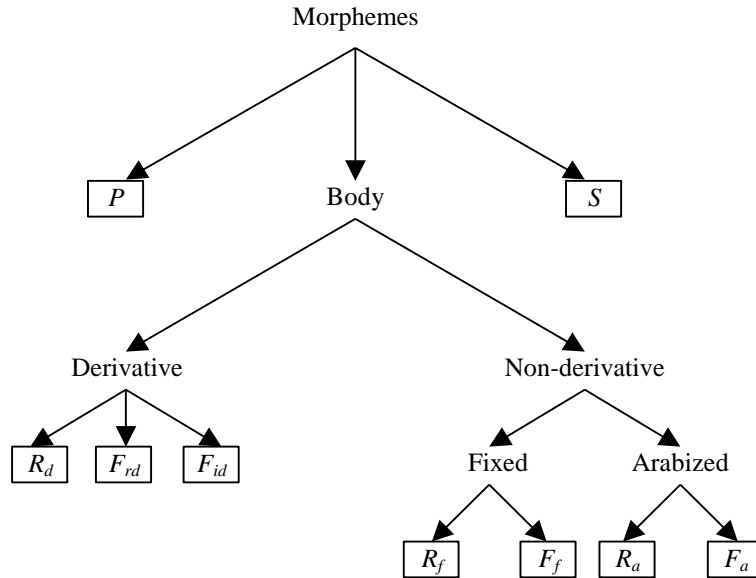


Figure 2.1: Classifying the 9 types of morphemes in the Arabic lexicon of *ArabMorpho*[®] ver.4. (With courtesy to Attia 2000 [4]).

The total number of lexemes of all these categories in this model is around 7,800. With such a limited set of lexemes, the dynamic coverage exceeds 99.8% measured on large Arabic text corpora excluding transliterated words [4]. The sizes of each kind of morphemes in figure 2.1 are as follows:

- 1- P : About 260 Arabic prefixes.
- 2- R_d : About 4,600 Arabic derivative roots.
- 3- F_{rd} : About 1,000 Arabic regular derivative patterns.
- 4- F_{id} : About 300 Arabic irregularly derived words.
- 5- R_f : About 250 Roots of Arabic fixed words.
- 6- F_f : About 300 Arabic fixed words.
- 7- R_a : About 240 Roots of Arabized words.
- 8- F_a : About 290 Arabized words.
- 9- S : About 550 Arabic suffixes.

While table 2.1 below shows this model applied on few representative sample Arabic words, the reader is kindly referred to [4] for the detailed documentation of this Arabic morphological factorization model and its underlying lexicon along with the dynamics of the involved analysis/synthesis algorithms.

<i>Sample word</i>	<i>Word type</i>	<i>Prefix & prefix code</i>	<i>Root & root code</i>	<i>Pattern & pattern code</i>	<i>Suffix & suffix code</i>
فَمَا	<i>Fixed</i>	فَـ 2	الَّذِي 87	مَا 48	— 0
تَتَنَاوَلَهُ	<i>Regular Derivative</i>	تَـ 86	ن و ل 4077	تَفَاعَلَ 176	هـ 8
الْكِتَابَاتِ	<i>Regular Derivative</i>	الـ 9	ك ت ب 3354	فِعَال 684	ات 27
الْعِلْمِيَّةِ	<i>Regular Derivative</i>	الـ 9	ع ل م 2754	فِعْل 842	يَّة 28
مِنْ	<i>Fixed</i>	— 0	مِنْ 63	مِنْ 118	— 0
مَوَاضِعِ	<i>Regular Derivative</i>	— 0	و ض ع 4339	مَفَاعِيل 93	— 0
مُتَّخِذَةً	<i>Irregular Derivative</i>	— 0	أ خ ذ 39	مُتَّخِذَ 13	ة 26

Table 2.1: Exemplar Arabic morphological analyses.
(With courtesy to Attia 2005 [6]).

2.2 Arabic POS Tagging

RDI's Arabic POS-tagging model relies on a compact set of Arabic POS tags containing only 62 tags that cover all the possible atomic context-free syntactic features of Arabic words. While many of these Arabic POS tags may have corresponding ones in other languages, few do not have such counterparts and may be specific to the Arabic language [3], [6].

This POS tags-set has been extracted after a thorough scanning and redundancy elimination of the morpho-syntactic features of the 7,800 lexemes in this morphologically factorized Arabic lexicon. Completeness, atomicity, and insurability of these scanned morpho-syntactic features were the criteria adhered to during that process. Table 2.2 displayed below shows RDI Arabic POS tags set along with the meaning of each tag verbalized in both English and Arabic [3], [6].

Cat.	Mnemonic	Meaning in English	Meaning in Arabic
Start of word marker	SOW	Start-Of-Word marker	بداية كلمة
	Padding string	Padding string	حشو
Features of noun and verb prefixes	NullPrefix	Null prefix	لا سابق
	Conj	Conjunctive	عطف
	Confirm	Confirmation by <i>Laam</i>	لام التوكيد
	Interrog	Interrogation by <i>Hamza</i>	همزة الاستفهام
Features of noun and verb suffixes	NullSuffix	Null suffix	لا لاحق
	ObjPossPro	Object or possession pronoun	ضمير نصب أو جر
Verb and noun syntactic cases	MARF	1 st Arabic syntactic case	مرفوع
	MANSS	2 nd Arabic syntactic case	منصوب
Features of noun-only prefixes	Definit	Definitive article	"ال" التعريف
Features of noun-only stems	Noun	Nominal	اسم
	NounInfinit	Nouns made of infinitives	مصدر
	NounInfinitLike	"NounInfinit" like	اسم مصدر
	SubjNoun	Subject noun	اسم فاعل
	ExaggAdj	Exaggeration adjective	صيغة مبالغة
	ObjNoun	Object noun	اسم مفعول
	TimeLocNoun	Noun of time or location	اسم زمان أو مكان
	NoSARF	An Arabic feature of a specific class of nouns	منوع من الصرف
Features of noun-only suffixes	PossessPro	Possessive pronoun	ضمير جر
	RelAdj	Relative adjectives maker	نسب
	Femin	Feminine	تانيث
	Masc	Masculine	مذكر
	Single	Singular	مفرد
	Binary	Binary	مثنى
	Plural	Plural	جمع
	Adjunct	Adjunct	مضاف
	NonAdjunct	NonAdjunct	غير مضاف
	MANSS_MAGR	2 nd or 3 rd Arabic syntactic case	منصوب أو مجرور
MAGR	3 rd Arabic syntactic case	مجرور	
Features of verb-only prefixes	Present	Present tense	مضارع
	Future	Future tense	استقبال
Features of verb-only stems	Active	Active sound	مبني للمعلوم (للفاعل)
	Passive	Passive sound	مبني للمجهول (للمفعول)
	Imperative	Imperative	أمر
	Verb	Verb	فعل
	Transitive	Transitive verb	لازم
	MAJZ	4th Arabic syntactic case	مجزوم
	Past	Past tense	ماضي
	PresImperat	Present tense, or imperative	مضارع أو أمر
Features of verb-only suffixes	SubjPro	Subject form pronoun	ضمير رفع
	ObjPro	Object form pronoun	ضمير نصب
	MANS_MAJZ	2nd or 4th Arabic syntactic case	منصوب أو مجزوم
Features of: mostly functional fixed words, and scarcely affixes	Prepos	Preposition	حرف جر
	Interj	Interjection	حرف نداء
	PrepPronComp	Preposition-Pronoun Compound	جار ومجرور
	RelPro	Relative pronoun	اسم موصول
	DemoPro	Demonstrative pronoun	اسم إشارة
	InterrogArticle	Interrogation article	أداة استفهام
	JAAZIMA	For specific articles that make the consequent verb in the 4th Arabic syntactic case	جازمة
	CondJAAZIMA	Feature of a class of Arabic conditionals	شرطية جازمة
	CondNotJAAZIMA	Feature of a class of Arabic conditionals	شرطية غير جازمة
	LAA	Arabic specific article	لا
	LAATA	Arabic specific article	لات
	Except	Article of exception	استثناء
	NoSyntaEffect	A class of articles that have no syntactic effect	غير عاملة
	DZARF	Feature for certain kind of Arabic adverbs	ظرف
ParticleNAASKH	A class of particles that make the subject of the consequent nominal sentence in 2nd Arabic syntactic case	حرف ناسخ	
VerbNAASIKH	A class of auxiliary verbs that make the predicate of the consequent verbal sentence in 2nd Arabic syntactic case	فعل ناسخ	
ParticleNAASSIB	Arabic specific class of particles that make the consequent verb in 2nd Arabic syntactic case	ناصب	
MASSDARIYYA	Arabic specific article	مصدرية	
For words beyond our morphological model	Translit	Transliterated Arabic string	كلمة أجنبية مكتوبة بحروف عربية

Table 2.2: Arabic POS tags set. (With courtesy to Attia 2005 [6]).

Due to the atomicity of these Arabic POS-tags as well as the compound nature of Arabic lexemes in general, the POS labels of Arabic lexemes are represented by POS tags-vectors. Each lexeme in this Arabic factorized lexicon is hence labeled by a POS tags-vector as exemplified by table 2.3 below.

While the Arabic POS-tagging of stems is retrieved from the POS label of the pattern lexeme only, not the root's, the POS-tagging of the affixes is obtained from the POS labels of the prefix and suffix. So, the Arabic POS-tagging of a quadruple corresponding to a morphologically factorized input Arabic word is given by the concatenation of its POS labels of the prefix, the pattern, and suffix respectively after eliminating any redundancy [3], [6].

While table 2.4 shows the Arabic POS-tagging of few sample words, the reader is kindly referred to [3] and chapter 3 of [6] for the detailed documentation of this Arabic POS-tagging model along with its underlying POS tags-set.

<i>Lexeme</i>	<i>Type & Code</i>	<i>Arabic POS tags vector label</i>
الـ	P 9	[Definitive] [ال التعريف]
سيـ	P 125	[Future, Present, Active] [استقبال، مضارع، ميني للمعلوم]
مُفَاعِلِ	F _{rd} 482	[Noun, Subjective Noun] [اسم، اسم فاعل]
اسْتِفْعَالِ	F _{rd} 67	[Noun, Noun Infinitive] [اسم، مصدر]
مَلَائِكِ	F _{id} 29	[Noun, No SARF, Plural] [اسم، ممنوع من الصرف، جمع]
هُوَ	F _f 8	[Noun, Masculine, Single, Subjective Pronoun] [اسم، مذكر، مفرد، ضمير رفع]
ذُو	F _f 39	[Noun, Masculine, Single, Adjunct, MARFOU'] [اسم، مذكر، مفرد، مضاف، مرفوع]
اتـ	S 27	[Feminine, Plural] [مؤنث، جمع]
وَنَهُمْ	S 427	[Present, MARFOU', Subjective Pronoun, Objective Pronoun] [مضارع، مرفوع، ضمير رفع، ضمير نصب]
سَيِّتَانِ	S 195	[Relative Adjective, Feminine, Binary, Non Adjunct, MARFOU'] [نسب، مؤنث، مثنى، غير مضاف، مرفوع]

Table 2.3: POS labels of sample Arabic lexemes.
(With courtesy to Attia 2005 [6]).

Sample word	Arabic POS tags vector
فَمَا	[Conjunction, Noun, Relative Pronoun, Null Suffix] [عطف، اسم، اسم موصول، لا لاحقة]
تَتَنَاوَلُهُ	[Present, Active, Verb, Objective Pronoun] [مضارع، مبني للمعلوم، فعل، ضمير نصب]
الْكِتَابَاتِ	[Definitive, Noun, Plural, Feminine] [ال التعريف، اسم، جمع، مؤنث]
الْعِلْمِيَّةِ	[Definitive, Noun, Relative Adjective, Feminine, Single] [ال التعريف، اسم، نسب، مؤنث، مفرد]
مِنْ	[Null Prefix, Preposition, Null Suffix] [لا سابقة، حرف، لا لاحقة]
مَوَاضِيَعِ	[Null Prefix, Noun, No SARF, Plural, Null Suffix] [لا سابقة، اسم، ممنوع من الصرف، جمع، لا لاحقة]
مَتَّخِذَةً	[Null Prefix, Noun, Objective Noun, Feminine, Single] [لا سابقة، اسم، اسم مفعول، مؤنث، مفرد]

Table 2.4: POS tags-vectors of sample Arabic words.
(With courtesy to Attia 2005 [6]).

2.3 Morphological and Syntactical Diacritization According to ArabMorpho[©] ver.4 View point

The morphological diacritization of a given word is directly extractable from the prefix, pattern, and suffix lexemes of its morphological analysis of that word. The issue here is to disambiguate the multiple analyses proposed by the Arabic morphological analyzer. In the absence of deeper linguistic processing, statistical disambiguation is deployed to infer the sequence of analyses from the Arabic morphological disambiguation trellis shown in figure 2.2 below with maximum likelihood probability according to a statistical language model built from a morphologically annotated training corpus [3], [6].

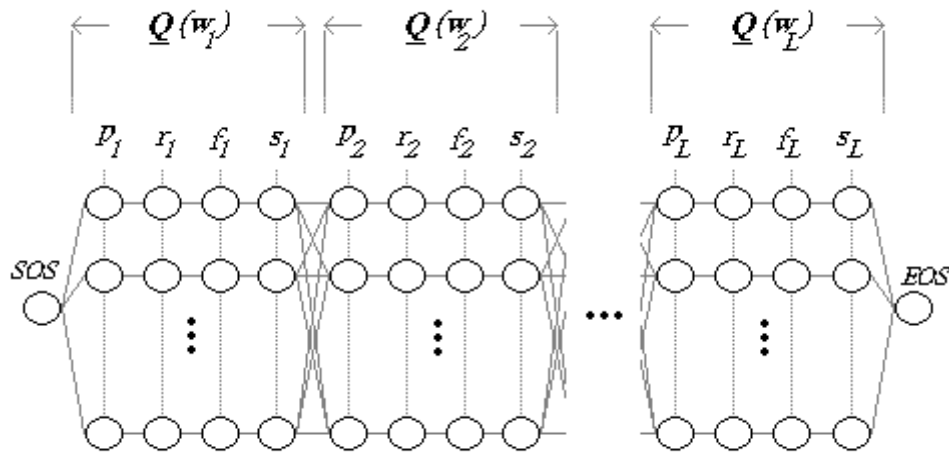


Figure 2.2: The Arabic lexical disambiguation trellis.

(With courtesy to Attia 2005 [6]).

For syntactic diacritization the POS-tags vectors of a sequence of Arabic words along with the possible syntactic diacritics of each are obtained after its morphological disambiguation. Statistical disambiguation is deployed again to infer the sequence of syntactic diacritics & POS tags from the Arabic POS tags and possible syntactic diacritics disambiguation trellis shown in figure 2.3 below with maximum likelihood probability according to a statistical language model built from a training corpus annotated with POS tags & syntactic diacritics [3].

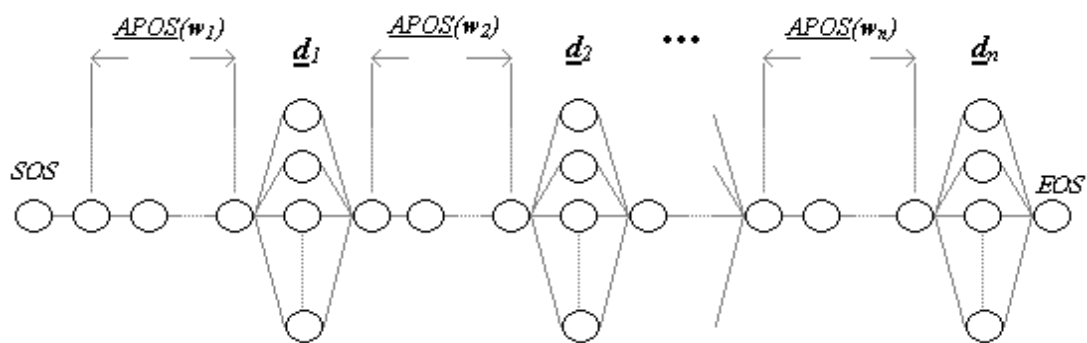


Figure 2.3: The Arabic POS Tags–Syntactic Diacritics search trellis.
(With courtesy to Attia 2005 [6]).

2.4 Morphological and Syntactical Diacritization According to our View point using *ArabMorpho*[®] ver.4

Actually, the idea of making the problems of morphological diacritization and syntactical diacritization two different problems as shown in figure 2.2 and figure 2.3 above is a smart idea. Since dealing with them as a one problem as shown in figure 2.4 below produces a huge search dimensionality, i.e. this idea decreases the size of the search space. So, the search complexity becomes better, which results in faster search time and higher output accuracy.

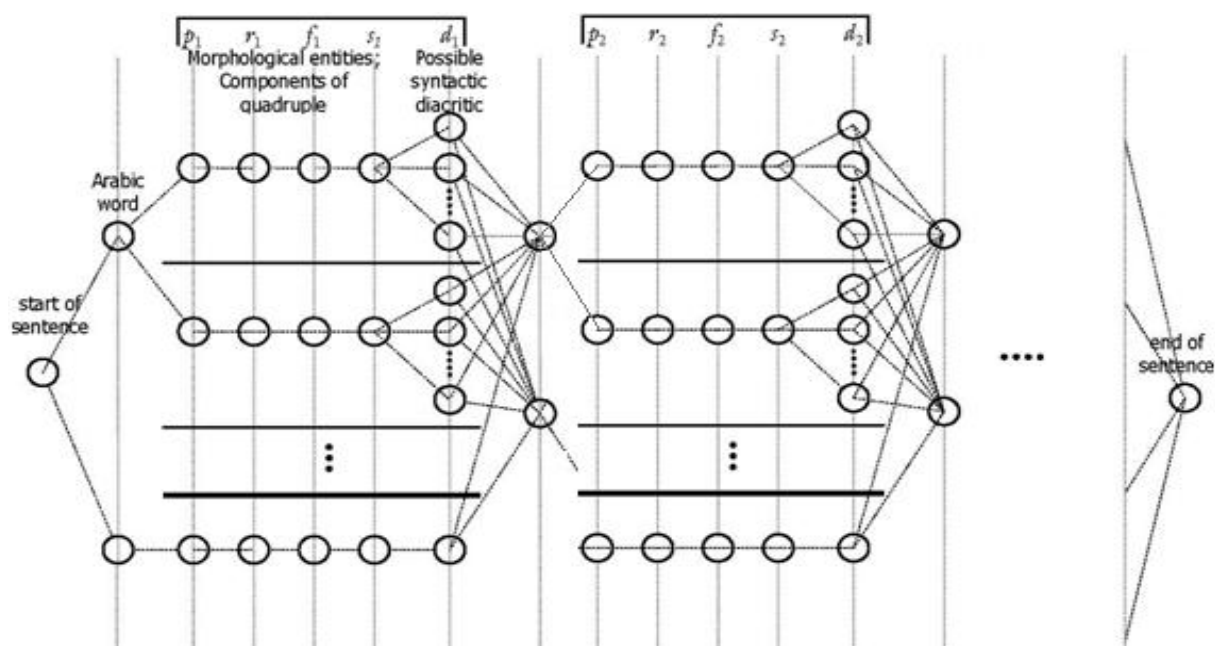


Figure 2.4: Disambiguation lattice for morphological disambiguation, syntactic diacritization. (With courtesy to Attia, Rashwan, Khallaaf 2002 [2]).

So, we used the same morphological diacritization method of *ArabMorpho*[®] ver.4 that depends on the morphological analyses. But for the syntactical diacritization problem, we wanted to use a syntax analyzer, but due to the weakness of the syntax analyzers of Arabic; we had to use the statistical methods.

The statistical method that is used by *ArabMorpho*[®] ver.4 depends on the POS tags of the words. Actually, the POS tags are necessary for syntax analyses process, and they are sufficient for the context-free grammar languages (CFG) [19] like any programming language (ex. C, C++, V. Basic ...), but these POS tags not sufficient for syntax analyses process for the context-sensitive languages [19] like human languages (ex. Arabic ...), since in human languages like Arabic the word sense is very important for the syntax analyses process.

According to the above concepts; the POS-tagging is necessary for syntactical diacritization process, but not sufficient, since it does not carry any information about the senses which are important for the syntactic diacritization process. So, using the morphological analyses of the word in the syntactical diacritization process is important since it contain the sense information inside it.

To illustrate that, consider the two highlighted words in the following two phrases "أكل الأسد الطفل" and "أكل البلح الطفل"; the two words "الأسد" and "البلح" have the same prefix, the same suffix, and the same form, so they have the same POS tags according to *ArabMorpho*[©] ver.4 methodology [3], [6]. However, they have different morphological analyses (i.e. different semantics). So, while the word "الأسد" in the first phrase is a "subject" and its syntactic diacritic is "Damma", but the word "البلح" in the second phrase is an "object" and its syntactic diacritic is "Fatha". But according to the statistical method that is used by *ArabMorpho*[©] ver.4 [3], [6] - that depends on the POS tags of the word to determine the syntactic diacritic of the word – the two words will have the same statistical probabilities; that will produce the same syntactic diacritic for the two words, which is wrong.

Since for any morphological analyses there are unique POS tags as discussed in section 2.2 above. So, this means that the information of the POS tags is already impeded inside the morphological analyses of the word, while the information of the POS tags does not carry the information of the morphological analyses of the word, or by other ways it does not carry the word sense. So, depending on the morphological analyses in the syntactical diacritization process instead of POS-tagging is enough.

According to the above discussion, it has got clear that the word sense plays a big role in the syntactical diacritization process, also Arabic is a natural language does not obey a fixed phrase structure (ex. there is no rule to force the subject to appear before the object, but at any time any one of them can appear before the other. And only the semantic is the key to understand the phrase). So, depending only on the POS tags in the syntactical diacritization problem is not the best choice. But the best strategy is to marry the semantic information with the syntactical information while working in the syntactical diacritization problem. Since we do not have a powerful semantic analyzer, we decided to use the morphological analyses of the word that is already confirmed from the morphological diacritization process instead of the POS tags alone.

So, Statistical disambiguation is deployed again to infer the sequence of syntactic diacritics and morphological analyses from the morphological analyses and possible syntactic diacritics disambiguation trellis shown in figure 2.5 below with maximum likelihood probability according to a statistical language model built from a training corpus annotated with morphological analyses and syntactic diacritics.

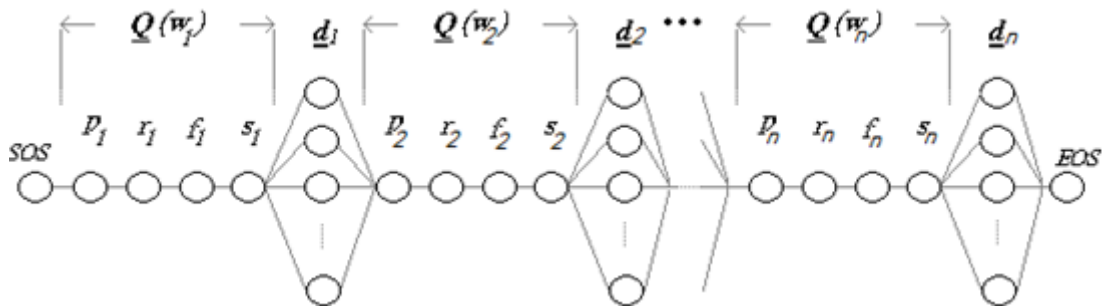


Figure 2.5: The Arabic morphological analyses–Syntactic Diacritics search trellis

While the deployed statistical disambiguation and language modeling [2], [6] are reviewed on chapter 3 of this thesis, the aforementioned scheme of this diacritization is illustrated by figure 2.6 below. The performance of this architecture is analyzed on chapter 5 of this thesis.

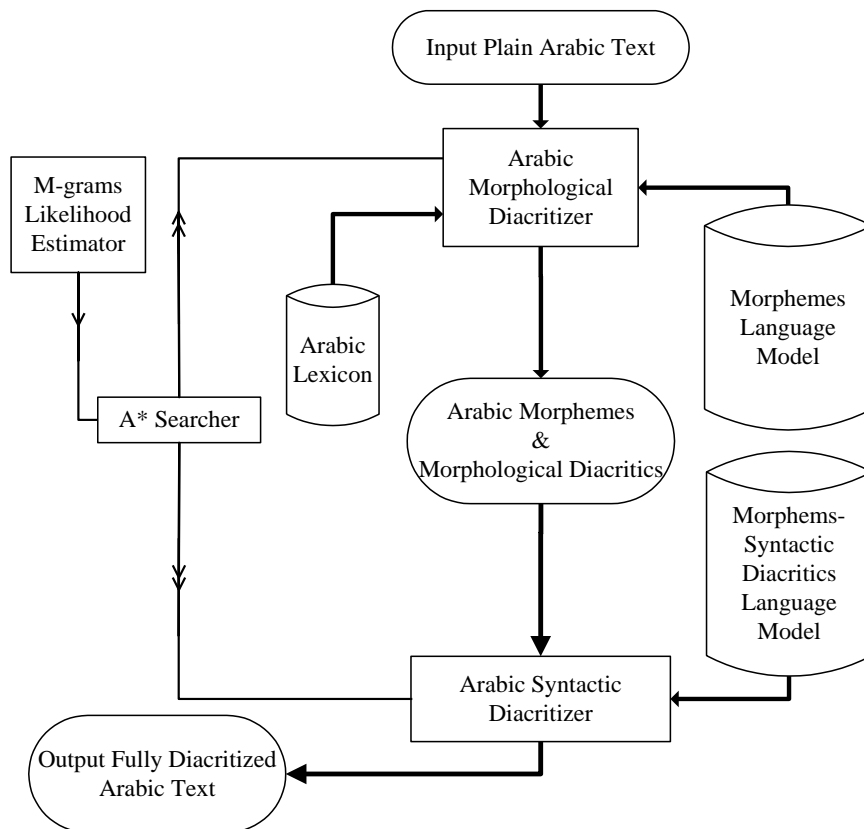


Figure 2.6: The architecture of Arabic diacritizer statistically disambiguating factorized Arabic text.

Chapter 3

**Statistical Disambiguation
Methods**

In both architectures presented in chapter 2 above and chapter 4 below, the challenging ambiguity of multiple possible solutions at each word of the input text lead to the composition of a trellis abstracted in figure 3.1 below. To resolve this ambiguity and infer the most statistically sound sequence of solutions $\hat{\underline{I}}$, we rely on the well established approach of maximum a posteriori (MAP) probability estimation [2], [6], [15], [21], [23].

w_1	w_2	...	w_L
$a_{1,1} \bullet$	$a_{2,1} \bullet$...	$a_{L,1} \bullet$
$a_{1,2} \bullet$	$a_{2,2} \bullet$...	$a_{L,2} \bullet$
\vdots	\vdots	...	\vdots
$a_{1,J_1} \bullet$	$a_{2,J_2} \bullet$...	$a_{L,J_L} \bullet$
\underline{a}_1	\underline{a}_2	...	\underline{a}_L

Figure 3.1: The ambiguity of multiple solutions of each word in the input text \underline{W} leading to a solution trellis of possible analyses ($\underline{a}_1 \times \underline{a}_2 \times \dots \times \underline{a}_L$). (With courtesy to Attia 2005 [6]).

3.1 Maximum a Posteriori Probability Estimation

The maximum a posteriori probability (MAP) estimation [2], [6], [15], [21], [23] famously formulated by:

$$\hat{\underline{I}} = \arg \max_{\forall \underline{I}} \{P(\underline{I} | \underline{O})\} = \arg \max_{\forall \underline{I}} \left\{ \frac{P(\underline{O} | \underline{I}) \cdot P(\underline{I})}{P(\underline{O})} \right\} = \arg \max_{\forall \underline{I}} \{P(\underline{O} | \underline{I}) \cdot P(\underline{I})\} \dots \dots \text{Eq (3.1)}$$

Where (\underline{O}) is the output observations and (\underline{I}) is the input observations.

In other pattern recognition problems like optical character recognition (OCR) and automatic speech recognition (ASR), the term $P(\underline{O} | \underline{I})$ referred to as the likelihood probability, is modeled via probability distributions; e.g. Hidden Markov Model (HMM) in ASR.

Our aforementioned language factorization models and/or dictionary retrieval enable us to do better by viewing the formal available structures, in terms of probabilities, as a binary decision; i.e. a decision of whether the observation obeys the formal rules or not. This simplifies MAP formula above into:

$$\hat{\underline{I}} = \arg \max_{\forall \underline{I} \in \mathfrak{R}} \{P(\underline{I})\} \dots \dots \text{Eq (3.2)}$$

Where \mathfrak{R} is the space of factorization model or dictionary, and $P(\underline{I})$ is the independent probability of the input which is called the statistical language model (SLM). The term $P(\underline{I})$ then expresses the n-grams probability estimated according to the distributions computed from the training corpus.

Using the chain rule for decomposing marginal into conditional probabilities, the term $P(I)$ may be approximated by:

$$P(\underline{Q}) \cong \prod_{i=1}^L P(a_i | a_{i-N}^{i-1}) \dots \dots \text{Eq (3.3)}$$

Where N is the maximum affordable n-gram length in the SLM.

These conditional probabilities are simply calculated via the famous *Bayesian* formula. However, the severe *Zipfian* sparseness [23] of n-grams of whatever natural language entities necessitates more elaboration. So, the *Good-Turing* discount and *back-off* techniques are also deployed to obtain reliable estimations of rarely or never occurring events respectively [2], [6], [15], [16], [23]. These techniques are used for both building the discrete distributions of linguistic entities from labeled corpora, and also for estimating the probabilities of any given n-gram of these entities in the runtime.

3.2 Probability Estimation Via Smoothing Techniques

Any entity in the language vocabulary must have usage in some context; but it seems impossible to cover all entities in a certain training set. The process of biasing the uncovered set on the expense of discounting the other covered set is called smoothing. If there is no smoothing technique is used with the disambiguation system it would refuse the correct solution if any of its input entities was unattested in the training corpus.

One of the effective techniques widely adopted today, namely Back-Off¹, is briefed below¹.

¹ This brief is quoted without modification from [4]. In this thesis the “Bayes’, Good-Turing Discount, Back-Off” technique is further explained. (With courtesy to Attia 2000 [4]).

The off-line phase; building the statistical knowledge base:

1. Build the m -grams w_1^m and their counts $c(w_1^m)$ for all $1 \leq m \leq M$.
Where M is the maximum size of the m -grams.
2. Get the counts $n_{r,m}$ for all $1 \leq r \leq k_1 + 1$; $1 \leq m \leq M$.
 k_1 is a constant with the typical value $k_1=5$.
 $n_{r,m}$ =number of m -grams that occurred exactly r times.
3. Compute

$$\delta_m = \frac{(k_1 + 1) \cdot n_{k_1+1,m}}{n_{1,m}}; 1 \leq m \leq M. \quad \dots\dots\text{Eq (3.4)}$$
4. Sort all the m -grams in ascending order using the quick-sort algorithm for later fast access using binary search.
5. Compute the α parameters as:

$$\alpha(w_1^{m-1}) = \frac{1 - \sum_{w_m: c(w_1^m) > 0} P(w_m | w_1^{m-1})}{1 - \sum_{w_m: c(w_1^m) > 0} P(w_m | w_2^{m-1})}; m > 1 \quad \dots\dots\text{Eq (3.5)}$$
6. Discard w_1^m and the corresponding $\alpha(w_1^{m-1})$ for which $c(w_1^m) \leq k_2$.
 k_2 is a constant with the typical values $k_2=1$.

The run-time phase; estimating the m -gram conditional probabilities:

if $c(w_1^m) > k_1$,

Apply Bayes' rule:

$$\left\{ \begin{array}{l} \text{if } m > 1, P(w_m | w_1^{m-1}) = \frac{\alpha(w_1^m)}{\alpha(w_1^{m-1})} \\ \text{if } m = 1, P(w_m | w_1^{m-1}) = \frac{\alpha(w_1^1)}{N}; \\ N = \text{Total number of the occurrences of monograms} \end{array} \right. \quad \dots\dots\text{Eq (3.6)}$$

if $(m = 1 \text{ AND } c(w_1^m) \leq k_1) \text{ OR } (m > 1 \text{ AND } k_2 < c(w_1^m) \leq k_1)$,

Apply Good- Turing discount:

$$\left\{ \begin{array}{l} c^*(w_1^m) = \frac{(c(w_1^m) + 1) \cdot n_{c(w_1^m)+1,m}}{n_{c(w_1^m),m}} \\ d_{c(w_1^m)} = \frac{\frac{c^*(w_1^m)}{c(w_1^m)} - \delta_m}{1 - \delta_m} \\ \text{if } m > 1, P(w_m | w_1^{m-1}) = d_{c(w_1^m)} \cdot \frac{c(w_1^m)}{c(w_1^{m-1})} \\ \text{if } m = 1 \\ \left\{ \begin{array}{l} \text{if } (c(w_1^1) \geq k_2), P(w_m | w_1^{m-1}) = d_{c(w_1^1)} \cdot \frac{c(w_1^1)}{N} \\ \text{if } (c(w_1^1) < k_2), P(w_m | w_1^{m-1}) = \frac{n_{1,1}}{N} \end{array} \right. \end{array} \right.$$

}Eq (3.7)

if $(c(w_1^m) \leq k_2) \text{AND}(m > 1)$,

Apply the back-off recursive procedure

{ if $c(w_1^{m-1}) > k_2$
 { if $m > 2$, $P(w_m | w_1^{m-1}) = \alpha(w_1^{m-1}) \cdot P(w_m | w_2^{m-1})$
 if $m = 2$, $P(w_m | w_1^{m-1}) = \alpha(w_1^1) \cdot P(w_2)$
 }
 if $c(w_1^{m-1}) \leq k_2$
 { if $m > 2$, $P(w_m | w_1^{m-1}) = P(w_m | w_2^{m-1})$
 if $m = 2$, $P(w_m | w_1^{m-1}) = P(w_2)$
 }
 }

.....Eq (3.8)

3.3 Disambiguation Technique

Using a variant of A^* -based algorithm; e.g. beam search, is the best known way for obtaining the most likely sequence of analyses among the exponentially increasing space $S = \underline{a}_1 \times \underline{a}_2 \times \dots \times \underline{a}_L$ of possible sequences (paths) implied by the trellis's topology in light of the MAP formula by obtaining:

$$\begin{aligned} \underline{Q} &= \arg \max_{\mathbf{s}} \{P(a_{1,j_1}^{L,j_L})\} = \\ & \arg \max_{\mathbf{s}} \left\{ \prod_{i=1}^L P(a_{i,j_i} | a_{(i-h),j_{(i-h)}}^{(i-1),j_{(i-1)}}) \right\} = \dots \text{Eq (3.9)} \\ & \arg \max_{\mathbf{s}} \left\{ \sum_{i=1}^L \log P(a_{i,j_i} | a_{(i-h),j_{(i-h)}}^{(i-1),j_{(i-1)}}) \right\} \end{aligned}$$

To obtain the sequence realizing this maximization, the A^* algorithm follows a best-first path strategy while selecting the path (through the trellis) for expanding next. This best-first strategy is interpreted in the sense of the statistical score of the path till its terminal expansion node $a_{k,j}$ given by:

$$g(k, a_{k,j_k}) = \sum_{i=1}^k \log P(a_{i,j_i} | a_{(i-N+1),j_{(i-N+1)}}^{(i-1),j_{(i-1)}}) \dots \text{Eq (3.10)}$$

To realize maximum search efficiency; i.e. minimum no. of path expansions, a heuristic function (typically called h^*) is added to the g function while selecting the next path to expand during the A^* search so that:

$$f^*(k, a_{k,j_k}, L) = g(k, a_{k,j_k}) + h^*(k, a_{k,j_k}, L) \dots \text{Eq (3.11)}$$

To guarantee the admissibility of A^* search; i.e. the guarantee for the search to terminate with the path with maximum score, the h^* function must not go below a minimum upper bound of the probability estimation of the remainder of the nodes sequence in the path being expanded. For our problem this function is being estimated according to:

$$h^*(k, q_{k,j_k}, L) = \begin{cases} \sum_{i=k+1}^L \log(P_{\max,N}) = (L-k) \cdot \log(P_{\max,N}); & L \geq N, k \geq N-1 \\ \sum_{i=N}^L \log(P_{\max,N}) + \sum_{i=k+1}^{N-1} \log(P_{\max,i}) & L \geq N, k < N-1 \\ = (L-N+1) \cdot \log(P_{\max,N}) + \sum_{i=k+1}^{N-1} \log(P_{\max,i}); & \\ \sum_{i=k+1}^L \log(P_{\max,i}); & L < N \end{cases} \dots \text{Eq (3.12)}$$

Where $P_{\max,k} = \max_{\forall w_1^k} P(w_k | w_1^{k-1})$; $1 \leq k \leq N$ which can be obtained from the n-gram statistical language model that is already built.Eq (3.13)

For proofs and full details on the statistical disambiguation methods reviewed here, the reader is kindly referred to [2], [6], [16], [21], [23].

The “ A^* algorithm” is illustrated below¹.

1. Initialize by creating a stack SS holding nodes of type Q (path L , score s) and push the root node (a fake node) with score $s = \log 1 = 0$ and empty path.
2. Pop up the surface node Q .
3. If length of path in Q is L , exit with the most likely path L .
4. Expand Q to nodes of next column with scores calculated from equations (4.10), (4.11), (4.12), and (4.13), and push them into SS .
5. Reorder SS in a descending order according to the s field.
6. Go to step 2.

One of the advantages of this search process is that not only the 1^{st} most likely path can be obtained, but also the 2^{nd} , the 3^{rd} , etc., and in general the m^{th} most likely path can be obtained using the following modification.

¹ This illustration is quoted without modification from [6] since it is one of the best representations for the A^* search algorithm. (With courtesy to Attia 2005 [6]).

1. Initialize by creating a stack SS holding nodes of type Q (path L , score s) and push the root node (a fake node) with score $s = \log 1 = 0$ and empty path. Set a counter c to zero.
2. If SS is empty, exit with the c^{th} most likely path L_c .
3. Pop up the surface node Q .
4. If length of path in Q is L , increment c .
5. If c is equal to m , exit with the m^{th} most likely path L_m .
6. Expand Q to nodes of next column with scores calculated from equations (4.10), (4.11), (4.12), and (4.13), and push them into SS .
7. Reorder SS in a descending order according to s field.
8. Go to step 2.

The number of nodes in the stack could increase quickly that it may become impractical to hold them all in. Fortunately, we know that only a few best-scoring nodes at any column will most probably contribute to the most likely path, hence deleting some least-scoring nodes from the stack is very unlikely to miss the most likely solution. This practical modification leads to the Beam search listed below.

1. Initialize by creating a stack SS holding nodes of type Q (path L , score s) and push the root node (a fake node) with score $s = \log(1) = 0$ and empty path.
2. Pop up the surface node Q .
3. If length of path in Q is L , exit with the most likely path L .
4. Expand Q to nodes of next column with scores calculated from equations (4.10), (4.11), (4.12), and (4.13), and push them into SS .
5. Reorder SS in a descending order according to s field.
6. Truncate bottom nodes from SS according to some criteria R (e.g. predefined number of best scoring paths, predefined ratio of the score of the top scoring path on the stack, ...) keeping the size of the remaining nodes - *beam size* - within the processing power and storage capacity of the target hardware.
7. Go to step 2.

Finally we present below the most practically general “ m^{th} most likely path Beam-Search” algorithm as a combination of the “ m^{th} most likely path A^* ” algorithm and the “Beam-Search” one.

1. Initialize by creating a stack SS holding nodes of type Q (path L , score s) and push the root node (a fake node) with score $s = \log 1 = 0$ and empty path. Set a counter c to zero.
2. If SS is empty, exit with the c^{th} most likely path L_c .
3. Pop up the surface node Q .
4. If length of path in Q is L , increment c .
5. If c is equal to m , exit with the m^{th} most likely path L_m .
6. Expand Q to nodes of next column with scores calculated from equations (4.10), (4.11), (4.12), and (4.13), and push them into SS .
7. Reorder SS in a descending order according to s field.
8. Truncate bottom nodes from SS according to some criteria R so that the number of remaining nodes equals the maximum allowable stack size (beam size).
9. Go to step 2.

Chapter 4

Disambiguating a Hybrid of Unfactorized and Factorized Words' Sequences

The performance of the architecture shown in figure 2.6 above that is presented later in chapter 5 of this thesis shows a sluggish attenuation of the disambiguation error margin, especially for the syntactic diacritization, versus the increase of expensive annotated training corpus.

A disambiguation of full-form words instead of factorized ones has been tried in figure 4.1 below. The experimental results of this system in chapter 5 show a faster decrease of the error margin, but it suffers from a deficient coverage over the huge Arabic vocabulary.

Note: all components of figure 4.1 are described below while talking about figure 4.2 below.

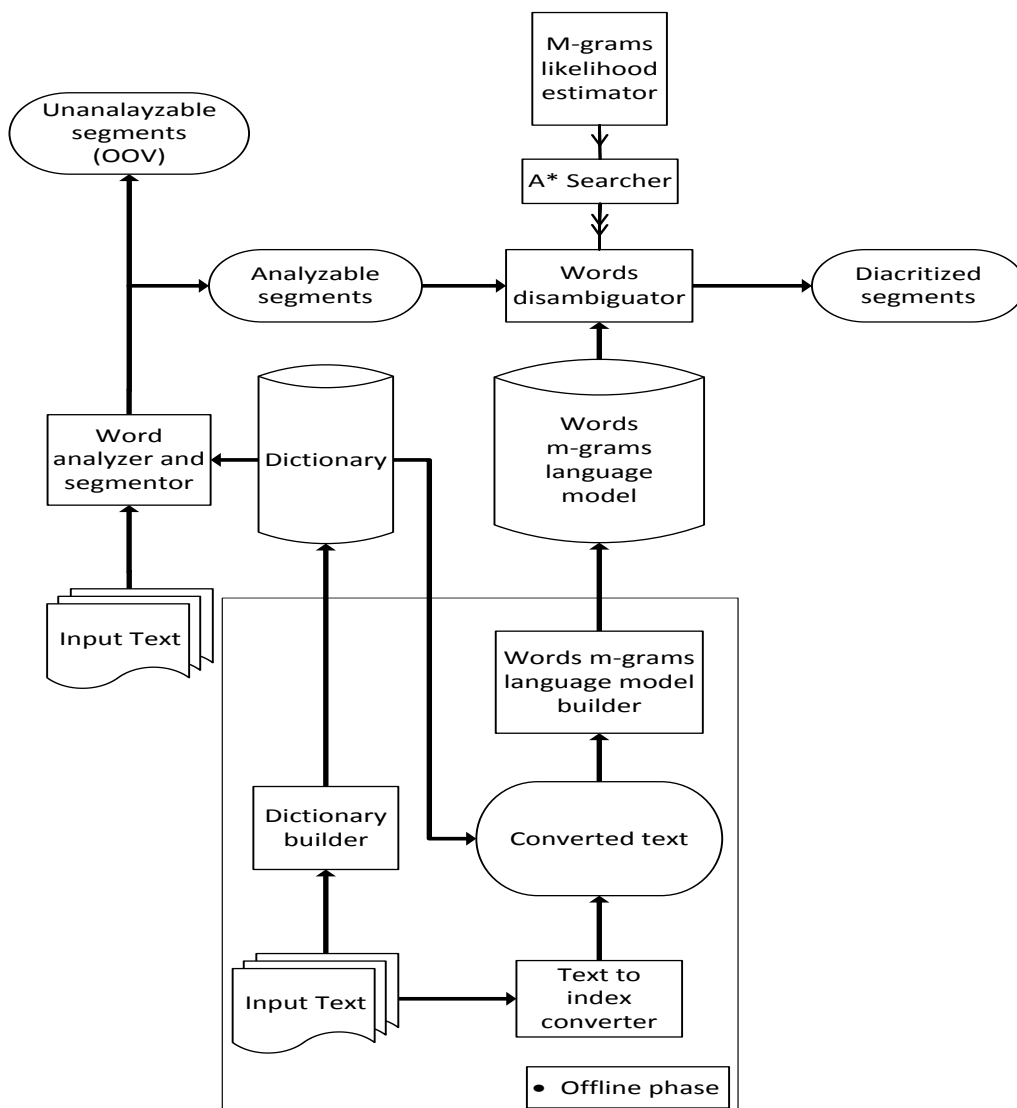


Figure 4.1: The hybrid Arabic text diacritization architecture disambiguating factorized and full-form words.

To capture both a high coverage and a rapid attenuation of the error margin, the hybrid architecture of both approaches illustrated by figure 4.2 below has been tried and illustrated in

this chapter. The hybrid system consists of two phases, the offline phase and the runtime phase; each of them is discussed below.

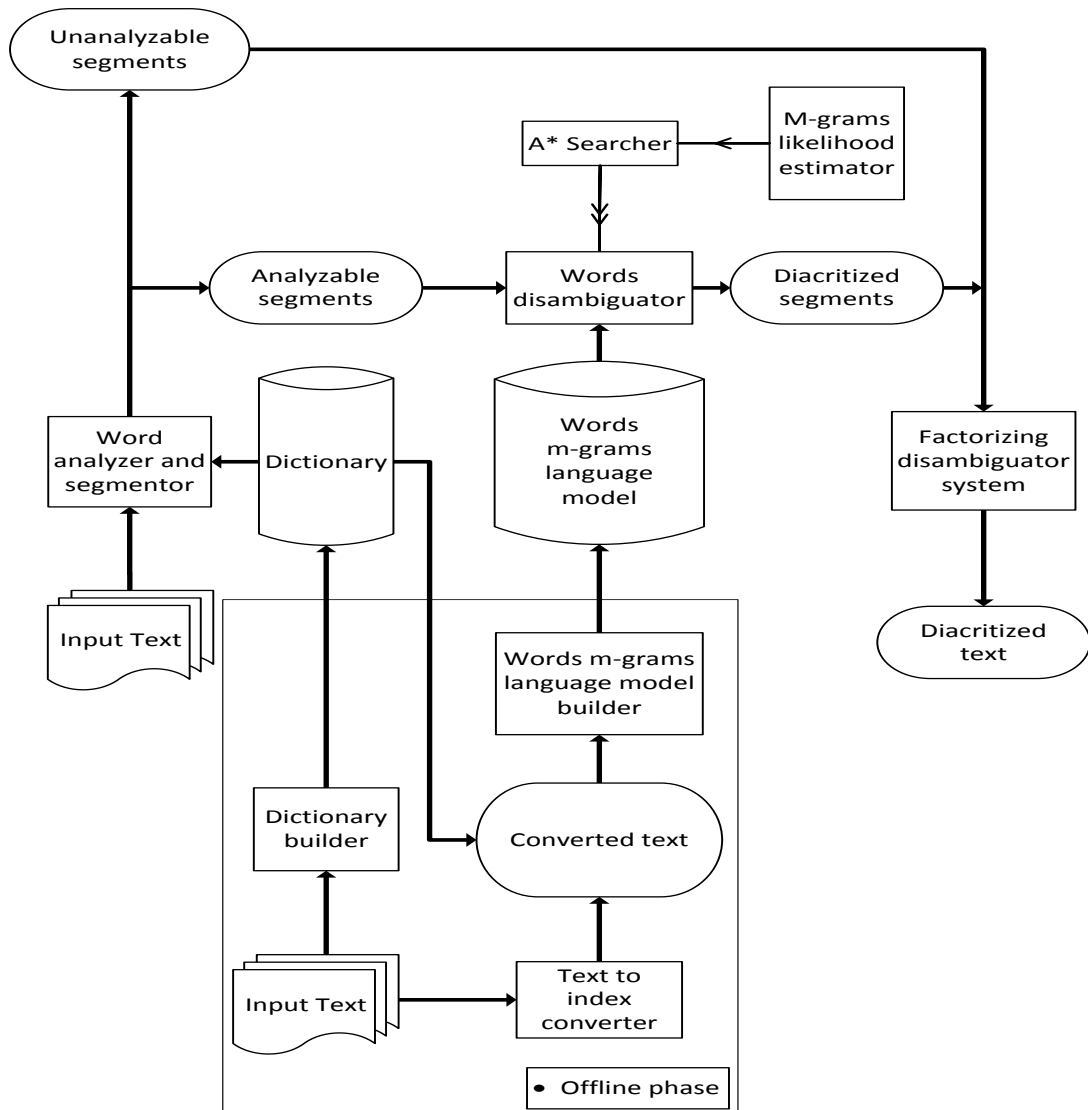


Figure 4.2: The hybrid Arabic text diacritization architecture disambiguating factorized and full-form words.

4.1 The Offline Phase

In the disambiguation in the full-form words, a huge dictionary of words is required; this means a huge search time may occur, which is a big problem; since this dictionary will be used and called many times during the offline phase to build the language model and also during the runtime phase to analyze the input word to its possible analyses.

So, it is clear now that the dictionary should be built in such smart enough way that decreases the search time as much as possible to speed up the whole process of diacritization.

4.1.1 Dictionary Building Process

The presented way for building the dictionary requires the alphabets to have serialized IDs to increase the search speed, so all keyboard characters are collected and divided into groups, and then using the ASCII code of each character; a tabulated function that maps the ASCII code to its ID is implemented. Table 4.1 below represents the characters distributed over the groups with their ASCII¹ codes, the mapping key and the new IDs for the characters.

As shown in table 4.1 below; we have 35 ASCII code ranges. This means that to map a certain character to its ID, we search for its ASCII code range using the famous binary search algorithm [21]; which means that the maximum number of iterations is $\text{Log}_2(35) = 5$ iterations as maximum. Then we add the mapping key to the character's ASCII code (e.g.: the ID of “i” that has the ASCII code “-61” is “-61+91” = 30).

Given the mapping table mentioned above; the design of the dictionary structure is easy where each word is represented as follows:

Number of word letters	1st letter ID	Sum of the word's letters' IDs	The word without diacritization	The word with diacritization
------------------------	---------------	--------------------------------	---------------------------------	------------------------------

e.g. the word “كُتِبَ” will be represented as follow:

3	8	48	كتب	كُتِبَ
---	---	----	-----	--------

Using the representation mentioned above for the words; the dictionary is implemented as shown in figure 4.3 below.

¹ The codes that are mentioned here is signed characters. i.e. their range is varying from -127 to 127

Cat.	Characters	ASCII Range	Mapping key	New IDs
Arabic letters come at the start of the words	ي	-19	+19	0
	م, ن, ه, و	-29 → -26	+30	1→4
	ل	-31	+36	5
	ف, ق, ك	-35 → -33	+41	6→8
	ط, ظ, ع, غ	-40 → -37	+49	9→12
	ت, ث, ج, ح, خ, د, ذ, ر, ز, س, ش, ص, ض	-54→-42	+67	13→25
	ا, ب	-57→ -56	+83	26→27
	! , آ	-59 -62→ -61	+87 +91	28 29→30
Arabic letters does not come at the start of the words	ى	-20	+51	31
	ة	-55	+87	32
	ئ	-58	+91	33
	ؤ	-60	+94	34
	ء	-63	+98	35
Diacritics	°	-6	+42	36
	˘	-8	+45	37
	˙	-11→-10	+49	38→39
	˚	-16 →-13	+56	40→43
Arabic signs	÷	-9	+53	44
	—	-36	+81	45
	×	-41	+87	46
	؟	-65	+112	47
	؛	-70	+118	48
	،	-95	+144	49
	“ ”	-111→ -110	+161	50→51
Numbers	0,1,2,3,4,5,6,7,8,9	48→ 57	+4	52→61
Delimiters	Tab, New line	9→ 10	+53	62→63
	Enter	13	+51	64
	Space	32	+33	65
Arabic and English signs	!, ", #, \$, %, &, ', (,), *, +, ,, -, ., /	33→47	+33	66→80
	:, ;, <, =, >, ?, @	58→ 64	+23	81→87
	[, \,], ^, _ , `	91→ 96	-3	88→93
	{, , }, ~	123→ 126	-29	94→97
Capital English letters	A→Z	65→90	+33	98→123
Small English letters	a→z	97→122	+27	124→149

Table 4.1: Characters mapping table.

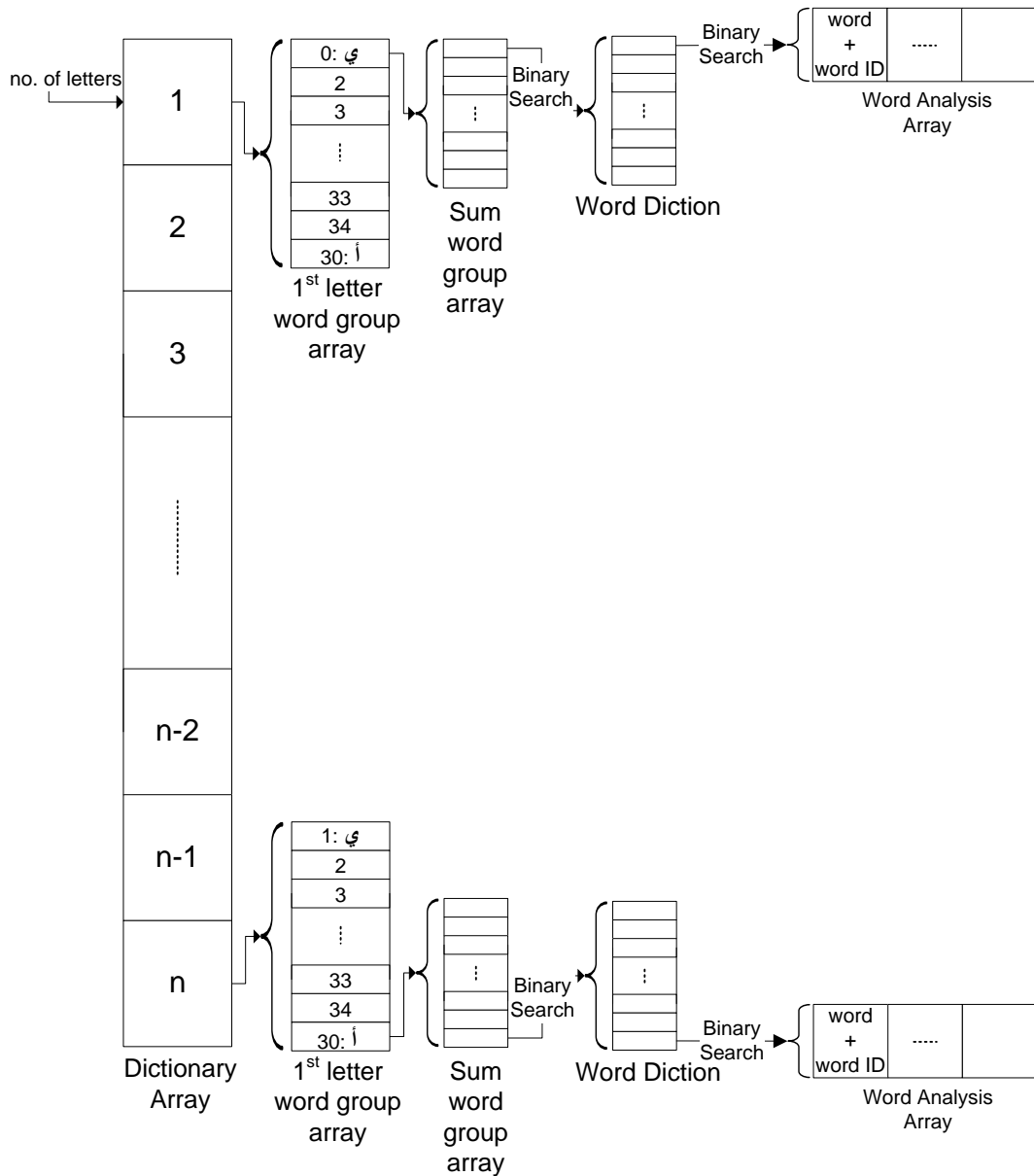


Figure 4.3: Dictionary structure.

According to the structure shown in figure 4.3 above the dictionary carries the word and its analyses (e.g. the word is “كتب” and the analyses are “كُتِبَ - كَتَبَ - كُتِبَ - كُتِبَ...etc”). So to find the ID of a certain word in the dictionary; the binary search is used twice; the first one is in the “Undiacritized word array” to find the location of the analyses of the Undiacritized word and the second one is in “Word analyses array” to find the ID of the diacritized word as follow:

$$u = D[n].F[i].S[l].W. \mathcal{FD}(\underline{uw}) \quad \dots\dots\text{Eq (4.1)}$$

$$ID = D[n].F[i].S[l].W[u]. \mathcal{FA}(\underline{dw}) \quad \dots\dots\text{Eq (4.2)}$$

Where:

- D : is the “Dictionary” array.
- F : is the “First Letter Word Group” array.
- S : is the “Sum Word Group” array.
- W : is the “Word Diction” array.
- \mathcal{FD} : is a function that apply the binary search technique on the “Word Diction” array to find an undiacritized word.
- \mathcal{FA} : is a function that apply the binary search technique on the “Word Analyses” array to find a diacritized word.
- u : is the location of the undiacritized word in the “Word Diction” array.
- n : is the number of letters in the undiacritized word.
- i : is the first letter ID.
- l : is the summation of all letters IDs in the undiacritized word.
- \underline{uw} : is the undiacritized word string.
- \underline{dw} : is the diacritized word string.
- The “.” operator means that the right side of the “.” is an element of a structure (ex. A.B: this means that A is a structure and B is an element of A).

ex. to find the ID of the word “كُتِبَ”:

$$u = D[3].F[8].S[48].W. \mathcal{FD}(\text{“كُتِبَ”})$$

$$ID = D[3].F[8].S[48].W[u]. \mathcal{FA}(\text{“كُتِبَ”})$$

Figure 4.4 below shows how the word “كُتِبَ” is found in the dictionary.

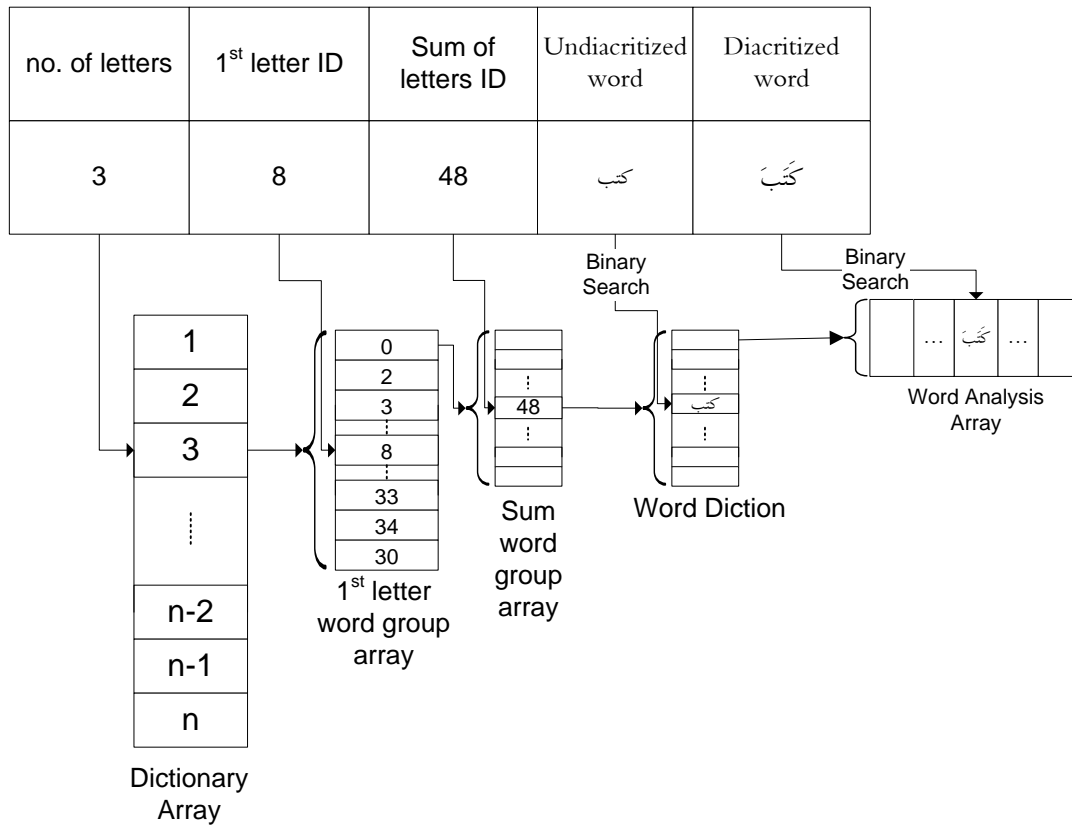


Figure 4.4: Dictionary searching criterion.

In the corpus of (3,250K words) that is used in this thesis and discussed in details in chapter 5; it is found that the total unique words are about 260,909 words, the maximum number of the undiacritized words that have the same number of letters, same first letter and the same sum of letters is 173 words, and it is found also that the maximum number of analyses per word is 25 analyses. So considering the worst case we find that the maximum number of iterations to find a word will be about $\text{Log}_2(173) + \text{Log}_2(25) \approx 13$ iterations. This method is used instead of applying the binary search on all words that will have a number of iterations of $\text{Log}_2(260,909) \approx 18$ iterations. This means that the worst-case performance increase is about (27%); but for the practical cases it is found that the average number of the undiacritized words that have the same number of letters, same first letter and the same sum of letters is 10 words, and the average number of analyses per word is 8 analyses. This means that the average number of iterations to find a word will be about 7 iterations; i.e. the average performance increase is about 61%.

4.1.2 Language Model Building Process

Since it is easier to deal with numbers than strings in software programming, so all training corpus words are converted to their IDs using the “Text-to-index converter” module that is shown in figure 4.2, that uses the built dictionary above in the conversion process.

After corpus conversion, the converted corpus is then passed to the “Word n-gram language model builder” that is shown in figure 4.2 to build a tri-gram language model for the given corpus according to the criterion that was discussed in chapter 3 above..

4.2 The Runtime Phase

In this phase, the input text is passed to the “Word analyzer and segmentor” module that is shown in figure 4.2 to search for each word of the input text in the dictionary. If the word is found, the word is called “analyzable” and all its diacritization occurrences (i.e. word analyses) are retrieved from the dictionary. A consequent series of analyzable words in the input text is called “analyzable segment” and the remaining words called “un-analyzable segment”. The next example illustrates this idea:

Assume that the input text is:

"أوجد المونديال فرصا لعمل الشباب من خلال أجهزة العرض والشاشات للمقاهي والأسواق والفنادق".

Then by passing this text to the “Word analyzer and segmentor” module; the output will be as follows: (note: “A” means Analyzable Segment and “U” means Un-Analyzable Segment).

U	A	U	A				U	A				
والفنادق	والأسواق للمقاهي	والشاشات	العرض	أجهزة	تسويق	خلال	من	الشباب	لعمل	فرصا	المونديال	أوجد
	وَالْأَسْوَاقِ لِلْمَقَاهِي		الْعَرْضِ	أَجْهَزَةَ	تَسْوِيقِ	خِلَالَ	مِنْ	الشَّبَابِ	لِعَمَلِ	فُرْصًا		أَوْجَدَ
			الْعَرْضُ	أَجْهَزَةُ	تَسْوِيقُ	خِلَالَ	مِنْ	الشَّبَابِ	لِعَمَلِ			أَوْجَدَ
			الْعَرْضِ	أَجْهَزَةُ	تَسْوِيقِ	خِلَالَ	مِنْ	الشَّبَابِ				
			⋮	⋮	⋮	⋮	⋮	⋮				

All the diacritization occurrences of the words in an analyzable segment constitute a lattice, as shown by figure 4.5 below, that is disambiguated via n-grams probability estimation and A* lattice search to infer the most likely sequence of diacritizations in the “Word disambiguator” module [2], [16], [21].

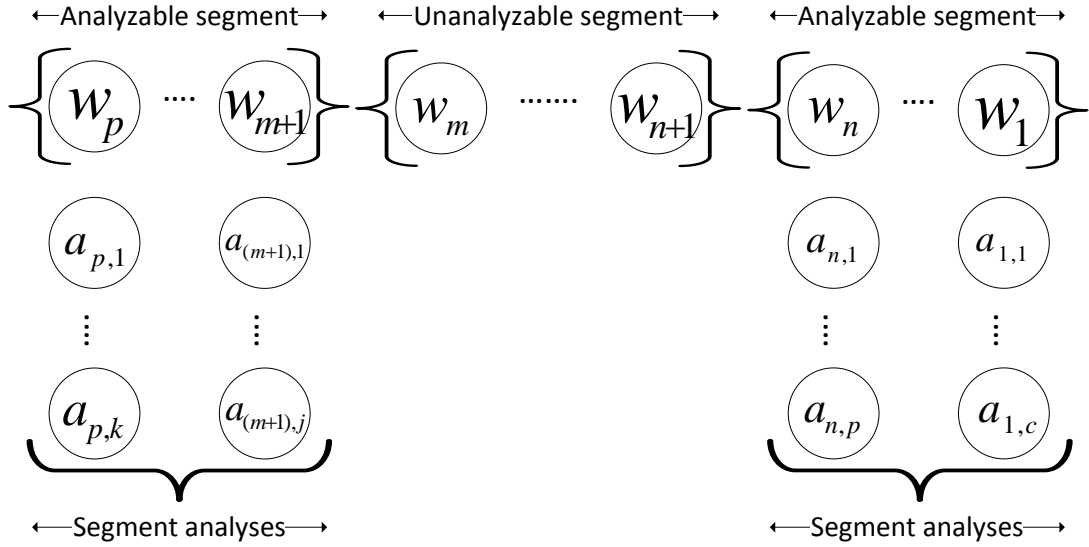


Figure 4.5: Phrase segmentation

The diacritized full-form words of the disambiguated analyzable segments are concatenated to the input words in the un-analyzable segments to form a less ambiguous sequence of Arabic text words. The latter sequence is then handled by the “Factorizing Disambiguator” module that is illustrated in chapter 2 above.

By applying that on the above example; it is found that the new text that will be sent to the “Factorizing Disambiguator” module become as follows:

" أُوْجِدَ الْمُؤْنِدِيَالُ فُرْصًا لِعَمَلِ الشَّبَابِ مِنْ خِلَالِ أَجْهَزَةِ الْعُرْضِ وَالشَّاشَاتِ لِلْمَقَاهِي وَالْأَسْوَاقِ
وَالْفَنَادِقِ".

After the “Factorizing Disambiguator” module the output text becomes as follows:

" أُوْجِدَ الْمُؤْنِدِيَالُ فُرْصًا لِعَمَلِ الشَّبَابِ مِنْ خِلَالِ أَجْهَزَةِ الْعُرْضِ وَالشَّاشَاتِ لِلْمَقَاهِي وَالْأَسْوَاقِ
وَالْفَنَادِقِ".

Chapter 5

Experimental Results

At the beginning of this chapter, a comparison with recent related work is held. Then the specifications of the training and testing corpus that are used for our experiments are discussed, after that some useful statistics that are found during the training phase are mentioned, a detailed discussion for the experiments and the results for the hybrid system with some comparisons with the factorizing system are then mentioned, at the end of this chapter complete error analyses for the results are discussed.

5.1 A Comparison with the Recent Related Work

Among the other recent attempts on the tough problem of Arabic text diacritization, two groups have made remarkable attempts.

- 1 Zitouni et al. [27] follow a statistical model based on the framework of maximum entropy. Their model combines different sources of information ranging from lexical, segment-based, and POS features. They use statistical Arabic morphological analysis to segment each Arabic word into a prefix, a stem, and a suffix. Each of these morphemes is called a segment. POS features are then generated by a parsing model that also uses maximum entropy. All these features are then integrated in the maximum entropy framework to infer the full diacritization of the input words' sequence [27].
- 2 Habash and Rambow [14] use Morphological Analysis and Disambiguation of Arabic (MADA) system [13]. They use the case, mood, and nunation as features, and use the Support Vector Machine Tool (SVM Tool) [12] as a machine learning tool. They then build an open-vocabulary SLM with Kneser-Ney smoothing using the SRILM toolkit [27]. Habash and Rambow made experiments using the full-form words and a lexemes (prefix, stem, and suffix) citation form.

The best results are the ones; they obtain with the lexemes form with trigram SLM [14]. The Arabic diacritizer of Zitouni et al. and that of Habash and Rambow are both trained and tested using the LDC's Arabic Treebank of diacritized news stories, text-part 3, v1.0. This text corpus which includes 600 documents ($\approx 340K$ words) from AnNahar newspaper text is split into a training data ($\approx 288K$ words) and test data ($\approx 52K$ words) [14], [27].

To our knowledge, their systems are the best performing currently, and we have set up our experiments to allow a fair comparison between our results to theirs.

So, in order to allow a fair comparison with the work of Zitouni et al. [27] and that of Habash and Rambow [14], we used the same training and testing corpus; and also we adopt their metric.

1. Counting all words, including numbers and punctuation. Each letter (or digit) in a word is a potential host for a set of diacritics [14].
2. Counting all diacritics on a single letter as a single binary choice. So, for example, if we correctly predict a “Fatha” but get the vowel wrong, it counts as a wrong choice [14].
3. We approximate non-variant diacritization by removing all diacritics from the final letter (Ignore Last), while counting that letter in the evaluation [14].
4. We give diacritic error rate (DER) which tells us for how many letters we incorrectly restored its and word error rate (WER), which tells us how many words had at least one DER [14].

We use the following terms during the rest of this chapter:

1. “Quadruple citation form”: to describe the system that is illustrated in chapter two above that makes the morphological analyses in the form of (Prefix, Root, Form, and Suffix).
2. “Lexemes citation form” to describe the system that makes the morphological analysis in the form of (Prefix, Stem, and Suffix).
3. “Q-Tagging model”: is an Arabic diacritizer system that uses the “Quadruple citation form” in the diacritization process. This system is illustrated in chapter two above.
4. “L-Tagging model”: is an Arabic diacritizer system that uses the “Lexemes citation form” in the diacritization process and if there exists (OOV) in the stems dictionary, it backs-off to the “Q-Tagging model”.
5. “Hybrid-Tagging model”: is an Arabic diacritizer that uses the Full word citation form in the diacritization process and if there exists (OOV) in the words' dictionary, it backs-off to the “Q-Tagging model”. This system is illustrated in section four above.
6. “WL-Tagging model”: is an Arabic diacritizer that uses the Full word citation form in the diacritization process and if there exists (OOV) in the words' dictionary, it backs-off to the “L-Tagging model”.

The results in table 5.1 below show that our presented hybrid system has the best results.

Model	All Diacritics		Ignore last	
	WER	DER	WER	DER
Habash and Rambow [14]	14.9%	4.8%	5.5%	2.2%
Zitouni et al. [27]	18.0%	5.5%	7.9%	2.5%
Q-Tagging model with trigram SLM: An Arabic diacritizer system that uses the “Quadruple citation form” in the diacritization process.	31.8%	10.3%	7.2%	2.7%
L-Tagging model with trigram SLM: An Arabic diacritizer system that uses the “Lexemes citation form” in the diacritization process and if there exists (OOV) in the stems dictionary, it backs-off to the “Q-Tagging model”	22.1%	6.9%	6.9%	2.5%
WL-Tagging model with trigram SLM: An Arabic diacritizer that uses the Full word citation form in the diacritization process and if there exists (OOV) in the words dictionary, it backs-off to the “L-Tagging model”	17.1%	5.8%	3.8%	2.1%
Hybrid -Tagging model with trigram SLM: Arabic diacritizer that uses the Full word citation form in the diacritization process and if there exists (OOV) in the words' dictionary, it backs-off to the “Q-Tagging model”	12.5%	3.8%	3.1%	1.2%

Table 5.1: Propose Hybrid diacritizer morphological & syntactical (case ending) diacritization error rates versus other state-of-the-art systems; our best results are shown in boldface.

By studying a sample of (2,243 words) from the corpus that is used in the above comparison, we found that:

1. Around 4.99% of the words do not have a syntactic diacritic, but most of them are Arabic and transliterated proper names.
2. Around 2.98% of the words do not have a syntactic diacritic, but most of them are Arabic and foreign names.

The reader should take into consideration the drawbacks of the above corpus when reading the results.

Following, a comprehensive study for the comparison between the factorized and the unfactorized systems.

5.2 Experimental Setup

The annotated DB used to train our aforementioned Arabic diacritizers consists of the following packages:

- I. A standard Arabic text corpus with as size $\approx 750K$ words collected from numerous domains over diverse domains as shown by table 5.2 below. This package is called TRN_DB_I.

Domain	Size	% of total size
General news	150,000	20.0%
Political news	120,000	16.0%
Qur'an	80,000	10.7%
Dictionary entries explanation	57,000	7.6%
Scientific press	50,000	6.7%
Islamic topics	50,000	6.7%
Sports press	50,000	6.7%
Interviews	49,000	6.5%
Political debate	35,000	4.7%
Arabic literature	31,000	4.1%
Miscellaneous	30,000	4.0%
IT Business & management	20,000	2.7%
Legislative	20,000	2.7%
Text taken from Broadcast News	8,500	1.1%
Phrases of common words	5,500	0.7%
Total:	750K words	100%

Table 5.2: Distribution of TRN_DB_I over diverse domains.

It should be noted that the text of each domain is collected from several sources. This text corpus is morphologically analyzed and phonetically transcribed. All these kinds of annotations are manually revised and validated [26].

- II. An extra standard Arabic text corpus with as size $\approx 2,500K$ words, those are only phonetically transcribed in full without any extra annotation. This corpus is mainly extracted from classical Islamic literature. This package is called TRN_DB_II¹. This kind of annotation is done manually but with just one revision. So, it might contain some errors that could be a source of some errors.
- III. The test data is rather challenging. It consists of 11K words that are manually annotated for morphology and phonetics. This test text covers diverse domains as shown by table 5.3 below. This test package is called TST_DB². The text of TST_DB is collected from several domains other than those used to obtain the text of TRN_DB_I and TRN_DB_II.

¹ <http://www.RDI-eg.com/RDI/TrainingData> is where to download TRN_DB_II.

² <http://www.RDI-eg.com/RDI/TestData> is where to download TST_DB.

Domain	% of total size
Religious-Heritage	17.4%
Religious-Contemporary	17.1%
Legislative-Contemporary	13.0%
Social-Contemporary	9.1%
Sports- Contemporary	8.8%
Social-Heritage	8.5%
Arts-Contemporary	9.0%
Scientific- Contemporary	6.7%
Political- Contemporary	5.5%
Historical-Heritage	2.7%
Literature-Heritage	2.2%
Total:	100%

Table 5.3: Distribution of TST_DB over diverse domains.

The error counting conventions in the following experiments are as follows:

1. The word is said to has a “Morphological error” if the form of the word is wrong.
2. The word is said to has a “Syntactical error” if the syntactical diacritic (Case-ending) of the word is wrong.
3. If the word has a “Morphological error” and a “Syntactical error” it will be counted in the “Morphological errors” only, and it will not be counted in the “Syntactical errors”. So at any time the sum of the errors is the summation of the morphological errors and the syntactical errors.

5.3 The effect of the corpus size on the system behavior

During the implementation of the dictionary and the language model of the un-factorizing part of the hybrid system; some useful statistics are found, and they are stated here.¹

- i. The increase of the training data the increase of the unique words in the dictionary as shown in table 5.4 below. Actually, the data used is balanced from (64K words) to (750K words), so we notice a real growing rate in the dictionary size; but after that the data is biased to the Islamic domain, so we notice that the growing rate decreased especially from (3,250K words) to (14,000K words).

¹ The maximum used training size was the 3,250K but just to see the effect of increasing the training size on the dictionary, so the 4,000K and 14,000K training corpora were tried. But they did not be used for the training because they are mainly Islamic domain; which will make the training data very biased.

Corpus size(K words)	Dictionary size(words)
64	21,105
128	34,785
256	60,491
512	97,449
750	136,602
1,000	157,854
2,000	216,835
3,250	260,909
4,000	297,026
14,000	506,690

Table 5.4: The effect of the training corpus size on the dictionary size.

- ii. To study the effect of the corpus size increase on the possible analyses for words (e.g. the word is “كُتِبَ” and the analyses are “كُتِبَ - كُتِبُ - كَتَبَ - كُتِبَ...etc”), it is found that by increasing the corpus size the possible analyses per words increase too; But again the saturation is occurred by the excess increase in a single domain (this is clear from 3,250K to 14,000K words) in table 5.5 below.

Corpus size(K words)	Maximum number of analyses per word
64	9
128	11
256	13
512	15
750	16
1,000	19
2,000	23
3,250	25
4,000	25
14,000	30

Table 5.5: The effect of the training corpus size on the number of analyses per word.

- iii. Although the increasing of the corpus size increases the number of analyses per word, but the Out Of Vocabulary (OOV) did not become small as shown in figure 5.1 below. This means that the many increases in a single domain may cover most of the words in that domain, but still there are more uncovered analyses for the words in other different domains. So may be a medium corpus size (about 5,000K words), but it has a balanced distribution on all domains will decrease the ratio of the OOV and will increase the coverage ratio.

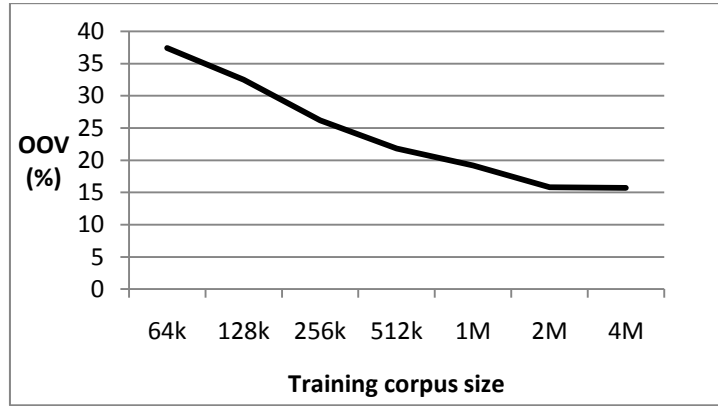


Figure 5.1: Corpus size versus the out of vocabulary OOV.

5.4 Experiments Design and Results Analysis

The experiments that are discussed below have been conducted to evaluate the performance of Arabic text diacritization via both the two architectures presented in this thesis; the one disambiguating factorized text features - called “Factorizing Diacritizer” - and the hybrid one – called “Hybrid Diacritizer”.

5.4.1 Experiment no. 1:

This experiment compares the diacritization accuracy of the two architectures with both relying on SLM’s built from the same Arabic text corpus. The change of diacritization accuracy of both with the gradual increase of training corpus size is also sensed. All these measures are registered in table 5.6 below.

Training corpus size (words)	Morphological errors		Syntactical errors	
	Factorizing diacritizer	Hybrid diacritizer	Factorizing diacritizer	Hybrid diacritizer
128K	11.5%	9.2%	25.9%	20.8%
256K	11.8%	7.9%	24.8%	18.2%
512K	9.9%	6.5%	22.7%	16.3%
SizeOf(TRN_DB_I) = 750K	7.5%	7.0%	21.6%	15.8%

Table 5.6: Morphological & syntactic diacritization accuracies of the factorizing diacritizer versus the hybrid one.

These results show that the hybrid diacritizer outperforms the factorizing one with the mentioned training and test data. While the difference between the syntactical diacritization errors' rate is clearly wide, the difference between the morphological errors' rate is much closer and is vanishing with the increase of training data.

So, one may also speculate that the accuracy of the factorizing diacritizer may catch that of the hybrid one with a much more increase in the size of the training

data that is needed to capture the more complicated behavior of the Arabic syntactic phenomenon than the Arabic morphological one.

Unfortunately, at the moment there is no more annotated data of the type of TRN_DB_I that is needed to build language models for the factorizing diacritizer to compare at a higher training corpus size.

5.4.2 Experiment no. 2:

As the training data of type TRN_DB_II is less expensive to afford than that of type TRN_DB_I, we could afford a training corpus of the former type of size 2,500K words. So, the un-factorizing part of the hybrid diacritizer can rely on SLM's from up to 750K + 2,500K words.

The factorizing diacritizer can of course not benefit from training data beyond that of the annotated 750K words of TRN_DB_I.

This experiment hence aims to study the effect of increasing the training data size in the un-factorizing SLM on the error rate of the hybrid Arabic diacritizer. Table 5.7 below shows the obtained measured error rates.

Training corpus size (words)	Hybrid diacritizer	
	Morphological errors	Syntactical errors
SizeOf(TRN_DB_I) = 750K	7.0%	15.8%
SizeOf(TRN_DB_I) + ½ SizeOf(TRN_DB_II) = 2000K	4.9%	13.1%
SizeOf(TRN_DB_I) + SizeOf(TRN_DB_II) = 3250K	3.6%	12.7%

Table 5.7: Morphological and syntactic diacritization error rate of the hybrid diacritizer at large training data.

This experiment reveals that the syntactical diacritization accuracy seems to asymptote its saturation at training corpora exceeding 2000K words (This may be due to the biasing in the training data).

5.4.3 Experiment no. 3:

From the above experiment, we wanted to test the effect of the training data on the different domains, so the experiment is done over the Islamic domain which is the dominant in the training corpus and the news domain for its importance. From table 5.8 below, we can deduce that:

- i. Across domains, there is some improvement for the Islamic domain over the others (because most of the training data is Islamic data).
- ii. This experiment reveals that the syntactical diacritization accuracy seems to asymptote its saturation at training corpora exceeding 2000K

words even on the Islamic domain. It seems that it is hard to get further significant enhancement via statistical means alone by increasing the training corpus. Achieving error rates below that 12.7% or so seems to need some genuine merger with more linguistically informed tools.

Training corpus size (words)	Test data domain	Hybrid diacritizer	
		Morphological Errors	Syntactical Errors
SizeOf(TRN_DB_I) = 750K	Islamic part in TST_DB	5.4%	14.1%
	News part in TST_DB	4.3%	16.3%
	Total TST_DB	7%	15.8%
SizeOf(TRN_DB_I) + 1/2 SizeOf(TRN_DB_II) = 2,000K	Islamic part in TST_DB	5.2%	10.9%
	News part in TST_DB	4.2%	15.5%
	Total TST_DB	4.9%	13.1%
SizeOf(TRN_DB_I) + SizeOf(TRN_DB_II) = 3,250K	Islamic part in TST_DB	3.7%	10.8%
	News part in TST_DB	3.7%	15.4%
	Total TST_DB	3.6%	12.7%

Table 5.8: studying the effect of the training data size changing on different domains

5.4.4 Experiment no. 4:

The architecture of the hybrid diacritizer has been explained in chapter 4 above where input words not found in the full-form words' dictionary (also called OOV words) are handled by the factorizing diacritizer within the statistical context of neighboring diacritized words retrieved from that dictionary. The diacritization word error rate of the hybrid diacritizer (WER_h) has hence two components; the un-factorizing one (WER_{un-fac}) and the factorizing one (WER_{fac});

$$WER_h = WER_{fac} + WER_{un-fac} \quad \dots\dots Eq (5.1)$$

As it is insightful to know the share of both WER_{un-fac} and WER_{fac} in WER_h , all these rates are measured for the hybrid diacritizer running on SLM built from the largest available training data sets; i.e. TRN_DB_I + TRN_DB_II.

These measurements are given by table 5.9 below:

Training corpus size (words)	Test data domain	Ratio of OOV (%)	Morphological Errors (%)			Syntactical Errors (%)		
			WER_{fac}	WER_{un-fac}	$WER_h =$ $WER_{fac} +$ WER_{un-fac}	WER_{fac}	WER_{un-fac}	$WER_h =$ WER_{fac} $+$ WER_{un-fac}
SizeOf(TRN_DB_I) + SizeOf(TRN_DB_II) = 3,250K	Islamic part in TST_DB	13.3	1.9	1.8	3.7	5.3	5.5	10.8
	News part in TST_DB	17.9	2.6	1.1	3.7	10.4	5	15.4
	Total TST_DB	13.7	2.1	1.5	3.6	7.8	4.9	12.7

Table 5.9: Shares of the factorizing & un-factorizing diacritization error rates in the hybrid diacritization error rate.

From table 5.9 above, we can deduce that:

- i. There is a clear correlation between better results and lower OOV.
- ii. If we imagined that we can get enough diacritized data with negligible OOV (which might not be easy, but we like to predict the asymptotic performance of the system), the results will approach 1.5% (or a little less) morphological diacritization errors and 4.9% syntactical diacritization errors. (The performance of the unfactorized for the seen vocabulary only).
- iii. The OOV could be considered a good reference for the unfactorized system performance; i.e. if we build a system that diacritize the complete words directly without any need to back-off to the factorizing system, the errors of this system are partially from the OOV (the higher percentage) and from the internal errors for the seen vocabulary.

5.4.5 Experiment no. 5:

This experiment is to study the system performance, so the memory (for the language models) and the processing time needed for each system are recorded to evaluate the cost of the gain in results shown above.

As shown in table 5.10 below, there is some increase in the memory for the hybrid system compared to the factorizing one. The more data used by the hybrid system the more memory size is needed. The size of the memory increases linearly with the increase of the data size. This increase of the required memory is not that serious with nowadays computer resources.

Training corpus size (words)	Language model size (byte)		
	Factorizing diacritizer	Un-Factorizing diacritizer	Hybrid diacritizer
64K	15.7M	2.3M	18M
128K	33.3 M	4 M	37.3M
256K	60.3 M	8 M	68.3M
512K	113M	15.7 M	128.7M
SizeOf(TRN_DB_I) = 750K	167M	24.2 M	191.2M
SizeOf(TRN_DB_I) + ½ SizeOf(TRN_DB_II) = 2,000K	167M	46.3 M	213.3M
SizeOf(TRN_DB_I) + SizeOf(TRN_DB_II) = 3,250K	167M	60 M	227M

Table 5.10: studying the effect of the increase of the training data on the memory size.

Regarding the time needed by the two systems; the hybrid system outperforms the factorizing system considerably. This is noticed in all the experiments; however, we recorded one of these experiments as shown in table 5.11 below. Our explanation for that is as follows: The hybrid system uses the unfactorized words which form a more compact set for the A^* search than the factorizing system which make the search process faster, but if the right word is not in the search set, this may results in a wrong solution.

Training corpus size (words)	Testing time(min.) [for the TST_DB (11079 words)]	
	Factorizing diacritizer	Hybrid diacritizer
Size Of(TRN_DB_I) = 750K	21.5	9.7

Table 5.11: studying the time consumption by the factorizing and the hybrid systems

From the above experiments, it is found that; in the two systems, the morphological errors are near to each other; which mean that the factorizing and the un-factorizing systems almost acting the same in the morphological diacritization; but the syntactical errors in the two systems are slightly high; however, it is higher in the factorizing system than the un-factorizing one.

5.5 Errors Analysis

Although the presented hybrid system produces low error margins as discussed above (3.6% for the morphological errors and 12.7% for the syntactical errors) which is a good result; since it competes with the stat-of-the-art systems, but the error margins

need to be decreased especially for the syntactical errors. So the produced errors from the presented hybrid system using the test set (TST_DB) are deeply studied below.

It is found that the errors can be divided into two categories, but before talking about these categories, we should revisit the concept of Out-Of-Vocabulary again.

According to chapter 4 the dictionary can be simulated as a set of containers, each container is labeled by an undiacritized word (e.g. “كتب”) and inside this container the possible analyses of this word that appeared during the training phase are put (e.g. “كتب” → “كَتَبَ”, “كُتِبَ”, “كُتِبُ”... etc.). So according to the above definition of the dictionary, the word is said to be OOV if there is no container labeled with the undiacritized string of that word; so, for the above example if the container that is named “كتب” does not exist, then the word “كتب” is declared as an OOV word.

Now back to the above two categories of errors:

1. The first category is from the factorizing system, since if the word is declared as an OOV word, the hybrid system will back-off to the factorizing system. This category is responsible for 2.1% of the total morphological errors (will be named *Morpho_WER_{fac}*) and 7.8% of the total syntactical errors (will be named *Synta_WER_{fac}*) as discussed in experiment no. 4 above. These errors are due to the statistical disambiguation.
2. The second category is from the un-factorizing system. This category is responsible for 1.5% of the total morphological errors (will be named *Morpho_WER_{un-fac}*) and 4.9% of the total syntactical errors (will be named *Synta_WER_{un-fac}*) as discussed in experiment no. 4 above. In this category, we have three types of errors:
 - i. Morphological and syntactical errors due to the statistical disambiguation. About 1% of the total morphological errors (will be named *Stat_Morpho_WER_{un-fac}*) and 1.4% of the total syntactical errors (will be named *Stat_Synta_WER_{un-fac}*). These errors are due to the statistical disambiguation.
 - ii. Morphological errors due to the wrong declaration of the word to be in vocabulary word although it should be declared as an OOV one. For example assume that the input word is “كتب” and assume that the right diacritization of this word is “كُتِبَ”; the problem is happened when the

container named “كتب” is existed in the dictionary, but it does not contain the word “كُتِبَ” instead it includes (“كَبَّ”, “كَبِّ”, “كُتِبَ”... etc.); so the module of word analyzing and segmentation will not declare the word “كتب” as an OOV word, but it will return the found analyses in the dictionary which does not contain the right solution, hence the word will be mistakenly diacritized giving a morphological error. The percentage of this type of errors is about 0.5% of the total morphological errors (will be named $OOV_Morpho_WER_{un-fac}$).

- iii. Syntactical errors due to the wrong declaration of the word to be in vocabulary word although it should be declared as an OOV one. For example assume that the input word is “كتب” and assume that the right diacritization of this word is “كُتِبَ”; the problem is happened when the container named “كتب” is existed in the dictionary but it does not contain the word “كُتِبَ” instead it includes (“كَبَّ”, “كَبِّ”, “كُتِبَ”... etc.); so the module of word analyzing and segmentation will not declare the word “كتب” as an OOV word but it will return the found analyses in the dictionary which does not contain the right solution, hence the word will be mistakenly diacritized giving a syntactical error. The percentage of this type is about 3.5% of the total syntactical errors (will be named $OOV_Synta_WER_{un-fac}$).

Tables 5.12 and 5.13 below can summarize the above discussion.

Total morphological errors		
$Morpho_WER_{fac}$	$Morpho_WER_{un-fac}$	
	$Stat_Morpho_WER_{un-fac}$	$OOV_Morpho_WER_{un-fac}$
	1%	0.5%
2.1%	1.5%	
3.6%		

Table 5.12: Morphological errors analyses.

Total syntactical errors		
<i>Synta_WER_{fac}</i>	<i>Synta_WER_{un-fac}</i>	
	<i>Stat_Synta_WER_{un-fac}</i>	<i>OOV_Synta_WER_{un-fac}</i>
	1.4%	3.5%
7.8%	4.9%	
12.7%		

Table 5.13: Syntactical errors analyses.

Chapter 6

**Conclusion and Future
Work**

6.1 Conclusion

It has got clear after extensive research and experimentation on the tough problem of entity factorizing versus unfactorizing that:

1. The unfactorizing systems are faster to learn but suffer from poor coverage (OOV).
2. The fast learning of the unfactorizing systems and the low preprocessing needed for the training corpus reflect the low cost of these systems; since by using a small corpus size with a small preprocessing, good results can be obtained.
3. The unfactorizing systems need a lower memory size. For example; to build a 2-grams language model for unfactorizing systems, this is corresponding to building $(2*k)$ -grams in the factorizing systems; while (k) is the factorization depth.
4. Although the factorizing systems need a large training corpus size, but the problem of the (OOV) does not exist.
5. Both factorizing systems and unfactorizing ones almost have the same performance when using a large training corpus size.

From the above observations; our recommendation for this problem is to use a hybrid combination of factorizing systems and unfactorizing systems to enjoy the advantages of each of them.

For the problem of Arabic text diacritization; the best strategy to realize usable results is to marry statistical methods with linguistic factorization ones; e.g. morphological analysis. Fully non factorizing statistical methods working on full-form words are faster to learn but suffer from poor coverage (OOV) which can be complemented by linguistic factorization analyzers.

Moreover, there seems to be an asymptotical error margin that cannot be squeezed by the state-of-the-art systems including the presented system in this thesis, especially for the syntactical diacritization without some assistance of a higher-level NLP layer(s); e.g. semantic analysis [1]. After all, syntactic diacritization (case ending) is a projection of a hierarchical grammatical phenomenon that cannot be fully modeled via the statistical inference of linear sequences whatever long is its horizon [6], [23].

The presented hybrid system shows competent error margins with other state-of-the-art systems attacking the same problem. It has a clear plus with morphological diacritization. Moreover, when one account for the sophistication of our training and

test data versus the reported training and test data used with the other systems, some extra credit may be given to ours, especially under realistic conditions.

6.2 Future Work

For the problem of entity factorizing versus unfactorizing; we need to increase the size of the training data of type TRN_DB_I to study the effect of this increase on the factorizing system.

For the problem of automatic Arabic text diacritization:

1. To decrease the errors of types “*Synta_WER_{fac}*” and “*Morpho_WER_{fac}*”, the training set size of type TRN_DB_I should be increased and to be balanced.
2. To decrease the errors of type “*OOV_Morpho_WER_{un-fac}*”, the training set size of type TRN_DB_II should be increased and to be balanced.
3. For the errors of type “*OOV_Synta_WER_{un-fac}*” as discussed above the reason of this type of errors is that the right word is not in the existed analyses in the dictionary; the direct solution for this problem is to increase the training set size of type TRN_DB_II to cover all possible syntactical cases to all words, but actually this is difficult, and we will not be able to cover all possible words in Arabic. So it is suggested to design a system that can generate the possible syntactic diacritics for each word depending on the context that is containing the word, and then choose the most relevant diacritic according to the statistics.
4. Furthermore, it is suggested to add some syntactical linguistic rules and to add a semantic layer to increase the accuracy of both morphological and syntactical diacritizations.

References

The list of References in English

- [1] (Attia, M. et.al., Aug. 2008) M. Attia, M. Rashwan, A. Ragheb, M. Al-Badrashiny, H. Al-Basoumy, S. Abdou, *A Compact Arabic Lexical Semantics Language Resource Based on the Theory of Semantic Fields*, Lecture Notes on Computer Science (LNCS): Advances in Natural Language Processing, Springer-Verlag Berlin Heidelberg; www.SpringerOnline.com, LNCS/LNAI; Vol. No. 5221, Aug. 2008.
- [2] (Attia, M. et.al., 2002) M. Attia, M. Rashwan, G. Khallaaf, *On Stochastic Models, Statistical Disambiguation, and Applications on Arabic NLP Problems*, The Proceedings of the 3rd Conference on Language Engineering; CLE'2002, by the Egyptian Society of Language Engineering (ESoLE); www.ESoLE.org.
- [3] (Attia, M. et.al., 2004) M. Attia, M. Rashwan, *A Large-Scale Arabic POS Tagger Based on a Compact Arabic POS Tags- Set, and Application on the Statistical Inference of Syntactic Diacritics of Arabic Text Words*, Proceedings of the Arabic Language Technologies and Resources Int'l Conference; NEMLAR, Cairo, 2004.
- [4] (Attia, M., 2000) M. Attia, *A Large-Scale Computational Processor of the Arabic Morphology, and Applications*, M.Sc. thesis, Dept. of Computer Engineering, Faculty of Engineering, Cairo University, 2000.
- [5] (Attia, M., Dec. 2004) Attia, M., *Arabic Orthography vs. Arabic OCR*, Multilingual Computing & Technology magazine www.Multilingual.com, Dec. 2004.
- [6] (Attia, M., Sept. 2005) M. Attia, *Theory and Implementation of a Large-Scale Arabic Phonetic Transcriber, and Applications*, PhD thesis, Dept. of Electronics and Electrical Communications, Faculty of Engineering, Cairo University, Sept. 2005.
- [7] (Cavalli-Sforza, V. et.al., 2000) V. Cavalli-Sforza, A. Soudi, T. Mitamura, *Arabic Morphology Generation Using a Concatenative Strategy*, ACM International Conference Proceeding Series; Proceedings of the first conference on North American chapter of the Association for Computational Linguistics (ACL), 2000.
- [8] (Chomsky, N.) A publications list of Noam Chomsky: <http://www.NYbooks.com/Authors/867>.
- [9] (Chomsky, N., 2000) Chomsky, N., *New Horizons in the Study of Language and Mind*, 2000.
- [10] (Dichy, J. et.al., 2005) J. Dichy, M. Hassoun, the DINAR.1 (Dictionnaire INformatisé de l'ARabe, version 1) *Arabic Lexical Recourse, an outline of contents and methodology*, The ELRA news letter, April-June 2005, Vol.10 n.2, France.

- [11] (Fatehy, N., 1995) Fatehy, N., *An Integrated Morphological and Syntactic Arabic Language Processor Based on a Novel Lexicon Search Technique*, master thesis, Faculty of Engineering, Cairo University, 1995.
- [12] (Giménez, J. et.al., 2004) J. Giménez, L. Màrquez, *SVMTool: A General POS Tagging Generator Based on Support Vector Machines*; The proceedings of LREC'04, 2004.
- [13] (Habash, N. et.al., 2005) N. Habash and O. Rambow, *Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop*, Proceedings of ACL'05, 2005.
- [14] (Habash, N. et.al., 2007) N. Habash, O. Rambow, *Arabic Diacritization through Full Morphological Tagging*, Proceedings of the 8th Meeting of the North American Chapter of the Association for Computational Linguistics (ACL); Human Language Technologies Conference (HLT-NAACL), 2007.
- [15] (Jurafsky, D. et.al., 2000) D. Jurafsky, J. H. Martin, *Speech and Language Processing; an Introduction to Natural Language Processing, Computational Linguistics, and Speech Processing*, Prentice Hall, 2000.
- [16] (Katz, S. M., March 1987) S. M. Katz, *Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer*, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-35 no. 3, March 1987.
- [17] (Kirchhoff, K. et.al., Feb. 2008) Kirchhoff, K., Bilmes, J., Duh, K., *Factored Language Models Tutorial*, University of Washington, Department of Electrical Engineering (UWEE), <http://www.ee.washington.edu>, UWEE Technical Report No. UWEETR-2008-0004, Feb. 2008.
- [18] (Maamouri, M. et.al., 2004) Maamouri, M., Bies, A., Buckwalter, T., Mekki, W., *the Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus*. Proceedings of the Arabic Language Technologies and Resources Int'l. Conference; NEMLAR, Cairo 2004. <http://papers.LDC.uPenn.edu/NEMLAR2004/Penn-Arabic-Treebank.pdf>
- [19] (Martin, John C., 1996) John C. Martin *Introduction to Languages and the Theory of Computation*, McGraw Hill (2nd edition), 1996
- [20] (Nadas, N., 1985) Nadas, A., *On Turing's Formula for Word Probabilities*, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-33 no. 6, December 1985.
- [21] (Nilsson, N. J., 1971) N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, 1971.

- [22] (Ratenaparkhi, A., 1998) A. Ratenaparkhi, *Maximum Entropy Models for Natural Language Ambiguity Resolutions*, PhD thesis in Computer and Information Science, Pennsylvania University, 1998.
- [23] (Schütze, H. et.al., 2000) H. Schütze, C. D. Manning, *Foundations of Statistical Natural Language Processing*, the MIT Press, 2000.
- [24] (Stolcke, A., 2002) A. Stolcke, (*SRILM*) *An Extensible Language Modeling Toolkit*, The Proceedings of the International Conference on Spoken Language Processing (ICSLP), 2002.
- [25] (Vapnik, V.N., 1998) Vapnik, V.N., *Statistical Learning Theory*, John Wiley & Sons, Inc., 1998.
- [26] (Yaseen, M. et al., May 2006) M. Yaseen, et al., *Building Annotated Written and Spoken Arabic LR's in NEMLAR Project*, LREC2006 conference <http://www.lrec-conf.org/lrec2006>, Genoa-Italy, May 2006.
- [27] (Zitouni, I. et.al., July 2006) I. Zitouni, J. S. Sorensen & R. Sarikaya, *Maximum Entropy Based Restoration of Arabic Diacritics*, Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL); Workshop on Computational Approaches to Semitic Languages; Sydney-Australia, July 2006; <http://www.ACLweb.org/anthology/P/P06/P06-1073>.

قائمة المراجع العربية

- [28] (الدّالي، عبد العزيز، 1998) الخطاطة (الكتابة العربية)، عبد العزيز الدّالي، مكتبة الخانجي، مصر، 1998م.
- [29] (الراجحي، عبده، 1993) التطبيق الصّري، عبده الراجحي، دار المعرفة الجامعية، الإسكندرية، 1993.
- [30] المعجم الوسيط، مجمع اللغة العربية بالقاهرة، الطبعة الثالثة، 1985م.
- [31] (درمان، مصطفى أغور، 1998) فن الخط العربي؛ مولده وتطوره حتى العصر الحاضر، مصطفى أغور درمان، ترجمة؛ صالح سعادوي، الطبعة الأولى، إستانبول، 1990م.
- [32] (عفيفي، فوزي سالم، 1998) نشأة وتطور الكتابة الخطية العربية ودورها الثقافي والاجتماعي، فوزي سالم عفيفي، الطبعة الأولى، وكالة المطبوعات، الكويت.
- [33] (عمر، أحمد مختار، 1998) دراسة الصوت اللغوي، أحمد مختار عمر، عالم الكتب، مصر، 1990م.