A LARGE-SCALE COMPUTATIONAL PROCESSOR OF THE ARABIC MORPHOLOGY, AND APPLICATIONS

by

Mohamed Attia Mohamed Elaraby Ahmed

A Thesis Submitted to the
Faculty of Engineering, Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
COMPUTER ENGINEERING

Faculty of Engineering, Cairo University
Giza, Egypt
January 2000

A LARGE-SCALE COMPUTATIONAL PROCESSOR OF THE ARABIC MORPHOLOGY, AND APPLICATIONS

by

Mohamed Attia Mohamed Elaraby Ahmed

A Thesis Submitted to the
Faculty of Engineering, Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
COMPUTER ENGINEERING

UNDER THE SUPERVISION OF

Aly Hassan Fahmy
Professor
Faculty of Engineering
Cairo University

Mohsen Abdul Raziq Rashwan
Associate Professor
Faculty of Engineering
Cairo University

Yaseen Al-Shareef
Associate Professor
Faculty of Engineering
Cairo University

Faculty of Engineering, Cairo University
Giza, Egypt
January 2000

A LARGE-SCALE COMPUTATIONAL PROCESSOR OF THE ARABIC MORPHOLOGY, AND APPLICATIONS

by

Mohamed Attia Mohamed Elaraby Ahmed

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

In

COMPUTER ENGINEERING

Approved by
the Examining Committee

Prof. Dr. Ibrahim Faraj

Prof. Dr. Mohammad Jamal Darwish

Prof. Dr. Aly Hassan Fahmy Thesis Supervisor

Dr. Mohsen Abdul Raziq Rashwan Thesis Supervisor

Faculty of Engineering, Cairo University
Giza, Egypt
December 1999

Acknowledgements

I'm greatly indebted to both of the two honorable gentlemen Prof. Dr. Aly H. Fahmy and Dr. Mohsen Rashwan for not only their generous efforts at supervising this thesis, but also for the many good things that I learnt from them inside and outside the college.

Many other gentlemen deserve more than mere thanks for their appreciable roles that follow:

I regard Eng. Nagy Fatehy Mohammad as the grandfather of this thesis for two things. First, his master thesis was the starting point of mine. Second, my direct long discussions with him highly enriched my understanding of the subject.

I proudly consider myself as a student of Dr. Mohammad Afifi, the brilliant researcher at speech recognition, for he taught me a lot of mathematics and sparked me with a lot of ideas that I used here and in other works.

Two experts at the classical Arabic linguistics Yasser Sarhan and Ahmad Atef shared remarkably at this thesis by guiding me through the jungles of literature on classical Arabic linguistics as well as systematically providing to me their precious notes and corrections.

My close friend Galal Khallaf was a real hero that struggled a lot for achieving a respectable edition of this thesis.

Three bright software engineers Mohammad Abdullah, Ibrahim Sobh and Tamer Atef made my morphological processing engine a useful thing by making it the core of a large and integrated system for Arabic text-retrieval and multimedia-book-authory.

I sincerely feel that only Allah can reward all the good guys mentioned above.

Finally, I hope this thesis be a useful addition to the still-poor research activity at the practical Arabic computational linguistics as well as a starting point of more advanced achievements in this field.

List of Contents

List of Figures

Abstract

For a natural language, morphology is the very basic layer over which the higher syntactic and semantic layers are built. So, most of the software systems concerned with the natural text-based human-machine interaction are either direct or indirect candidates of reliable and efficient computational morphological processors. Through out all the significant contemporary natural languages, Arabic has the most elaborate though systematic morphological system. It is hence an excellent candidate of modeling without which any serious computational processing of the Arabic language, with all of its layers, is infeasible.

An industry-quality computational processor of the Arabic morphology − called Morpho3 − along with a host of dependent applications as well as complementary utilities have been implemented. This thesis presents the theoretical basis upon which this morphological processor has been built as well as its domain of applications. Special attention was given to explaining how Morpho3 approaches the main problems that challenge any serious system for processing the Arabic morphology such as coverage, tolerability and ambiguity.

Morpho3 may be regarded as a demonstration of how a rule-oriented knowledge base and a statistical knowledge base can be married towards solving problems in computational linguistics.

Chapter 1

Introduction

1.1 Morphological Processing

The morphology of any natural language is the linguistic system that governs how the words of this language are built.

For a written language, morphology is the most basic layer over which the higher syntactic and semantic layers are built. So, most of the software systems concerned with the natural text-based human-machine interaction are candidates of the computational processing of morphology. Several such systems rely on morphological processing directly such as smart search engines, spell checkers, dictionary binders, high accuracy OCR systems, text-to-speech converters, etc[1]. Some others need morphological processing indirectly, as a basis of higher linguistic layers, such as the natural language interfaces of expert systems.

The morphology of any language is different from that of any other one. In specific, the morphology of Arabic is distinctly sophisticated as well as regular. To produce a practical and reliable computational processor of the Arabic morphology that overcomes such sophistication and utilizes such regularity, one must rely on a clear, simple, coherent and compact mathematical model of the Arabic morphology. Building such a model is the main issue of this thesis.

1.2 Previous Work on the Arabic Morphological Processing

Since the mid 80's, when PC's started to be popular in the Arabic world, the question of "computerizing" Arabic has become a serious one. In spite of the flood of meetings, conferences, literature, scientific societies and even public speeches that one can find all around, one can count only few real initiatives towards the computational processing of the Arabic language. Amazing enough is that some of these few initiatives are majorly led by non-native Arabic speakers!

The actually implemented systems of the computational processing of Arabic are mostly Arabic morphological 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.

---

[1] Chapter 3 is devoted to a detailed discussion of several such software systems.

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. Unfortunately, many significant problems do not manifest themselves unless a real-life large scale system is intended.

On the other hand and driven by the market need for the applications of Arabic morphological processing (specially the need for smart-search engines), the systems of the second category enhanced the theoretical arguments presented by the ones of the first category to produce usable Arabic morphological engines.

As the system presented here belongs to the second category rather than the first one, the two most representative commercial Arabic morphological processors; Sakhr™'s and Xerox™'s are discussed here.

Sakhr's Arabic morphological processor was the best system achieved by native Arabic speakers. Nevertheless, it suffers from the following shortcomings:

1. Each regular derivative root is allowed to combine with a selected set of forms (or patterns) to produce only words that can be found in standard Arabic dictionaries. 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 (like ref.[3] which is regarded as the most elongated the medieval Arabic language). Moreover, the language is a dynamic phenomenon, i.e. an unused possible word at some time may be indispensable at a later time. Also, an unused word at some Arabic country (e.g.: Egypt) may be famous at another Arabic country (e.g.: Syria).
2. The orthography (the writing style) of the Arabic text makes the written words morphologically ambiguous, i.e. each word has multiple possible analyses. The approach of disambiguation is statistical which is powerful in principle. However, the actual implementation of statistical disambiguation at Sakhr suffers from the two following problems:
    i) The statistical database was built from a text corpus once and forever, i.e. the system can not be trained by another text corpus or even be extra trained by extending the primary text corpus.
    ii) Disambiguation 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 the neighboring words. Considering correlation makes statistical correlation far more effective than overlooking it.

3. The morphological model is a heterogeneous one, i.e. it treats regular derivative words in some way, the irregular derivative ones in another way, the fixed words in a yet different one. This heterogeneity is not theoretically sound. More important is that it unnecessarily complicates the applications built over the morphological analyzer.
4. As Sakhr did not publish any technical documents about its Arabic morphological analyzer, no one knows exactly how does its model of Arabic morphology look like. From the experimentation with its morphological analyzer, however, it can be inferred that ad hoc's exist here and there. A lot of ad hoc's causes a lot of subtle problems. This is specially noticed with the conjunction of affixes with the word body (or stem).

On the other hand, Xerox's Arabic morphological processor[1] was the best system implemented by non-native Arabic speakers. Also, it suffers from the following shortcomings:

1. Each regular derivative root is allowed to combine with a selected set of forms (or patterns) to produce only words that can be found in standard Arabic dictionaries. 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.
2. Xerox system has no mechanism of disambiguation!
3. The underlying morphological model is a heterogeneous one, i.e. it treats regular derivative words in some way, the irregular derivative ones in another way, the fixed words in a yet different one. This is the same corresponding shortcoming of Sakhr's system mentioned above.
4. This system is made by non-native Arabic speakers. Moreover, they also selected dictionaries and references on the classical Arabic morphology written by non-native Arabic speakers. Some concepts as well as many fine points are misunderstood or simply overlooked. Also, a significant portion of the morphological entities (roots,

[1] One can find a rich material (history, papers, NLP systems to test, ..., etc.) on Xerox's NLP research reactivities on http:\\www.xrce.xerox.com/research/mltt. Materials specifically on the Arabic language are found on http:\\www.xrce.xerox.com/research/mltt/arabic. Also this material can be requested by paper mail at "Xerox Research Centre Europe, Grenoble Lab., 6, chemin de Maupertuis, 38240 MEYLAN, FRANCE".

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[1].

5. The finite-state based morphological model is an involved one. To transform a written rule from its classical statement to its finite-state form, one needs an automatic tool to do so. Because Arabic does not fit this model perfectly[2], one can not automate the transformation process and have to tediously do so by hand[3]!

1.3 This System "Morpho3"

The system described here is called Morpho3. Morpho, which stands for Morphology, has been developed in RDI labs. Morpho2 and Morpho3, developed by the author of this thesis, can be considered as descendants of Morpho1 developed by Nagy Fatehy Mohammad (See ref.[22]). Morpho3 in its final form has the following advantages over the competing systems:

1. Each regular derivative root is allowed to combine with any form as long as this combination is morphologically allowed. This allows Morpho3 to deal with all the possible Arabic words and removes the need to be tied to a fixed vocabulary as shown in sections [2.5] and [2.6].
2. Morpho3 uses a powerful dynamic *m*-gram statistical disambiguation technique. "Dynamic" means that the statistical knowledge of the system may be altered or adjusted any time to consider any desired text corpus. M-gram means that the system considers the statistical correlation among the words and their neighbors as detailed in chapter 8.
3. The morphological model of Morpho3 is a homogeneous one, i.e. it treats regular derivative words in the same way as the irregular derivative and the fixed ones. This homogeneity is theoretically sound. More important is that it simplifies the applications built over Morpho3.

---

[1] Xerox developers confess that their Arabic morphological analyzer is intended to cover only modern Arabic (called sometimes the Arabic of press).

[2] Xerox developed this technology originally to deal primarily with modern European languages.

[3] Xerox developer K.R.Beesley says that Xerox started its work on the Arabic morphological analyzer in 1991 and ended with a system-under-test in 1997. Several talented people worked on its theory and implementation. The Arabic Morphological Analyzer described by this thesis (Morpho3) took only three years (1997 to 1999) by one man!

4. This system covers almost the whole Arabic morphological phenomenon. It can deal with modern Arabic text as well as ancient Arabic text. It can also deal with Arabic business text, literature and scientific-oriented text as well.
5. The morphological model employed by Morpho3 is a consistent and coherent one that almost contains no ad hoc's. So it is less error prone.
6. Also, this morphological model is a simple compact one. So, scripting the morphological entities in Morpho3 is an easy straightforward task as long as one knows the classical morphological properties of the entities as shown in chapter 5.

1.4 Thesis Overview

In the next chapter, Arabic morphological processing is gradually presented until it is formally defined by the end of the chapter. In chapter 3, several interesting applications are demonstrated. Chapter 4 puts hands on the main three challenges before Arabic morphological processing; namely coverage, tolerability and ambiguity. These three chapters may be regarded together as part one which presents the problem.

Chapter 5 shows how to build a consistent and compact morphological knowledge base. While chapter 6 details the morphological analysis algorithm, chapter 7 details the complementary morphological synthesis process. Finally, chapter 8 describes a powerful statistical technique for disambiguation. These four chapters may also be regarded together as part two which solves the problem.

Finally, chapter 9 presents the conclusions of this thesis and some suggestions for future work. Good reading.

Chapter 2

The Model of the Arabic Word

2.1 The Definition of a *Word*

As morphology is concerned with the analysis of words, it is primary to define the term *word*. Although linguists may enjoy endless discussions about what a word definitely means[1], we fortunately follow a much straightforward and easier approach. As words are in the form of written text, we operationally define a word as it is defined by any text editing program:

A word is the alphanumeric string between any two non-alphanumeric characters.

Example [2.1]: Illustrating the definition of a word.

This short paragraph illustrates the operational definition of the term "word" in Morpho3. Each underline marks a whole word.

2.2 The Definition of an *Arabic Word*

As with the definition of word, Arabic linguists do not agree on what an Arabic word definitely means. Here also, we operationally define an *Arabic word* as:

An Arabic word is a word, as defined above, which meets the following two conditions:

1. All its characters are bare or diacritized Arabic alphabets (Diacritics are introduced in section [2.3].)
2. It belongs to either of the following two categories:
   i) The original Arabic words.
   ii) The Arabized words.

   Original Arabic words are divided in turn into two sub-categories:
   - **Derivative Arabic words**: These are the verbs and nouns that are built according to the Arabic derivation rules[2]. The sweeping majority of the Arabic words belong to this category.
   - **Fixed Arabic words**: These are a set of words molded by Arabs, anciently, and do not obey the Arabic derivation rules. Most of these fixed words are neither verbs nor nouns, most of them are functional words like pronouns, prepositions, conjunctions, question words, and the like. They may be best regarded as the *glue* that ties the words of the Arabic sentence together. The

[1] See for example Ref.[25], ch. 28, pp. 579.

[2] Derivation is detailed in section[2.5].

category of the fixed Arabic words contains a limited number of members (While building our system "Morpho3", we found about 260 significant fixed words only.)

The Arabized words are nouns borrowed from foreign languages (perhaps with some phonetic adjustments to suit the Arabic pronunciation) and have become common among the native Arabic speakers. To preserve the purity of the Arabic language, it is not preferable to consider a word in this category unless its meaning has no counterpart in the category of the original Arabic words.

Figure[2.1] summarizes the definition of the Arabic word.

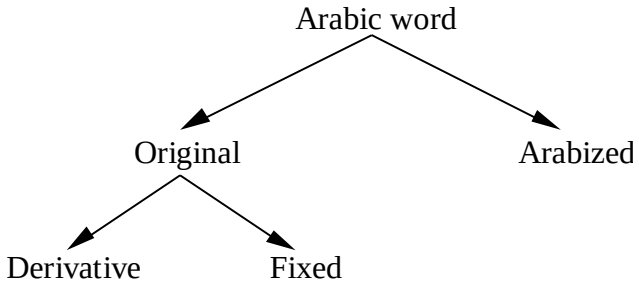

Figure [2.1]: The classification of the Arabic words.

Although the number of the derivative Arabic words is much larger than both the fixed Arabic and the Arabized words, the frequency of the latter ones, specially the fixed words, is considerably high that any treatment of Arabic must treat all the above categories with the same degree of care.

Example [2.2]: The fixed words are frequent in the Arabic text.

هذه الفقرة القصيرة توضِّح أنَّ تكرار الكلمات الجامدة في النَّصِّ العربيِّ مهمٌّ ويُعْتَدُّ به.
وتُمَيَّزُ كلُّ كلمة جامدة في هذه الفِقْرة بخطٍّ تحتها.

Total number of words in the paragraph = 23.
Number of fixed words in the paragraph = 8.
Frequency of fixed Arabic words in the paragraph = 8/23 ≈ 35%

2.3 Arabic is a Diacritized Language

The pronunciation of a word in some languages, like English, is almost always fully determined by its constituting characters. In these languages, the sequence of consonants and vowels determines the correct corresponding voice while pronouncing a word. Such languages are called *non-diacritized* languages.

On the other hand, there are languages, like Latin, where the pronunciation of their words cannot be fully determined by their spelling characters only. In such languages, two different words may have identical spelling whereas their pronunciations and meanings are totally different. To remove this ambiguity, special marks are put above or below the spelling characters to determine the correct pronunciation. These marks are called *diacritics* and the language that uses them is called a *diacritized* language. Arabic is also a diacritized language. In fact, Arabic has the most elaborate diacritization system. Table[2.1] shows the Arabic diacritics and the significance of each one.

Table [2.1]: The Arabic diacritics and the shadda.

Diacritic		Name	Sounds like	Examples	Comments
(1)	عَ	Fateha فتحة	a	مكَافَأة، مَصنَع، بَرَاءَة، عَلَم	ـــــ
(2)	عُ	Damma ضَمَّة	o	كُتُب، هُمُوم، صُراخ، عُود	ـــــ
(3)	عِ	Kasra كسرة	e	كِتاب، مِهنة، عِيال، هِمم	ـــــ
(4)	عْ	Sokoon سُكون	A non vowelized consonant	عوْن، إنْسان، رأْى، استعْمال	ـــــ
(5)	عًا	Tanween fateha تنوين فتحة	Fateha + نْ	كتابًا، نهايةً، طعامًا، ثراءً	Only the last character may be assigned this diacritic
(6)	عٌ	Tanween damma تنوين ضمة	Damma + نْ	حصرٌ، قصورٌ، استعدادٌ، سردٌ	Only the last character may be assigned this diacritic
(7)	عٍ	Tanween kasra تنوين كسرة	Kasra + نْ	مساءٍ، ملاقاةٍ، معانٍ، محامٍ	Only the last character may be assigned this diacritic
(8)	ا ، و ، ى	Vowel مَدّ	Long (a), (e), or (o) vowel	كَاتب، مغَانم، قَال، حمِير، عِيد، طِين، بيُوت، كُوفِئ، رُوح	ـــــ
(9)	ى	Alef leyna أَلِف لَيِّنة	Long (a) vowel	مصطفَى، مَشْتَى، نادَى، مغالَى	Only a terminal ى may be assigned this diacritic.
(10)	ٱ	Bypassed character حرف غير منطوق	Not pronounced	ٱلسَّماء، وٱلسَّماء، قالُوا ٱ، أُولئك	ـــــ
(11)	عٰ	Hidden alef vowel مَدٌّ مستتر بالأَلِف	Long (a)	هٰذا، ذٰلك، الرحمٰن، هٰؤلاء	ـــــ
(12)	عّ	Shadda شَدَّة	معلِّم؛ لِّ = لْ+لِ كُتَّاب؛ تَّ = تْ+تَ حقٌّ؛ قٌّ = قْ+قٌ الصُّبح؛ صُّ = صْ+صُ		In fact, shadda is not a diacritic but is a mark of doubling the character while pronouncing it. The character with a shadda needs another diacritic (from no.1 to no.7) to determine its vowel.

Each character in an Arabic word must be assigned two things about diacritics:

1. The shadda state of the character. (With Shadda/Without Shadda.)
2. The diacritic of the character.

These are called the diacritic information of the character.

Unfortunately, in today Arabic writing, people do not explicitly mention diacritics. They depend on their knowledge of the language and the context to supply the missing diacritics while reading a non-diacritized text. They only mention diacritics in writing when a severe ambiguity is feared or for educational purposes. An automatic morphological analyzer must consider diacritics in its model of Arabic word and must also have some mechanism of figuring out the missing diacritics of a given Arabic word (The mechanism of supplying the missing diacritics is presented in chapter 7.)

We conclude this section with a definition of each of the three diacritization states of an Arabic word:

1. **Full dicaritization**: It is the assignment of all the diacritic information for each character in the word including the last one. In Arabic, the diacritization of the last character sometimes depends on the syntactic analysis of the word within its sentence.
2. **Half diacritization**: It is the same as full diacritization except for that it does not provide the diacritic mark of the last character if it depends on the syntactic analysis of the word. As the morphological analysis deals with words one by one and does not analyze the sentence as a whole, it can only be hoped to provide half diacritization.
3. **Partial diacritization**: Any other diacritization state of the word that provides less diacritic information than half diacritization is called partial diacritization.

Example [2.3]: An example on each of the three diacritization states of the Arabic word.

No diacritization	فأنت ومالك الدنيا سواء	إذا كنت ذا قلب قنوع
Partial diacritization	فأنت ومالِك الدُّنيا سَوَاءُ	إذا كنتَ ذا قلبٍ قنوعٍ
Half diacritization	فَأَنْتَ وَمَالِك ٱلدُّنْيَا سَوَاء	إِذَا كُنْتَ ذَا قَلْب قَنُوع
Full diacritization	فَأَنْتَ وَمَالِكُ ٱلدُّنْيَا سَوَاءُ	إِذَا كُنْتَ ذَا قَلْبٍ قَنُوعٍ

2.4 The Prefix-Body-Suffix Structure of the Arabic Word

In some modern European languages like English, a word may be a verb, a noun, a preposition, an article, ..., etc; i.e., a word in such languages is usually a single entity. In Arabic, the situation is more complex. Example[2.4] shows some simple English sentences and their Arabic translations. It also tabulates the Arabic words and their English counterparts in these statements.

Example [2.4]: An Arabic word may correspond to more than one English word.

Mohammad gave me many things
أعْطاني محمَّدٌ أشياءَ عديدةً

Arabic word	English counterpart
محمَّد	Mohammad
أعْطاني	Gave me
عديدة	Many
أشياء	Things

He returned my book
أعادَ كتابي

Arabic word	English counterpart
أعادَ	He returned
كتابي	My book

I will do it
سأعمله

Arabic word	English counterpart
سأعمله	I will do it

The example above shows that the Arabic word may correspond to a single entity but can as well be compounded of more than one entity. In fact it may be a phrase or even a complete sentence. So, the Arabic word is in general a complex. If we study a sufficiently large sample of Arabic text, we can infer the following general simple structure of the Arabic words[1]:

- The main part, a noun or a verb, of the word occurs in the middle. Let us call this part the word's *body*.
- The body may be prefixed by something like the definitive article, a preposition, a gender determiner, a tense determiner, ..., etc., or some combination of them. When a *prefix* precedes a body, it may slightly modify its string and also be slightly modified. We should note that the prefix cannot be a standalone word.
- The body may also be suffixed by something like a pronoun, a gender determiner, a tense determiner, ..., etc., or some combination of them. When a *suffix* succeeds a body, it may slightly modify its string and also be slightly modified. We should also note that the suffix cannot be a standalone word.

  If the absence of a prefix is assumed as a null prefix and the absence of a suffix is

[1] This structure of the Arabic word is in fact a refined version of that introduced in Ref.[22]

assumed as a null suffix[1], we can then generalize the aforementioned structure of the Arabic word as:

Any Arabic word = Prefix Body Suffix

Example[2.5] shows this structure for the words of an Arabic sentence.

Example [2.5]: The prefix-body-suffix structure of the words of two verses from the Holy Quran.

(أَلَمْ نَشْرَحْ لَكَ صَدْرَكَ، وَوَضَعْنَا عَنْكَ وِزْرَكَ) قرآن كريم

	Arabic word	Prefix	Body	Suffix
1	أَلَمْ	أَ	لَمْ	—
2	نَشْرَحْ	نَـ	شَرَحَ	—
3	لَكَ	—	لَكَ	—
4	صَدْرَكَ	—	صَدْر	كَ
5	وَضَعْنَا	وَ	وَضَعَ	نَا
6	عَنْكَ	—	عَنْ	كَ
7	وِزْرَكَ	—	وِزْر	كَ

From the example above, we note that not only verbs and nouns have the Prefix-Body-Suffix structure but fixed words also do (like the words no.1, no.3, and no.6).

It is important also to note that the terms "prefix" and "suffix" here have different meanings from that in some modern European languages. Here, the prefix or the suffix can add an entity to the noun or the verb. For example the prefix may be a preposition and the suffix may be a pronoun. In modern English, modern French and the like languages, the prefix or the suffix is usually a modifier of the meaning of the noun or the verb but does not add to them another entity.

2.5 The Structure of the Arabic Word's Body

The body part of the Arabic word has in turn an internal structure which is different for the bodies of the derivative Arabic words from that of the non-derivative; fixed and Arabized ones.

Let us start with the derivative bodies. Arabic has a very powerful bodies-generation

[1] The null prefix or suffix is an empty string. While the prefix and/or the suffix of a given Arabic word may be a null, our model does not permit the body to be a null.

mechanism called *derivation* that makes Arabic have the richest vocabulary ever found among all the important contemporary natural languages. Arabic has a relatively small number, about one thousand, of derivative *forms*. Each form is a string of two types of characters; fixed characters (or constants) and 3 or 4 *generic* characters (or variables). Although the form carries one or more specific semantics, it cannot be a word's body on its own. Arabic also has a limited number, about five thousands[1], of derivative *roots*. Each root is a set of 3 or 4 fixed characters. Each root also cannot be a word's body on its own. Nevertheless, the root has one or more standalone specific semantics. The derivation of a word's body from a specific root and a specific form is made by respectively substituting the generic characters of the form with the fixed characters of the root. The derivation process is successful only if each constant character of the root is in the permitted range of the corresponding generic character of the form[2]. The semantic of the resulting word's body is a mix of the semantic of its form and the semantic of its root albeit the latter plays a greater role. Example[2.6] illustrates the derivation process for several derivative word bodies.

Example [2.6]: The derivation process of some regular derivative Arabic word bodies from their forms and roots.

Form	Root	The Resulting Body
$x_3\ x_2$ ا x_1= فَأعِل	ك ت ب	كَاتِب ⇒ x_1=ك ، x_2=ت ، x_3=ب
x_3 ا $x_2\ x_1$ تِ سْ ا= اسْتِفْعَال	ع د د	اسْتِعْدَاد ⇒ x_1=ع ، x_2=د ، x_3=د
$x_4\ x_3\ x_2\ x_1$ مُ= مُفَعْلِل	هـــ ن د س	مُهَنْدِس ⇒ x_1=هـــ ، x_2=ن ، x_3=د ، x_4=س
x_3 ا $x_2\ y_1$ مِ= مِفْعَال	و ل د	مِيلَاد ⇒ y_1= و←ى ، x_2=ل ، x_3=د
$o_3\ x_2$ ا x_1 مُ= مُفَاعِ	ح م ي	مُحَامٍ ⇒ x_1=ح ، x_2 =م ، o_3=↓ى

Each of the forms mentioned above may be regarded as a *general* derivation rule and the bodies made by combining them with roots are called *regular* derivative bodies. However, there is a small set, few hundreds, of derivative bodies that are *irregular*. The form of an irregular derivative body is tailored only for that specific body; i.e., such a form

[1] The most elongated Arabic dictionaries document a larger number (around 8500) of these roots. While building our system we elected the significant 4500 derivative roots and dropped the remaining almost extinct 4000 ones.

[2] In fact, "generic" is a collective word that may refer to a variable character, an altered character or an omitted character. All the generic characters used in our system are explained in table[5.3].

does not have any generic characters and can hence combine with only one specific root to produce one specific body. Irregular derivative bodies were in fact regular derivative ones that have been deviated from its regularity. People then preferred the deviated form that became famous and neglected the regular one which became extinct. Example[2.7] presents some famous irregular derivative word bodies.

Example [2.7]: Some famous irregular derivative bodies

Irregular derivative body	The corresponding (not used) regular body	Root
اللهَ	الْإِلَاٰه	أ ل هـــ
مَلَٰٓئِكُ	مَآلِك	أ ل ك
اتَّخَذَ	ائْتَخَذَ	أ خ ذ
تُهَم	وُهَم	و هـــ م
تَقْوَىٰ	وَقْوَىٰ	و ق ى

On the other hand, the bodies of the fixed Arabic words and the Arabized words have no internal structure because they are molded. The form of such a body is simply a string with all its characters fixed. The form in this case is in fact a body on its own. Now the root plays no role in building the body, nonetheless, we will define it in a way that gives it an important role. One may have several fixed or Arabized words that are semantically or functionally related to one another. In such a case they may be viewed logically as a *family*. If we select the most famous or the most important member of this family to be the root of every member of the family, this will lead to a classification that serves two purposes:

1. It makes the structure of the Arabic word bodies the same for all the Arabic word types.
2. It is invaluable for some applications on the computational Arabic morphological analysis such as smart search (See section [3.4] for adetailed discussion of a smart search.)

   Example[2.8] illustrates this classification for some fixed word bodies.

Example [2.8]: The concept of the root of a family of fixed bodies.

Fixed word bodies	Root of the family
هُوَ، هِيَ، هُمَا، هُمْ، هُنَّ، ...	هُوَ
كُلّ، كِلَا، كِلْتَا، ...	كُلّ
ذَا، هٰذَا، هٰذِهِ، هَاتَانِ، هٰذَانِ، ...	ذَا

2.6 A Unified Formal Description of the Arabic Word

In this section we will summarize the structure of the Arabic word in a compact formal form.

Let us agree on the following symbols:

w = the string of any Arabic word

p = the prefix of *w*

b = the body of *w*

r = the root of *w*

f = the form of *w*

s = the suffix of *w*

Let us assume that *p* and *s* are compatible and the constraints posed by *f* are met by each of *r*, *p* and *s*.[1] Let us also define the following two operators:

1. **The conjunction operator (+)**: The only two operations that can be carried out using this operator are:

$$p + b := p`b`$$

where $p` = p$ after being affected by *b*,

and $b` = b$ after being affected by *p*.

$$p`b` + s := (p`b`)`s`$$

where $(p`b`)` = p`b`$ after being affected by *s*

[1] The match, or compatibility, of a prefix and a suffix within one word as well as the match of a form with a root, a prefix and a suffix is determined by the properties stated in the descriptions of these entities. Describing entities is the subject of chapter 5.

and $s` = s$ after being affected by $p`b`$.

This conjunction operator is neither associative nor commutative.

Now, We can compactly write:

$$w = p + b + s \qquad [2.1]$$

2. **The combination operator (*)**: The only operation that can be carried out using this operator is:

   i. For regular derivative bodies:

   $$r * f := f(r) = b$$

   where $f(r)$ is the string of the form f after substituting its generic characters by the fixed characters of root r.

   ii. For non-derivative and irregular derivative bodies:

   $$r * f := f = b$$

This combination operator is also neither associative nor commutative.

Let us finally assign a precedence of the combination operator higher than that of the conjunction operator.

Using both of the conjunction and the combination operators defined above, we can write:

$$w = p + r * f + s \qquad [2.2]$$

Equation[2.2] summarizes the mechanism by which any Arabic word is built out of its constituting entities: combine the root and the form to get a body, put the prefix in conjunction with the body and finally put the resulting string in conjunction with the suffix. Two important facts are implicitly stated by this equation:

1. The building (or synthesis) mechanism is the same for any Arabic word.
2. The result of the synthesis process is unique, so the quadruple $Q = (t: p, r, f, s)$ constituting any Arabic word is a representation of this word[1].

Note finally that this equation says nothing about how to analyze w into Q. The analysis process is the subject of chapters 6, 7 and 8.

[1] The quadruple is (p, r, f, s) only. t mentions the type of the body and hence the type of the word. The four possible types are Regular_Derivative, Irregular_Derivative, Fixed and Arabized.

2.7 The Function of an Arabic Morphological Analyzer

Analysis means decomposition, so an Arabic morphological analyzer is a software engine that can decompose the words of a given Arabic text into their constituting entities; i.e., it replaces each word *w* in the input text with its quadruple *Q* = (*t: p*, *r*, *f*, *s*). If *w* is not a valid Arabic word, the analyzer flags an error. Our Arabic morphological analyzer "Morpho3" makes more than mere analysis. It reuses the quadruple it has found for each word to resynthesize the word. The resynthesis process supplies the missing diacritics of the Arabic word as well as it corrects common near-miss spelling mistakes. Done computationally, this processing of the Arabic morphology is essential for building several important Arabic software systems as will be explained in the next chapter.

Chapter 3

Applications of the Computational Processing of the Arabic Morphology

Morphological analysis by itself may seem a pure academic philosophy. Through this chapter we are showing that the availability of a computational Arabic morphological analyzer is essential for building many important end-user Arabic software systems. Several such systems are introduced. But first, we shed some lights on the role of text in the modern Arabic multimedia applications.

3.1 Text as an Active Medium in Multimedia Applications

Multimedia applications employ several media such as text, printed documents, sound, images, video, touch, etc. for smoothly transmitting/receiving information to/from the end user. We define an *active medium* in a multimedia application as the one that has all, or at least most of, the following properties:

1. It can be used easily and efficiently for both the output and the input of information to and from the end user.
2. It can be used to efficiently navigate the whole database of the application whatever the media contents of this database are.
3. It can be automatically transformed to the format of other media and vice versa.
4. It can be viewed at different levels of detail, clarity, or explanation according to the instantaneous need of the end user.

For knowledge oriented multimedia applications,[1] if we study the above properties and match the common media to them, we conclude that text, in principle, is graded the most active one. Consequently, this brand of multimedia applications should be built around their text contents. Being the central medium, the Arabic text needs more than the Arabic text editing functions. It needs *enough* linguistic processing to make it truly an active medium.

3.2 Arabic Text Enhancement

Spelling mistakes are common through the text typed on a keyboard, mainly because of mistypos. Automatically scanning such a text by a morphological analyzer reveals a high percentage of such mistakes.

As mentioned in section[2.3], Arabic is a diacritized language. Nevertheless, Arabs

[1] Where the transfer of knowledge not the entertainment or the advertisement is the main issue. Digital libraries, including on-line magazines, computerized books, computerized dictionaries, computerized encyclopedias, etc., are good examples of knowledge oriented multimedia. See ref.[24] for more on Digital Libraries and the role of each medium in them.

nowadays are taking the easy way of writing non-diacritized text and depend upon the linguistic knowledge of the reader that enables him to figure out the correct pronunciation of the words. Due to a lot of cultural and social reasons, the Arabic linguistic knowledge of the new generations is degrading. One way to aid preserving and enhancing the Arabic linguistic knowledge among the public is to automatically convert the commonly found non-diacritized Arabic text into half diacritized text using an Arabic morphological analyzer.

As they increase the correctness, the clarity, and the educability of the Arabic text, both of spell checking and automatic half-diacritization are collectively called Arabic text enhancement.

3.3 On-line Arabic Dictionaries

An exhaustive master dictionary of modern European language–like English is mainly an alphabetically arranged look-up table of words and their explanations for the whole vocabulary of the language. To build a master Arabic dictionary, the previous procedure is impractical! This is due to the structural and derivative nature of Arabic that makes its possible vocabulary very huge in size[1] that no master Arabic dictionary tries to follow the vocabulary look-up table technique. Instead, a traditional master Arabic dictionary builds an alphabetically arranged look-up table of the constituting entities of the Arabic words, especially the roots. This makes the table size within the reasonable range of few thousands of entries. Each entry then leads to the explanation of the root and the usage of its frequently used derivatives. So, to look up an Arabic word, one must have enough knowledge to morphologically analyze the word and get its root. Unfortunately, not a lot of Arabs today – software users are no exception – are skillful enough for such a task. To equip a multimedia application with an on-line Arabic dictionary, an Arabic morphological analyzer is needed to take the word in question and analyze it, on behalf of the user, for extracting its root.

[1] A master dictionary of English like "The Modern Dictionary of Contemporary English" for British English, or "The American Heritage Dictionary" for American English contains at most 650,000 words. On the other hand, given around 250 prefixes, 4500 regular derivative roots, 1000 derivative regular forms, 550 suffixes and assuming the constraints on the conjunction and the combination of these entities, we have around $(250 \cdot 4500 \cdot 1000 \cdot 550)/10 \cong 6 \cdot 10^{10}$ possible Arabic words!

3.4 Smart Text Search

Searching a certain scope of text for a query composed of a given word, a given sequence of characters, or a logical AND/OR combination of words and sequences of characters is a basic and common text editing function. This type of text search is called *exact* search because it is based on the exact matching of the text to the items of the query. Exact search may be helpful with a text written in some modern European languages like English. On the other hand, exact search is very restrictive with a structured and derivative language like Arabic. A user searching exactly for some word in an Arabic text will find only its occurrences where each of its type, prefix, root, form and suffix are the same. This is a very strict condition on matching. In Arabic, when the user is searching for some word, he is intuitively expecting to find the occurrences of the morphologically *relevant* words. The fuzzy word (relevant) may be interpreted according to several possible search *modes* of the word in the query. The most important morphological search modes are:

1. **Exact search**: where the whole *Q* of the word in query must equal the whole *Q* of the occurrences. This is the most strict and hence the weakest mode.
2. **Body-level search**: where only (*t: r*, *f*) of the word in the query must equal (*t: r*, *f*) of the occurrences regardless of the prefix and the suffix.
3. **Root-level search**: where only (*t: r*) of the word in the query must equal the (*t: r*) of the occurrences. This search mode allows finding the occurrences of words that are originated from the same root and probably the same general semantic also[1].

The tolerability and the flexibility provided by the search modes 2 and 3 is the reason why we call morphological search *smart search*. To build an Arabic morphological search engine, an Arabic morphological analyzer is, of course, needed. In fact, the primary reason behind the commercial support of research in the Arabic morphology was the need to provide the large Arabic text databases with smart search facilities. Example[3.1] shows a paragraph as the search scope, some search queries and the results obtained using morphological (or smart) search.

[1] As we mentioned in the first part of section[2.5], the semantic of the root plays the greatest role in the semantic of a regular derivative word.

Example [3.1]: Examples on morphological search.

إِنَّ تَصْنِيفَ أَسَالِيبِ الْكِتَابَةِ إِلَى أُسْلُوبٍ عِلْمِيٍّ جَافٍّ وَأُسْلُوبٍ أَدَبِيٍّ خَصْبٍ لَهُوَ فِي رَأْيِنَا حُكْمٌ تَقْلِيدِيٌّ ضَيِّقُ النَّظَرِ. فَمَا تَتَنَاوَلُهُ الْكِتَابَاتُ الْعِلْمِيَّةُ مِنْ فَلَكٍ أَوْ طَبِيعَةٍ أَوْ هَنْدَسَةٍ وِرَاثِيَّةٍ أَوْ حَتَّى رِيَاضِيَّاتٍ مَلِيءٌ بِالْإِثَارَةِ وَالْجِدَّةِ وَكَثِيرًا مَا يَتَجَاوَزُ حُدُودَ الْخِبْرَةِ الْيَوْمِيَّةِ لِلْقَارِئِ فَلَا يَجِدُ الْكَاتِبُ بُدًّا مِنِ اسْتِخْدَامِ التَّشْبِيهِ وَالِاسْتِعَارَةِ وَالْكِنَايَةِ لِتَقْرِيبِ حَقَائِقِ الْعِلْمِ مِنْ ذِهْنِ الْقَارِئِ، وَهَذَا هُوَ مَا يَدَّعِيهِ الْأُدَبَاءُ حِكْرًا عَلَى كِتَابَاتِهِمْ.

Query Search	Mode	Result
كتابة	Exact-Search	not found
علمى	Exact-Search	(عِلْمِيٍّ)
كتابة	Body-Level-Search	(الْكِتَابَةِ), (الْكِتَابَاتُ), (كِتَابَاتِهِمْ)
علمى	Body-Level-Search	(عِلْمِيٍّ), (الْعِلْمِيَّةُ)
كتابة	Root-Level-Search	(الْكِتَابَةِ), (الْكِتَابَاتُ), (الْكَاتِبُ), (كِتَابَاتِهِمْ),
الأساليب	Root-Level-Search	(أَسَالِيبِ), (أُسْلُوبٍ), (وَأُسْلُوبٍ)

3.5 Support for Arabic Text-to-Speech Systems

Several reasons make automatic Text-to-Speech conversion (TTS) very attractive:

1. **Compression**: There is no need to store huge sound files but only small text ones[1].
2. **Eyes-free reading**: The eyes and the hands are free for another useful activity while the required text is being automatically read. While this feature is invaluable for ordinary people, it is indispensable for the blind.
3. **Hear your favorite announcer**: TTS can synthesize a speech similar to the voice of a certain person provided that it had enough recordings of his speech. So, the voice of the famous people may be provoked for reading any time the user wants.

[1] To store 1 hour of mono sound sampled at 8K Samples/Sec. and quantized into 255 levels (8 bits), a storage space of $(60 \cdot 60 \cdot 8000 \cdot 8)/1024^2 \cong 220$ Mbits $\cong 27.5$ MBytes is needed. Assuming a fast reader uttering 100 Words/Min. and the diacritized word needs 15 bytes in average for storing it as a diacritized text, the text corresponding to 1 hour of fast reading needs $(60 \cdot 100 \cdot 15)/1024^2 \cong 0.086$ MBytes of storage. The compression ratio is hence $(27.5/0.086) \cong 320$.

4. **A reading teacher at your disposal**: Those who are learning reading in some language, specially adults learning a foreign language, may benefit very much from a high-quality TTS in attaining the correct pronunciation of words.

As we mentioned in section[2.3] Arabic is a diacritized language, whereas Arabs often write non-diacritized text which is not enough for an Arabic TTS to figure out the correct pronunciation of words. Adding to the TTS a tool that can supply full diacritization is perfect. Unfortunately, this needs a reliable Arabic syntax analyzer which is not available so far. However, adding an Arabic morphological analyzer that can supply half diacritization only is also acceptable[1].

3.6 Support for Arabic OCR Systems

Optical character recognition OCR (or machine reading) systems are applications of the field of pattern recognition. Although the pattern recognition techniques have become more and more mature, there is no OCR system that has a 100% recognition accuracy. Recognition errors most often occur due to the noise on the printed documents to be recognized. As the noise increases, the accuracy degrades. For Arabic OCR systems, an Arabic morphological analyzer may help as follows:

Some OCR systems are built with a backtracking feature. Given a printed word, the OCR system provides the best interpretation of it from the viewpoint of the pattern recognition technique it uses. If the OCR system is told, by a morphological analyzer, that interpretation is morphologically invalid or extremely improbable, it can backtrack; i.e., provide the second best solution. This process is repeated until a morphologically reasonable interpretation is found. With the companionship of an Arabic morphological analyzer, the Arabic OCR system becomes a self error-correcting one.

Even with non-backtracking OCR systems, the existence of an Arabic morphological analyzer that filters the recognized words into linguistically acceptable and linguistically rejected ones makes the whole system a self error-detecting one. This is an appreciable help for the user while checking the result of recognition.

---

1 Due to their limited knowledge about the Arabic syntax analysis, a lot of Arabs today pronounce the words without the last diacritic even while delivering formal speeches. Although we do not encourage this practice, it has become a widely accepted one.

3.7 Extracting the Morphological Print of an Arabic Writer

Everyone, regardless of his language, has a specific writing style that an intimate friend of him can decide whether a given written passage belongs to him or not. Like fingerprints, iris prints and genetic prints, it is not strange to claim that every one has a *linguistic print*. In fact, many researches have proven the existence of such prints for famous writers and poets of many cultures and languages including Arabic. Researchers have even devised several statistical measures and indices that define such linguistic prints. A lot, in fact most, of the practically feasible, measures and indices depend on the morphological analysis of the text under study (Ref.[7] discusses this topic in detail for the Arabic literature). Many poems and writings having an unknown or doubted identity could be attributed to their actual authors using computational morphological analysis.

3.8 Supporting the Higher Layers of Arabic Linguistic Analysis

The computational processing of any natural language in the aim of making machines attain linguistic capabilities similar to those of humans is a formidable task! In fact, the computational linguistics research is very far from realizing such an aim. To attack the problem, computational linguists usually divide the task into phases or layers. The most agreed-upon scheme of such layers is the one shown in figure[3.1] below (See Ref.[23], ch.15, pp.380.)

As shown in this figure, the lexical (or morphological) analysis occupies the very bottom layer of the scheme. Morphological analysis is hence the basis layer for any higher linguistic processing specially the syntax and the semantic analysis ones.

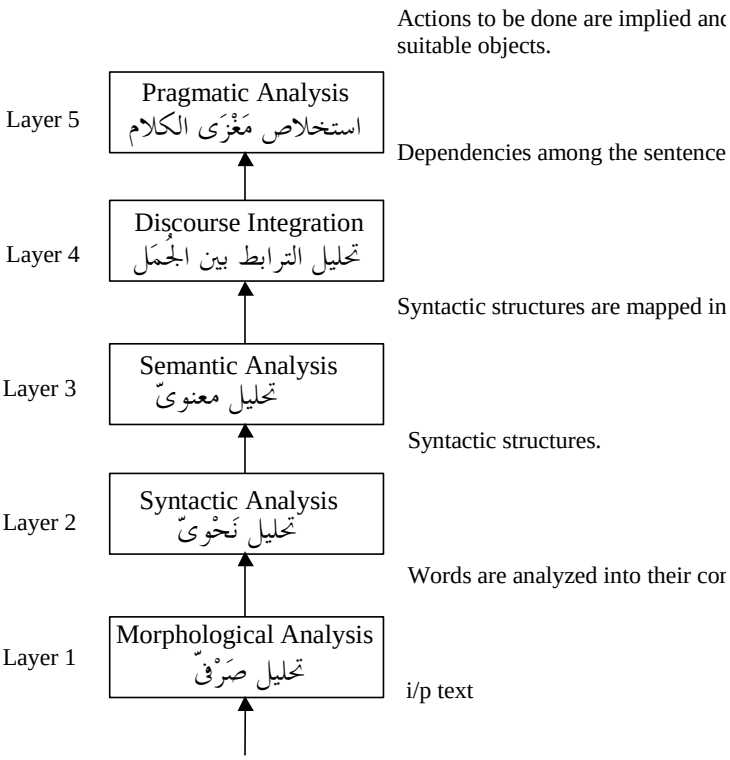

Figure [3.1]: Dividing the processing of a natural language into layers.

Chapter 4

The Main Challenges before the Computational Processing of the Arabic Morphology

We start building our computational Arabic morphological analyzer, "Morpho3", in the next chapter and on. But first in this chapter, we put hands on the main challenges before such a task so that effort is focused towards solving them.

4.1 Coverage

Given the Arabic morphological entities that constitute the Arabic words, and given the conditions that govern the combination of these entities, and according to the Arabic word model presented in section[2.6], the number of possible Arabic words is a finite, though a huge, number[1]. Let us call this number N_T. Assume that we have already built our morphological analyzer and tried it on each of the N_T possible Arabic words to find that it can correctly analyze N words only. We then define the *coverage* of the analyzer to be:

$$c = \frac{N}{N_T} \leq 1 \qquad [4.1]$$

The closer from 1, the better is the morphological analyzer. As N_T is huge, no one is going to compute c using formula[4.1]. To practically compute an approximate value of the coverage of an Arabic computational morphological analyzer, a reasonably large and unbiased text sample containing M_T words is scanned by that analyzer. Assume M words only from the M_T ones are correctly analyzed. Then, the approximate coverage of the analyzer $\tilde{c}$ is computed as:

$$\tilde{c} = \frac{M}{M_T} \leq 1 \qquad [4.2]$$

A system with $\tilde{c}$<98% is classified as an inferior one as it is unable to deal or incorrectly deals with more than 1 word from each 50 ones; i.e., more than 4 faults are committed per the average page.

To compare between two Arabic computational morphological analyzers with respect to their coverage capabilities, repeat the procedure above with both of them and find their uncoverage ratio u_{12} as:

$$u_{12} = \frac{(1-\tilde{c_2})}{(1-\tilde{c_1})} \qquad [4.3]$$

If u_{12}>1, then the coverage of the first analyzer is better u_{12} times than that of the

[1] Given around 250 prefixes, 4500 regular derivative roots, 1000 derivative regular forms, 550 suffixes and assuming the constraints on the conjunction and the combination on these entities, we have $(250 \cdot 4500 \cdot 1000 \cdot 550)/10 \cong 6 \cdot 10^{10}$ possible Arabic words!

second one. If $u_{12}<1$, then the coverage of the second analyzer is better $1/u_{12}$ times than that of the first one.

To have a high coverage, the morphological analyzer cannot try vainly *remembering* the morphological analysis of each of the possible N_T Arabic words. Instead, it should build a reasonably small morphological knowledge base composed of the few thousand *separated* entities from which any Arabic word can be constituted; i.e., the prefixes, the roots, the forms, and the suffixes[1]. The analyzer must also employ some algorithm to analyze, if possible, a given Arabic word into its constituting entities. In the next chapter, we show how the morphological knowledge base of our analyzer is built. In chapter 6, the analysis algorithm is detailed.

4.2 Tolerability

Ideally, the given Arabic text to be morphologically analyzed should have all of its words be half or fully diacritized. Practically, the given Arabic text to be analyzed is expected to contain non-diacritized words, partially diacritized words, words with wrong diacritics and may even contain words with common near-miss spelling mistakes. Such a text is called a *crude* text. Example[4.1] shows a typical crude paragraph taken from an Arabic newspaper.

Example [4.1]: A crude paragraph where common near-miss spelling mistakes and wrongly diacritized words are underlined.

الدوله تهتم بالتوزيع العادل للناتج المحلى

رئيس الوزراء أمام مؤتمر تمويل الإستثمار

النمو الإقتصادى يصل إلى ثلاثة أمثال الزيادة السكانيه

أكد الدكتور "كمال الجنزورى" رئيس مجلس الوزراء ان مصر تنفّذ حاليا برنامجاً متكاملاً للإصلاح الإقتصادى يحقق معدل نمو يصل إلى ثلاثة امثال معدل نمو السكان، مشيراً إلى اشادة مؤسسَّات التقويم الدولية بنجاح البرنامج المصرى الذى حرص على البعد الاجتماعى بتوجيهات مباشرة من السيد رئيس الجمهورية.

The successful morphological analyzer should be able to process both ideal and crude text. Meanwhile, this analyzer should make use of any supplied diacritics to reduce the

[1] Our system employs around 250 prefixes, 4500 regular derivative roots, 1000 derivative regular forms, 300 irregular derivative bodies, 250 fixed bodies, 120 fixed roots, 300 Arabized bodies, 300 Arabized roots, and 550 suffixes. So, the total number of entities in our system = (250 + 4500 + 1000 + 300 + 250 + 120 + 300 + 300 + 550) = 7570 entities. Compare a know-ledge base of 7570 entities to a one of 6×10^{10} words!

morphological ambiguity. Such a morphological analyzer is called a *tolerable* one. Making our Arabic morphological analyzer tolerable is the concern of chapter 7.

4.3 Disambiguation

Ambiguity means having multiple interpretations of the same observation. As we mentioned in section[2.6], the result of synthesizing an Arabic word from its constituting entities is unique. Unfortunately, the same is not true for the analysis process. Example[4.2] shows how a given undiacritized Arabic word can have many possible morphological analyses. As we mentioned in section[4.2], the practically successful Arabic morphological analyzer must deal with both diacritized and non-diacritized text as well.

Example [4.2]: A non-diacritized word may have multiple morphological analyses.

	i/p word: مسترق	Prefix	Root	Form	Suffix
1	مُسْتَرِقّ	—	ر ق ق	مُسْتَفِلّ	—
2	مُسْتَرَقّ	—	ر ق ق	مُسْتَفَلّ	—
3	مُسْتَرِق	—	س ر ق	مُفْتَعِل	—
4	مُسْتَرَق	—	س ر ق	مُفْتَعَل	—
5	مُسْتَرْقٍ	—	ر ق ي	مُسْتَفْعٍ	—

Morphological ambiguity in a word can be decreased, and even sometimes eliminated, if the diacritics of certain characters in it are provided. Example[4.3] shows how the word in the example above can be ambiguity-free after careful partial diacritization. Unfortunately, careful partial diacri-tization needs a highly knowledgeable writer. So, careful partial diacritization is not a frequent clue to rely on.

Example [4.3]: Careful partial diacritization may remove the morpho-logical ambiguity.

i/p word: مسترَقّ	Prefix	Root	Form	Suffix
مُسْتَرَقّ	—	ر ق ق	مُسْتَفَلّ	—

Although the morphological ambiguity is decreased appreciably when the input text is half or fully diacritized, there remain many cases where half and even fully diacritized words are still morphologically ambiguous. Example[4.4] shows how a given fully diacritized Arabic word can have more than one morphological analysis.

Example [4.4]: Even a fully-diacritized word can still be ambiguous.

i/p word: قَائِلٌ	Prefix	Root	Form	Suffix
قَائِل (من القَوْل)	—	ق و ل	فَاعِل	—
قَائِل (من القيلولة)	—	ق ى ل	فَاعِل	—

The examples above show that morphological ambiguity of the Arabic words is an *inherent* problem that can never be ignored. There is no practical solution that can *terminate* the morphological ambiguity in Arabic text but there are three approaches that can greatly *reduce* the ambiguity to the level that makes the morphological analyzer a really helpful tool:

1. **Linguistic approach**: In this approach, the higher syntax and semantic layers, as shown in figure[3.1], should be built and used to reject those morphological analyses that lead to syntactically or semantically meaningless structures.
2. **Statistical approach**: Statistics of the occurrence frequency of each single entity and the occurrence frequency of sequences of entities are collected from a large and unbiased text corpus. When a word occurs within some sentence, these statistics are examined and used to select the most probable morphological analysis of the word.
3. **A mix of the two approaches above**.

The third approach is the best as it is the nearest to the one used by humans. Unfortunately, the first approach, and consequently the third one, is very difficult to achieve. In fact, no one, so far, can claim he has built a good-enough general syntactic or a semantic processor of Arabic. Consequently, the second approach is practically selected for our morphological analyzer. Statistical disambiguation is detailed in chapter 8.

Like the coverage, the disambiguation capability of a computational morphological analyzer can be evaluated by trying it on a reasonably large crude text sample with size of M_T words. Assuming the analyzer can *decide the correct single analysis* for M words only, its disambiguation[1] d can be written as:

$$d = \frac{M}{M_T} \leq 1 \qquad [4.4]$$

[1] The value of d resulting from statistical disambiguation is typically $\geq 95\%$. The ambiguity of the remaining 5% is reduced but still needs manual checking to remove it.

Chapter 5

Building the Morphological Knowledge Base

5.1 The 9 Types of Morphological Entities

According to the Arabic word model presented in chapter 2, we have the following types of entities from which any Arabic word can be constituted:

1. P: Prefixes.
2. R_d: Roots of the derivative words.
3. F_{rd}: Forms of the regular derivative words.
4. F_{id}: Forms of the irregular derivative words.
5. R_f: Roots of the fixed words.
6. F_f: Forms of the fixed words.
7. R_a: Roots of the Arabized words.
8. F_a: Forms of the Arabized words.
9. S: Suffixes.

Figure[5.1] classifies these types of morphological entities.

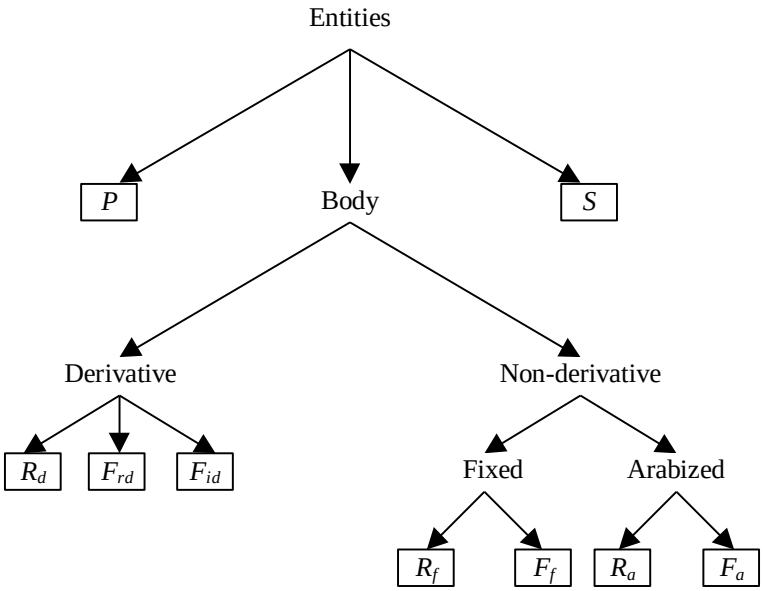

Figure [5.1]: Classifying the 9 types of morphological entities.

5.2 The Two Parts of Entity Description

To build the morphological knowledge base, a computational description must be developed for each of the nine types of entities listed above. How should such an entity description look like in general?

Analogous to the "mechano" bricks, the Arabic morphological entities are in essence word parts that are ready to fit one another in order to construct Arabic words according to the Arabic word model presented in section[2.6]. So, the entity description must in general be composed of the following two main parts:

1. **Isolated part**: This part describes the entity in isolation; i.e., as it exists *alone* before any interaction with other entities.
2. **Interactive part**: This part describes how the entity can interact with other entities, according to the Arabic word model, for building words.

Each of the remaining sections in this chapter details the description of similar types of entities.

5.3 The Description of the *P* and *S* Entities

The isolated part of a prefix or a suffix is composed of the following items:

1. **Spelling**: If common near-miss spelling mistakes are expected, the near-miss spellings are included after the correct one, else, the correct spelling only is registered.
2. **Shadda states**: For each character in the spelling, this item registers whether the character has a shadda (True) or not (False).
3. **Diacritics**: For each character in the spelling, this item registers the diacritic of this character.

The interactive part of a prefix or a suffix is composed of the following items:

1 **The IDP vector**: The elements of this vector are the *identification properties* of the prefix or the suffix. An identification property (IDP) is a code of a prefix morphological functionality class (prefix IDP) or a suffix morphological functionality class (suffix IDP). Our model needs define only 33 prefix IDP's and 52 suffix IDP's[1]. The prefix IDP's form a tree and so do the suffix IDP's. An IDP vector is composed of some terminal IDP's plus all the predecessors of these terminal IDP's without duplication. The prefix IDP's tree and the suffix IDP's tree are illustrated in figure[5.2] and figure[5.3] respectively.

2 **Action**: This item is a code that determines both the analytic and the synthetic mutual effects between the string of the prefix and the string of the body (prefix action) or between the string of the body and the string of the suffix (suffix action). Our model needs define only 20 prefix actions and 26 suffix actions. Due to the space limitations here, only some of the prefix actions and some of the suffix actions are listed in table[5.1] and table[5.2] respectively.

[1] As our morphological knowledge base is composed of the descriptions of the relatively few building entities only rather than the huge Arabic vocabulary, the number of items used for describing these entities must also be few. Otherwise, the entity descriptions quickly grow massive that the whole idea of dealing with only the building entities loses its value.

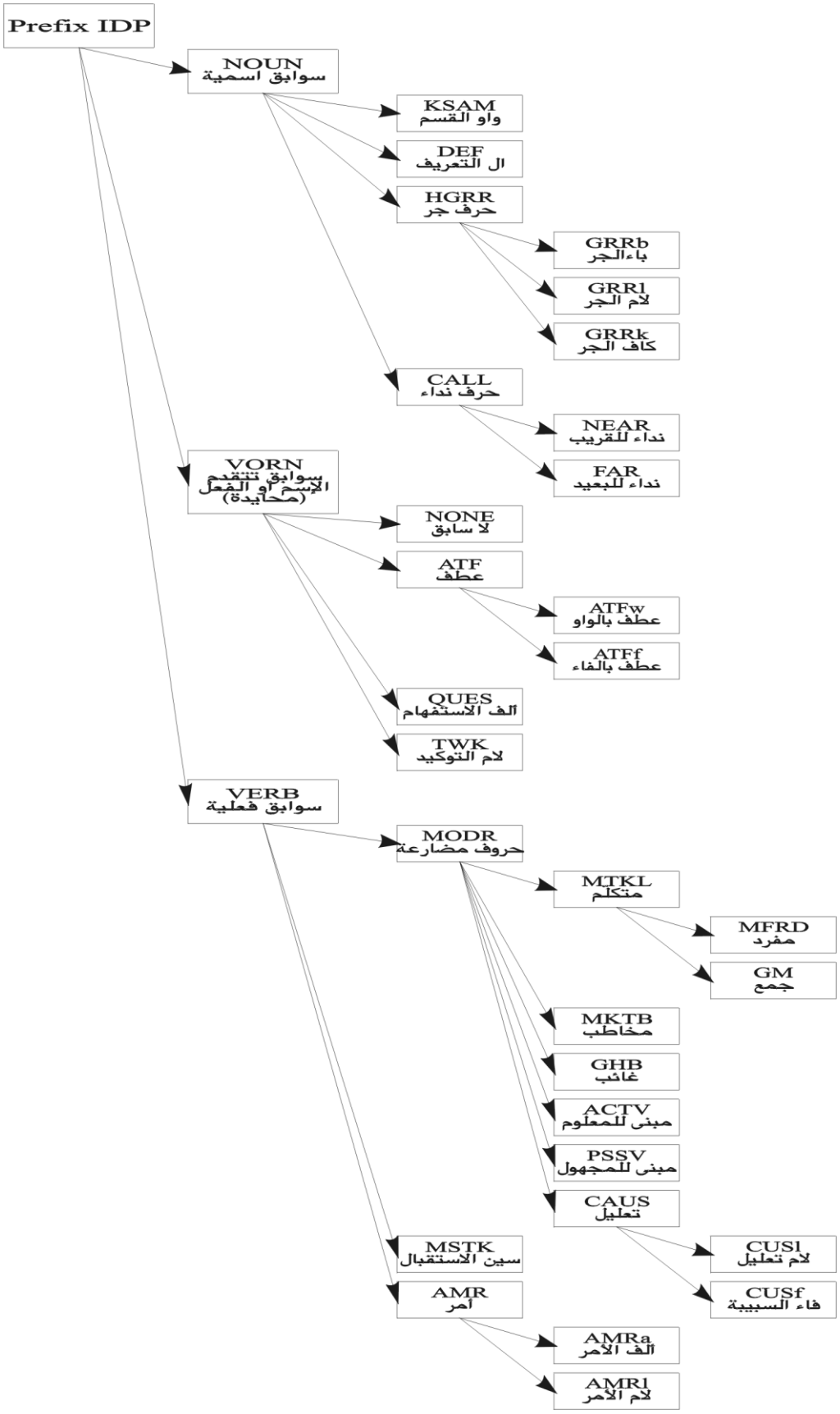

Figure [5.2]: The tree of prefix IDP's

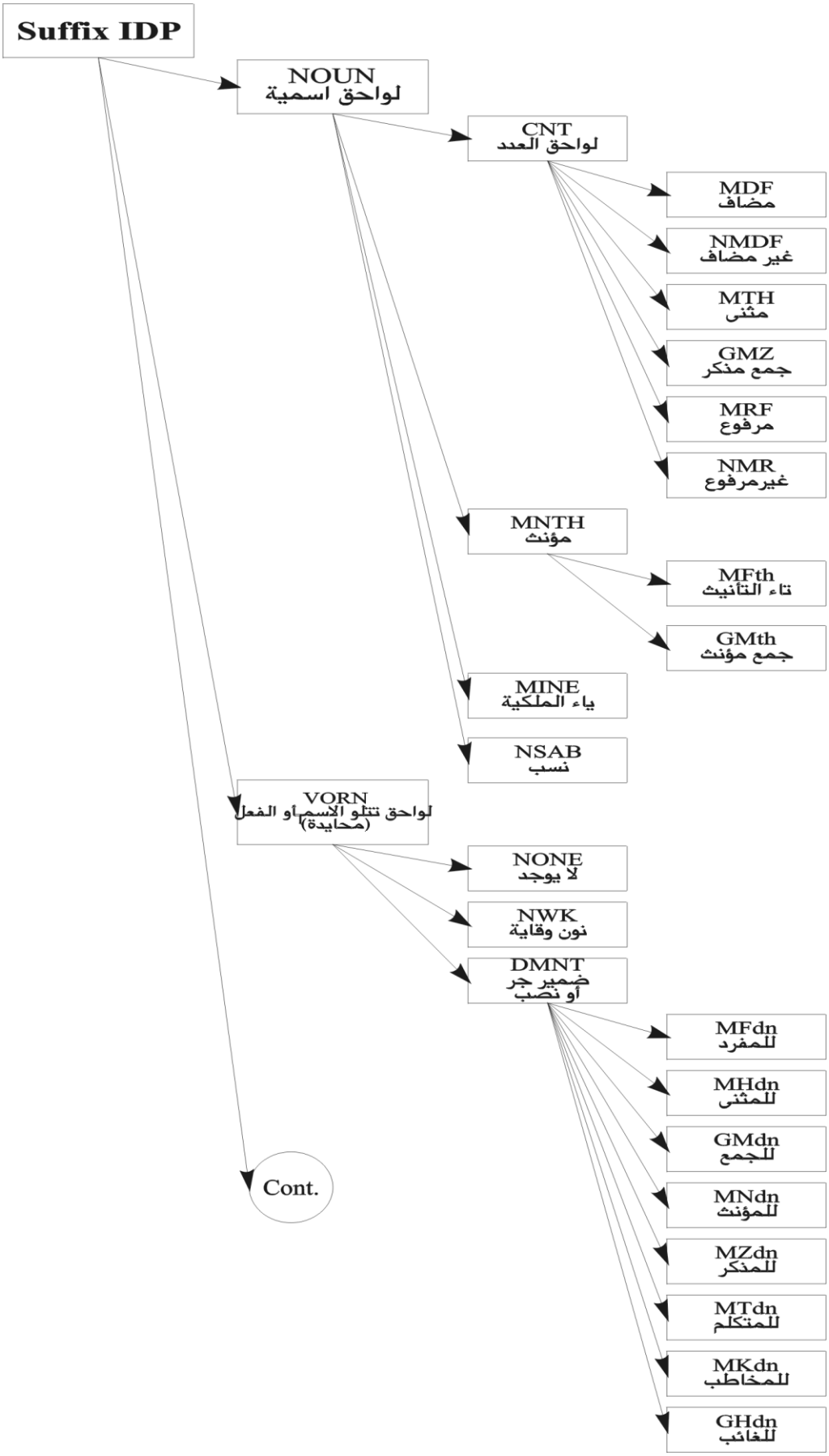

Figure [5.3] The tree of suffix IDP's (cont. on the next page)

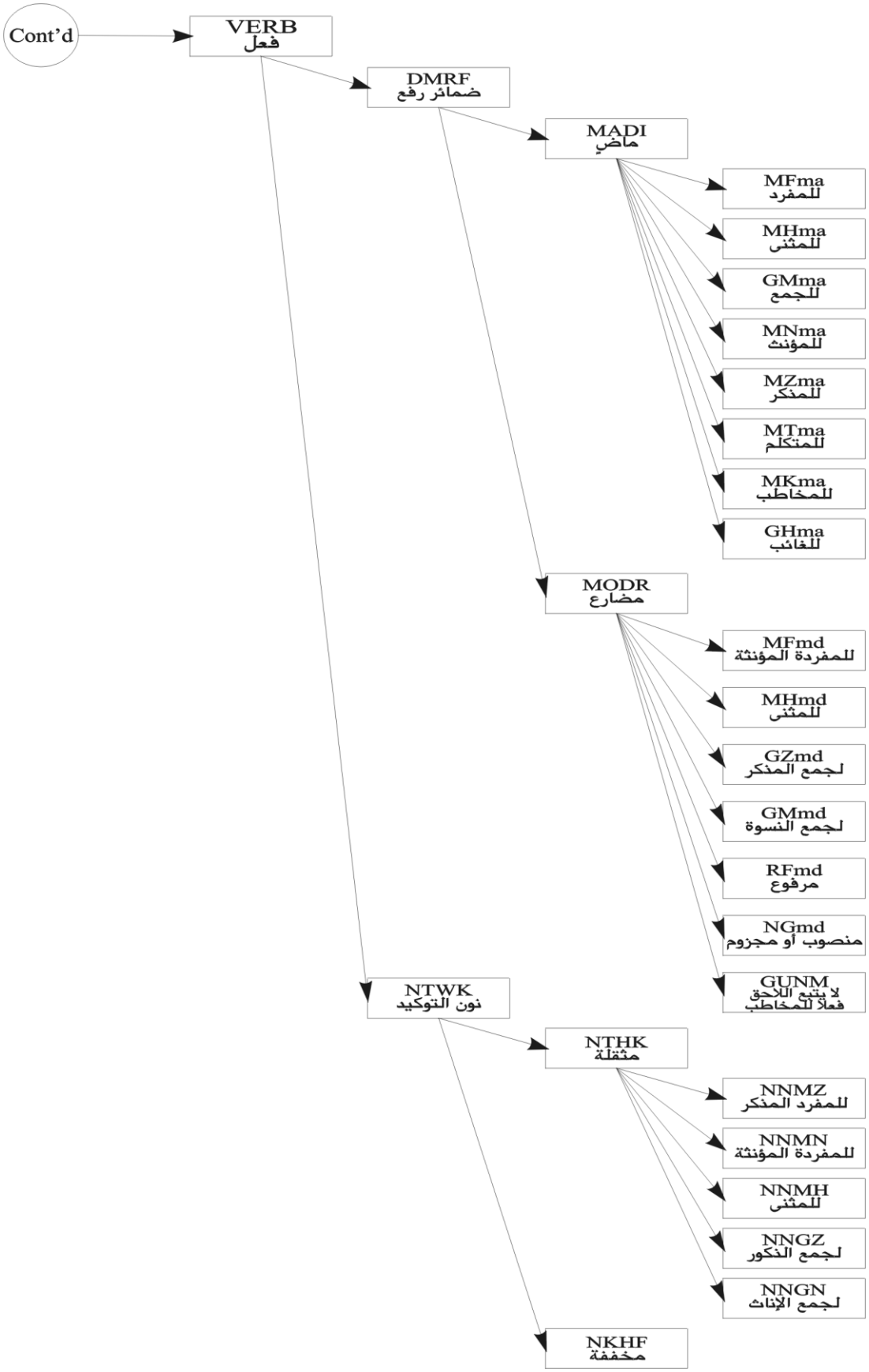

Figure [5.3] The tree of suffix IDP's (cont'd)

Table [5.1]: Some prefix actions.

Prefix Action Code A_p	**The procedure of mutual effects between The prefix $P=(p_1 \dots p_{m-1}\ p_m)$ and The body $B=(b_1\ b_2 \dots b_n)$**		**Examples**
Act_pDEF	I	$P+B=P`B`=(p_1 \dots p_{m-1}\ p_m\ b_1\ b_2 \dots {}^{\alpha}b_n)$ If (α is not a definite diacritic) Then If ($F_p \in$ {MNKS, NMMK})[1] Then ${}^{\alpha}b_n \in \{يَ، يٖ\}$ Else ${}^{\alpha}b_n \in \{b_n, b_n, b_n\}$	P=الـــ B=قُرْآن $P+B$=الْقُرْآن
	II	If ($b_1 \in$ {"ا"}) Then $P+B=P`B`=(p_1 \dots p_{m-1}$ [p_m="ِ"][b_1="ٱ"] $b_2 \dots b_n$)	P=فَٱلـــ B=اِسْتِقْرَار $P+B$=فَٱلِٱسْتِقرَار
	III	If ($b_1 \in$ {ت، ث،د،ذ،ر،ز،س،ش،ص،ض،ط،ظ،ل،ن}) Then $P+B=P`B`=(p_1 \dots p_{m-1}$ [p_m="ْ"] $b_1\ b_2 \dots b_n$) Else $P+B=P`B`=(p_1 \dots p_{m-1}$ [p_m="ْ"] $b_1\ b_2 \dots b_n$)	P=وَٱلـــ B=سَمَاء $P+B$=وَٱلسَّمَاء
Act_pDFGR	I	$P+B=P`B`=(p_1 \dots p_{m-1}\ p_m\ b_1\ b_2 \dots {}^{\alpha}b_n)$ If (α is not a definte diacritic) Then If ($F_p \in$ {MNKS, NMMK}) Then ${}^{\alpha}b_n$="يٖ" Else ${}^{\alpha}b_n=b_n$	P=لِلـــ B=قَاضِيٖ $P+B$=لِلْقَاضِيٖ
	II	If (p_{m-1}=="ل" AND p_m== "ل" AND b_1== "ل") Then $P+B=PB=(p_1 \dots$ [p_{m-1}="ل"] لّ$b_2 \dots b_n$)	P=لِلـــ B=لَيْل $P+B$=لِلَّيْلِ
	III	The same as (II) of Act_pDEF	P=كَٱلـــ B=سَابِق
	IV	The same as (III) of Act_pDEF	$P+B$=كَٱلسَّابِقِ

[1] F_p is the Prefix Action Flag which is a component of the description of the F_{rd}, F_{id} and F_f entities and is hence a component of the body. The descriptions of these entities are detailed in sections[5.4] and [5.6].

Table [5.2]: Some suffix actions.

Suffix Action Code A_s	The procedure of mutual effects between the body $B=(b_1\ b_2\ \dots\ b_n)$ and the suffix $S=(s_1\ \dots\ s_{m-1}\ s_m)$		Examples
Act_sMINE	I	$B+S=B`S`=(b_1\ \dots\ b_{n-2}\ {}^{\alpha}b_{n-1}\ {}^{\beta}b_n\ {}^{\gamma}s_1)$ If (b_n==ي AND b_n has no Shadda AND α≠Sokoon) Then omit ${}^{\beta}b_n$, add Shadda to s_1 , γ=Fateha Else If (b_n== "ا" AND β==Vowel) Then γ=Fateha Else If (β≠Vowel) Then β=Kasra	B=مُحَامِيْ S=ـــيْ $B+S$=مُحَامِيَّ B=رِضَاْ S=ـــيْ $B+S$=رِضَاْيَ B=كِتَاْب S=ـــيْ $B+S$=كِتَاْبِيْ
Act_sDMMA	I	$B+S=B`S`=(b_1\ \dots\ b_{n-1}\ {}^{\alpha}b_n\ {}^{\beta}s_1\ s_2\ \dots\ s_m)$ If (F_s==MNFT)[1] Then If (s_1 has no Shadda) Then β=Sokoon Else α=Damma	B=الْعَاْمِل S=ونَ $B+S$=الْعَاْمِلُوْنَ B= مُصْطَفَىٰ S=ونَ $B+S$= مُصْطَفَوْنَ

Act_sMDWW	I	$B+S=B`S`=(b_1\ \dots\ b_{n-1}\ {}^{\alpha}b_n\ {}^{\beta}s_1\ s_2\ \dots\ s_m)$
		If (F_s==VWYA OR F_s==VYW2 OR F_s==VERB)
		Then α=Damma, β=Vowel
		If (F_s==VAA OR F_s==VAYW)
		Then α=Fateha, β=Sokoon
			B=عَمِلَ
			S=واْ
			$B+S$=عَمِلُوْاْ
			B= نَسِيَ
			S=واْ
			$B+S$= نَسُوْاْ
			B=سَعَىٰ
			S=واْ
			$B+S$= سَعَوْاْ

Example[5.1] shows the description of some prefixes and suffixes.

Example [5.1.a]: The description of some prefixes.

Isolated part	Prefix IDP Vector	Prefix

[1] F_s is the Suffix Action Flag which is a component of the description of the F_{rd}, F_{id} and F_f entities and is hence a component of the body. The descriptions of these entities are detailed in sections[5.4] and [5.6].

		Action
الـــ like in الْـــقُرْآن	[NOUN, DEF]	Act_pDEF
بالـــ like in بِالْقُرْآنِ	[NOUN, HGRR, GRRb, DEF]	Act_pDFGR
نـــ like in نَـــتَعَلَّمُهُ	[VERB, MODR, ACTV, MTKL, GM]	Act_pACTV
سَنـــ like in سَنَـــعْرِفُهُ	[VERB, MSTK, MODR, ACTV, MTKL, GM]	Act_pTVMS

Example [5.1.b]:The description of some suffixes.

Isolated part	**Suffix IDP Vector**	**Suffix Action**
ـــيْ like in أَخْلَاقِيْ	[NOUN, MINE]	Act_sMINE
ـــة like in مُتَعَلِّمَـــة	[NOUN, MNTH, MFth]	Act_sMFth
ـــتهَا like in إِصْدَارَتُـــهَا	[NOUN, MNTH, MFth, DMNT, MFdn, GHdn, MNdn]	Act_sMNDN
ـــوْنَ like in الْعَامِلُـــوْنَ	[NOUN, CNT, GMZ, NMDF, MRF]	Act_sDMMA
ـــواْ like in تَعَلَّمُـــوْاْ	[VERB, DMRF, MADI, GHma, GMma, MZmd]	Act_sMDWW

5.4 The Description of the F_{rd} Entities

The isolated part of a F_{rd} is composed of the following items:

1. **The spelling**: For each character in the form, this item registers either of the following:
   a) An ordinary, constant, alphabetical letter.
   b) A generic letter. Table[5.3] shows the generic characters used for representing the F_{rd} entities as well as their semantics.

      If common near-miss spelling mistakes are expected, the near-miss spellings are included after the correct one. Otherwise, the correct spelling only is registered.

Table [5.3]: The generic characters.

N.B: The following notation is considered in this table:

1. The generic character, of order k within the root, is registered at location $i[k]$ within the body.
2. The set $G_{i[k]}$ is the corresponding range of the generic character.
3. A is the set of all the Arabic alphabetical characters.

Generic Character	Semantic	Example
x_k $1 \le k \le 4$ حَرْفٌ سالِمٌ لا يَمْنَعُ التَّضعيفَ	$x_k = c_{i[k]}$; $c_{i[k]} \in A\text{-}G_{i[k]}$	مَفْعُوْل مَكْتُوْب م x_1 x_2 و x_3
z_k $2 \le k \le 4$ حَرْفٌ سالِمٌ يَمْنَعُ التَّضعيفَ	$z_k = c_{i[k]}$; ($c_{i[k]} \in A\text{-}G_{i[k]}$) AND (($c_{i[1]} \neq c_{i[3]}$ AND $c_{i[2]} \neq c_{i[4]}$ for quad roots) OR ($c_{i[2]} \neq c_{i[3]}$ for tri roots))	مُفْعَلِلّ مُطْمَئِنّ م z_1 x_2 z_3 z_4
d_k $2 \le k \le 3$ حَرْفٌ مضعَّفٌ مَفْكُوك	$d_k = c_{i[k]}$; ($c_{i[k]} \in A\text{-}G_{i[k]}$) AND ($c_{i[2]} == c_{i[3]}$)	فَلَلْـــتُ رَدَدْتُ x_1 d_2 d_3
$d_{2\&3}$ حَرْفٌ مُضَعَّفٌ مُدْغَمٌ	$d_{2\&3} = c_{i[2\&3]}$; $c_{i[2\&3]} \in A\text{-}G_{i[2\&3]}$	فالّ حارٌّ x_1 ا $d_{2\&3}$
y_k $1 \le k \le 3$ حَرْفٌ مُعَلٌّ	$y_k = c_{i[k]}$; $c_{i[k]} \in G_{i[k]} - \{G_{i[k]}[0]\}$ $G_{i[k]}[0]$ is the original character of the root and is replaced by any one of $G_{i[k]} - \{G_{i[k]}[0]\}$	مِفْعَال مِيْلاد م y_1 x_2 ا x_3 و←ي
o_k $1 \le k \le 3$ حَرْفٌ مَحْذُوف	The original character $G_{i[k]}[0]$ is omitted. $G_{i[k]} = \{G_{i[k]}[0]\}$	إفَالَة إعَارَة إ x_1 o_2 ا x_3 ـــة و↓

2. **Shadda states:** For each character in the form, this item registers whether the character has a shadda (True) or not (False).
3. **Diacritics:** For each character in the form, this item registers the diacritic of this character.
4. **The character ranges:** For each character in the form:
   a) If the corresponding spelling letter is a constant one, a *don't-care* or CONS code is registered.
   b) If the corresponding spelling letter is a generic one, the code of its *range* is registered.

   For each generic character mentioned in table[5.3], one example of its possible ranges is shown in table[5.4].

Table [5.4]: Some possible ranges of the generic characters.

Generic character	One possible range
x_k	$G_{i[k]}$ = WYHx = {و،ى،أ} ⇒ $x_k \notin$ {و،ى،أ}
z_k	$G_{i[k]}$ = WYx = {و،ى} ⇒ $z_k \notin$ {و،ى}
d_k	$G_{i[k]}$ = WYx = {و،ى} ⇒ $d_k \notin$ {و،ى}
$d_{2\&3}$	$G_{i[2\&3]}$ = WYHx = {و،ى،أ} ⇒ $d_{2\&3} \notin$ {و،ى،أ}
y_k	$G_{i[k]}$ = WHy = {و،أ} ⇒ y_k = و→أ Original letter (و here) at position $i[k]$ within the body and position k within the root is transformed into (أ).
o_k	$G_{i[k]}$ = Wo = {و} ⇒ o_k = و ↓ Original letter (و here) at position $i[k]$ within the body and position k within the root is omitted.

The interactive part of a F_{rd} is composed of the following items:

1. **Relations:** A relation is composed in turn of the following sub-items:
   a) **A prefix must-have IDP**: If not null, then a prefix is permitted to precede the form[1] only if the IDP vector of the prefix includes[2] the (prefix must-have IDP).
   b) **Prefix may-have IDP vector**: A prefix is not permitted to precede the form, if any IDP from the prefix IDP vector is not *mutually inclusive*[3] with any IDP from the (prefix may-have IDP vector).
   c) **A suffix must-have IDP**: If not null, then a suffix is permitted to succeed the form only if the IDP vector of the suffix includes the (suffix must-have IDP).
   d) **Suffix may-have IDP vector**: A suffix is not permitted to succeed the form if any IDP from the suffix IDP vector is not mutually inclusive with any IDP from the (suffix may-have IDP vector).

A prefix, a suffix and a form are a match if the IDP vector of the prefix matches the IDP vector of the suffix[4], and both of them satisfy the form's relation. Some forms can be described by one relation whereas others need more than one. In case of more than one relation, at least one of them must be satisfied by the IDP vectors of the prefix and the suffix. Otherwise, the prefix and the suffix do not fit the form. In section[6.3] and [6.7], the Boolean formulae[6.1] through [6.3] compact the prefix-body-suffix match verbally expressed above.

2. **Prefix action flag:** A prefix action flag is a code of the morphological class of the form when preceded by a prefix. The prefix action of the prefix that precedes a form uses this flag to tune the mutual analytic or synthetic effect between the string of the prefix and the string of the form. Table[5.5] lists some prefix action flags and their semantics.

---

1 Or more precisely "precede the body produced by the combination of this form and a suitable root".

2 An IDP vector includes a given IDP if at least one IDP in the vector is a descendant of the given IDP.

3 Two IDP's are mutually inclusive if one of them is a descendant of the other.

4 If one IDP in the prefix IDP vector is not compatible with one IDP in the suffix IDP vector, the prefix IDP vector *does not match* the suffix IDP vector.

Table [5.5]: Some prefix action flags.

Prefix Action Flag	Semantic
pbf NMSR	اسْمٌ مُصَرَّفٌ (يُنَوَّنُ ويُجَرُّ بالكَسْرَةِ فى حَالَتَيْ الإفْرادِ والتَّنْكير)
pbf NMMN	اسمٌ مَمْنوعٌ مِن الصَّرْفِ (لا يُنَوَّنُ ويُجَرُّ بالفَتْحَةِ في حالَةِ الإفْرادِ والتَّنْكير)
pbf ELSE	حَرَكَةُ آخِرِ حَرْفٍ مِن الكَلِمَةِ ثابِتَةٌ لا تَتَأَثَّرُ بِأيِّ سابِقٍ.

3. **Suffix action flag:** A suffix action flag is a code of the morphological class of the form when succeeded by a suffix. The suffix action of the suffix that succeeds a form uses this flag to tune the mutual analytic or synthetic effect between the string of the form and the string of the suffix. Table[5.6] lists the suffix action flags and their semantics.

Table [5.6]: Some suffix action flags.

Suffix Action Flag	Semantic
bsf MNFT	اسمٌ مَقْصورٌ يَكُونُ مفتوحَ الآخِرِ عِنْدَ إلْحاقِهِ بِجَمْعِ المُذَكَّرِ ويُسَكَّنُ واوَ الجَماعَةِ بَعْدَه.
bsf NA	فِعْلٌ مِثالٌ واوىٌّ (يَبْدَأُ بالواو) يُحْذَفُ أَوَّلُهُ ويُفْتَحُ الحَرْفُ التَّالي فِي الزَّمَنِ المُضارِع.
bsf ELSE	حَرَكَةُ آخِرِ حَرْفٍ من الكَلِمَةِ ثابِتَةٌ لا تَتَأَثَّرُ بِأَيِّ لاحِقٍ.

Example[5.2] shows the description of some F_{rd}'s.

Example [5.2]: The description of some F_{rd} entities.

	Description of F_{rd} entity		**Examples**
مُفْعَلِلّ	Spell1:	(MEEM, x1, x2, z3, z4)	ش م أ ز ← مُشْمَئِزّ
	Shadda:	(FALSE, FALSE, FALSE, FALSE, TRUE)	ط م أ ن ← مُطْمَئِنّ
	Diacritics:	(DMMA, SOKOON, FATEHA, KASRA, UNDT)	ك ف هـ ر ← مُكْفَهِرّ
	Ranges:	(CONS, NNBS, NNBS, NNBS, NNBS)	
	Prefix must have IDP:	null	
	Prefix may have IDP vector:	[VORN, NOUN].	
	Suffix must have IDP:	null	
	Suffix may have IDP:	[NONE, DMNT, NOUN].	
	Prefix action flag:	NMSR	
	Suffix action flag:	NOUN	
اسْتِفْعَالْ	Spell1:	(ALEF, SEEN, TAA, x1, x2, ALEF, x3)	خ ر ج ← اسْتِخْرَاْج
	Spell2:	(HMZE, SEEN, TAA, x1, x2, ALEF, x3)	ع م ل ← اسْتِعْمَاْل
	Shadda:	(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE)	غ ل ل ← اسْتِغْلَاْل
	Diacritics:	(BPKS, SOKOON, KASRA, SOKOON, FATEHA, VWL, UNDT)	
	Ranges:	(CONS, CONS, CONS, WYx, NNBS, CONS, WYx)	
	Prefix must have IDP:	null	
	Prefix may have IDP vector:	[VORN, KSAM, CALL, HGRR]	
	Suffix must have IDP:	null	
	Suffix may have IDP vector:	[NONE, DMNT, MDF, NMDF, MTH, MRF, NMR, MNTH, MINE]	
	Prefix must have IDP:	DEF	
	Prefix may have IDP vector:	[VORN, NOUN]	
	Suffix must have IDP:	null	
	Suffix may have IDP vector:	[NONE, NMDF, MTH, MRF, NMR, MNTH]	
	Prefix must have IDP:	null	
	Prefix may have IDP vector:	[VORN, NOUN]	
	Sufffix must have IDP:	NSAB	
	Suffix may have IDP vector:	[DMNT, NOUN]	
	Prefix action flag:	NMSR	
	Suffix action flag:	NOUN	

5.5 The Description of the R_d Entities

The isolated part of a R_d contains a single item only:

- **Spelling:** The spelling of a R_d is composed of either three (triple R_d) or four (quad R_d) alphabetical letters.

The interactive part of a R_d also contains a single item only:

- **"Ella" flag**[1]: Many triple R_d's have one or more "Ella" characters. Although the constant characters of such R_d's may obey the corresponding ranges of the generic characters of a given F_{rd}, the Ella flag may ban the combination of such a R_d and the given F_{rd}. Our system needs define only 10 Ella flags, some of which are explained in table[5.7].

Table [5.7]: Some Ella flags.

"Ella" flag	Explanation	like in	
VERB	فِعْلٌ سالِمٌ (لا يَحْتَوِي عَلَى حُروفِ عِلَّةٍ)	عَلِمَ يَعْلَم	ع ل م
VLAA	فِعْلٌ مُعْتَلُّ الآخِرِ بالأَلِفِ فِي الماضِي وفِي المُضارِعِ الْمُجَرَّدَيْنِ.	سَعَىٰ يَسْعَىٰ	س ع ى
VLAWY	فِعْلٌ مُعْتَلُّ الآخِرِ بالأَلِفِ فِي الماضي وبالواوِ أو الياءِ (حَسَبَ الحَرْفِ الأَخيرِ في الجِذْرِ) في المُضارِعِ الْمُجَرَّدِ.	مَشَىٰ يَمْشِي	م ش ى
VMAA	فِعْلٌ مُعْتَلُّ الوَسَطِ بالأَلِفِ فِي الماضي وفِي المُضارِعِ الْمُجَرَّدَيْنِ.	غَارَ يَغَارُ	غ ى ر
VMWY	فِعْلٌ مُعْتَلُّ الوَسَطِ بالأَلِفِ فِي الماضِي وبالواوِ أو بالياءِ (حَسَبَ الحَرْفِ الأَوْسَطِ في الجِذْرِ) في المُضارِعِ الْمُجَرَّدِ.	عَادَ يَعُودُ	ع و د

Example[5.3] shows the description of some R_d's.

[1] In the Arabic morphology, "Ella" means the replacement of an original vowel character by another one in order to make the resulting word phonetically plausible.

Example [5.3]: The description of some R_d entities.

Spelling	(RAA, FAA, DDAD) (ر ف ض)
Ella flag	VERB
Spelling	(EIN, RAA, FAA) (ع ر ف)
Ella flag	VERB
Spelling	(TTAA, YAA, RAA) (ط ى ر)
Ella flag	VMWY
Spelling	(EIN, LAM, WAW) (ع ل و)
Ella flag	VLAWY
Spelling	(WAW, HA, BAA) (و هـــ ب)
Ella flag	VFFT

5.6 The Description of the F_{id}, F_f, and F_a Entities

The isolated part of a F_{id}, F_f or F_a entity is composed of the following terms:

1. Spelling: As that of the F_{rd} entities but with no generic characters.
2. Shadda states: As that of the F_{rd} entities.
3. Diacritics: As that of the F_{rd} entities.
4. Root spelling: As these entities do not have a built-in derivation mechanism that enables the analyzer to automatically extract the root (whether is a R_d, R_f or R_a) from the body, the root spelling must be explicitly mentioned in the description.

The interactive part of a F_{id}, F_f or F_a entity is composed of the following items:

1. Relations: As that of the F_{rd} entities.
2. Prefix action flag: As that of the F_{rd} entities.
3. Suffix action flag: As that of the F_{rd} entities.

Example[5.4] shows the description of one F_{id}, one F_f and one F_a.

Example [5.4]: The description of one F_{id}, one F_f, and one F_a entities.

Type of entity	Description of the entity	
F_{id}	Spell:	(TAA, ALEF, RAA, YAA, KHAA) (تَأْرِيخ)
	Shadda:	(FALSE, FALSE, FALSE, FALSE, FALSE)
	Diacritics:	(FATEHA, VOWEL, KASRA, VOWEL, UNDT)
	Root:	(HMZA, RAA, KHAA) (أ ر خ)
	Prefix must have IDP:	Null.
	Prefix may have IDP vector:	[VORN, KSAM, CALL, HGRR]
	Suffix must have IDP:	Null.
	Suffix may have IDP vector:	[NONE, DMNT, MDF, NMDF, MTH, MRF, NMR, MINE]
	Prefix must have IDP:	DEF
	Prefix may have IDP vector:	[VORN, NOUN]
	Suffix must have IDP:	Null.
	Suffix may have IDP vector:	[NONE, NMDF, MTH, MRF, NMR]
	Prefix must have IDP:	Null.
	Prefix may have IDP vector:	[VORN, NOUN]
	Suffix must have IDP:	NSAB.
	Suffix may have IDP vector:	[DMNT, NOUN]
	Prefix action flag:	NMSR.
	Suffix action flag:	NOUN.
F_f	Spell:	(HA, ZAL, ALEF) (هٰذَا)
	Shadda:	(FALSE, FALSE, FALSE)
	Diacritics:	(Hidden Alef vowel, FATEHA, VOWEL)
	Root:	(ZAL, ALEF) (ذَا)
	Prefix must have IDP:	Null.
	Prefix may have IDP vector:	[VORN, KSAM, HGRR]
	Suffix must have IDP:	Null.
	Suffix may have IDP vector:	[NONE]
	Prefix action flag:	ELSE.
	Suffix action flag:	ELSE.

Example [5.4]: The description of one F_{id}, one F_f, and one F_a entities. (cont'd)

Type of entity	Description of the entity	
F_a	Spell:	(GEEM,, HA, NOON, MEEM) (جَهَنَّم)
	Shadda:	(FALSE, FALSE, TRUE, FALSE)
	Diacritics:	(FATEHA, FATEHA, FATEHA, UNDT)
	Root:	(GEEM, HA, NOON, MEEM) (جهنم)
	Prefix must have IDP:	null
	Prefix may have IDP vector:	[VORN, HGRR, KSAM]
	Suffix must have IDP:	null.
	Suffix may have IDP vector:	[NONE]
	Prefix must have IDP:	null
	Prefix may have IDP vector:	[VORN, NOUN]
	Suffix must have IDP:	NSAB.
	Suffix may have IDP vector:	[DMNT, NOUN]
	Prefix action flag:	NMMN.
	Suffix action flag:	NOUN.

5.7 The Description of the R_f and R_a Entities

The isolated part of a R_f or R_a entity is composed of the following items:

1. Spelling: Only the correct spelling of the entity is stated here.
2. Shadda states: As that of the F_{rd} entities.
3. Diacritics: As that of the F_{rd} entities.

No interactive part is needed for these types of entities.

Example[5.5] shows the description of one R_f and one R_a.

Example [5.5]: The description of one R_f and one R_a entities.

Type of entity	Description of the entity	
R_f	Spell:	(ZAL, ALEF) (ذَا)
	Shadda:	(FALSE, FALSE)
	Diacritics:	(FATEHA, VOWEL)
R_a	Spell:	(GEEM, HA, NOON, MEEM) (جَهَنَّم)
	Shadda:	(FALSE, FALSE, TRUE, FALSE)
	Diacritics:	(FATEHA, FATEHA, FATEHA, FATEHA)

5.8 Notes on the Authory of the Morphological Entities

Through out this thesis we are working out a computational *model* of the Arabic morphology not the Arabic morphology itself. Books about the Arabic morphology written by the traditional Arabic linguists are presenting the Arabic morphology to ordinary human students and are not concerned at all with computational modeling. These books are speaking about neither morphological entities, isolated part, interactive part, IDP's, actions, etc, nor about a lot of the terms we have coined to describe our computational model.

To build the morphological knowledge base, we had to read, understand, filter and then convert the contents of these traditional books on the Arabic morphology (See references from [1] to [15]) into the form required by our model. Both the filtering and the conversion processes are delicate ones because they very often require selecting one from several opinions as well as answering fuzzy questions not normally discussed among the traditional Arabic linguists.[1] To be honest and scientific enough, we'd like to conclude this long chapter with the following statement:

Although systematic entity descriptions have been developed, subjective factors like the Arabic linguistic background, taste, sense and feel of the developer played a major role in the quality of the resulting morphological knowledge base which is in turn reflected on the performance of the whole system.

[1] When should two similar entities be considered as one entity and when should they be assumed different?, Which entities should be considered and which ones should be ignored?, How can we be sure that we have satisfactorily collected all the members of one type of entities?, ..., etc.

Chapter 6

The Morphological Analysis Algorithm

The morphological analysis alogrithm accepts the spelling of an input word[1] W_s and tries to find all the quadruples, if any, that can represent this undiacritized word. Each of the following sections details a step of the algorithm and applies this step to the following example:

Example [6.1]

Given an input string W_s = إستميلوا ;
Get all the possible morphological analyses of this input string.

6.1 Extract All the Possible Prefixes

- Prepare an empty list L_p.
- Match the spellings, the correct one and the near misses, of all the P entities in the morphological knowledge base against W_s from the beginning and forwards.
- Put the codes of the matching prefixes in L_p.
- If L_p is empty, exit declaring the input word as an invalid Arabic word.

After applying step 1 to our example, the contents of L_p are as follows:

L_p array index	**Code**	**Prefix**
0	0	لا سابِقَ (–)
1	82	1st near-miss spelling أَلِفُ الأَمْرِ (إ)

6.2 Extract All the Possible Suffixes

- Prepare an empty list L_s.
- Match the spellings of all the S entities in the morphological knowledge base against W_s from the end and backwards.
- Put the codes of the matching suffixes in L_s.
- If L_s is empty, exit declaring the input word as an invalid Arabic word.

[1] The analysis algorithm processes the spelling part of the input string only. Shaddas and diacritics are considered in the synthesis phase presented in the next chapter.

After applying step 2 to our example, the contents of L_s are as follows:

L_s array index	Code	Suffix
0	0	لا لاحِقَ (−)
1	81	أَلِفُ الْمُثَنَّى المُضَافِ المرفوعِ (اُ)
2	251	أَلِفُ الِاثْنَيْنِ مَعَ الماضِي (اُ)
3	253	واوُ الجَمَاعةِ مَعَ الماضِي (واْ)
4	375	أَلِفُ الِاثْنَيْنِ مَعَ المُضارِعِ المنصوبِ أَو المجزومِ (اُ)
5	377	واوُ الجَمَاعةِ مَعَ المُضارِعِ المنصوبِ أَو المجزومِ (واْ)

6.3 Match the Possible Prefixes with The Possible Suffixes

A prefix IDP can be *compatible* with a suffix IDP or not according to the following binary function:

$$C(\text{prefix IDP, suffix IDP}) = \begin{cases} 1 \textit{ if the } \text{IDP's } \textit{are compatible.} \\ 0 \textit{ if the } \text{IDP's } \textit{are not compatible.} \end{cases} \quad [6.1]$$

A part of the binary table that defines the *C* function is shown in the following figure:

	...	pVORN	pQUES	pTWK	pNONE	pATF	...
⋮	...	⋮	⋮	⋮	⋮	⋮	...
sNNMN	...	1	0	1	1	1	...
sNNMH	...	1	0	1	1	1	...
sNNGZ	...	1	0	1	1	1	...
sNNGN	...	1	0	1	1	1	...
sDMRF	...	1	1	1	1	1	...
sMODR	...	1	1	1	1	1	...
sGUNM	...	1	1	1	0	1	...
sNGmd	...	1	0	0	1	1	...
sRFmd	...	1	1	1	0	1	...
sGZmd	...	1	1	1	1	1	...
⋮		⋮	⋮	⋮	⋮	⋮	

Figure[6.1]: A part of the binary table that defines the *C* function.

If any IDP in the IDP vector of a prefix *P* is not compatible with any IDP in the IDP vector of a suffix *S*, then *P* and *S* can not coexist (do not match) within the same word. Otherwise *P* and *S* match. This may be expressed by the following Boolean logic expression:

$$MatchPS = \prod_{i=1}^{M}\prod_{j=1}^{N} C(\mathrm{IDP}_p[i], \mathrm{IDP}_s[j]) \qquad [6.2]$$

Now, step 3 of the analysis algorithm is as follows:

- Prepare an empty list L_{ps}.
- Match each prefix in L_p with each suffix in L_s.
- Put the code pairs of the matching prefix-suffix pairs in L_{ps}.
- If L_{ps} is empty, exit declaring the input word as an invalid Arabic word.

After applying step 3 to our example, the contents of L_{ps} are as follows:

L_{ps} **array index**	**The matching Prefix-Suffix pairs**	**The code pairs**
0	(لا سابِقَ)(لا لاحِقَ) (–)(–)	(0,0)
1	(لا سابِقَ)(أَلِفُ الْمُثَنَّى الْمُضَافِ المرفوعِ) (–)(ا)	(0,81)
2	(لا سابِقَ)(أَلِفُ الِاثْنَيْنِ مَعَ الماضِي) (–)(ا)	(0,251)
3	(لا سابِقَ)(واوُ الجَمَاعةِ مَعَ الماضِي) (–)(وا)	(0,253)
4	(لا سابِقَ)(أَلِفُ الِاثْنَيْنِ مَعَ المُضارِعِ المنصوبِ أَو المجزومِ) (–)(ا)	(0,375)
5	(لا سابِقَ)(واوُ الجَمَاعةِ مَعَ المُضارِعِ المنصوبِ أَو المجزومِ) (–)(وا)	(0,377)
6	(أَلِفُ الأَمْرِ)(لا لاحِقَ) (ا)(–)	(82,0)
7	(أَلِفُ الأَمْرِ)(أَلِفُ الِاثْنَيْنِ مَعَ المُضارِعِ المنصوبِ أَو المجزومِ) (ا)(ا)	(82,375)
8	(أَلِفُ الأَمْرِ)(واوُ الجَمَاعةِ مَعَ المُضارِعِ المنصوبِ أَو المجزومِ) (ا)(وا)	(82,377)

6.4 Extract All the Possible Body Strings

- Prepare an empty list L_b.

For each prefix-suffix code pair $L_{ps}[i]$:

- Using the prefix's spelling string and the analytic part of the prefix's action, dissolve the prefix string away from W_s. Put the remaining strings in a temporary list $W`_p$.

For each $W`_p[j]$:

- Using the suffix's spelling string and the analytic part of the suffix's action, dissolve the suffix string away from $W`_p[j]$. Put the remaining strings in a temporary list $W`_{ps}$.

For each $W`_{ps}[k]$:

- Add $W`_{ps}[k]$ to L_b associated with the index i of the corresponding Prefix-Suffix pair in L_{ps}.

After applying step 4 to our example, the contents of L_b are as follows:

L_b array index	Body string	Index of the corresponding Prefix-Suffix pair in L_{ps}
0	إستميلوا	0
1	إستميلو	1
2	إستميلو	2
3	إستميل	3
4	إستميلو	4
5	إستميل	5
6	ستميلوا	6
7	استميلوا	6
8	ستميلو	7
9	استميلو	7
10	ستميل	8
11	استميل	8

6.5 Unify the Similar Body Strings

L_b may contain similar body strings corresponding to different elements in L_{ps}. For the sake of efficiency, each group of similar body strings is reduced to a single item b_1 contaning a single body string and a group of pointers to the corresponding elements in L_{ps}. Put the b_1 items in a list called L_{b1}.

After applying step 5 to our example, the contents of L_{b1} are as follows:

L_{b1} array index	**Body string**	**Indeces of the corresponding Prefix-Suffix pair in L_{ps}**
0	إستميلوا	0
1	إستميلو	1, 2, 4
2	إستميل	3, 5
3	ستميلوا	6
4	استميلوا	6
5	ستميلوا	7
6	استميلو	7
7	ستميل	8
8	استميل	8

6.6 Dissolve the Bodies into Roots and Forms

- Prepare an empty list L_{rf}.

For each body string in L_{b1}:

- Match all the F_{rd} entities in the morphological knowledge base with the body string.

For each match with an F_{rd} entity having code *f*:

- Extract the corresponding root.
- If the extracted root exists in the R_d entities and has a code *r*, add to L_{rf} the pair (*t* = RegularDerivative: *r*, *f*) and a pointer to the body in L_{b1}.

- Match all the F_{id} entities in the morphological knowledge base with the body string.

For each match with an F_{id} entity having code *f*:

- Get the corresponding R_d root with code *r*.
- Add to L_{rf} the pair (*t* = IrregularDerivative: *r*, *f*) and a pointer to the body in L_{b1}.

- Match all the F_f entities in the morphological knowledge base with the body string.

For each match with an F_f entity having code *f*:

- Get the corresponding R_f root with code *r*.
- Add to L_{rf} the pair (*t* = Fixed: *r*, *f*) and a pointer to the body in L_{b1}.

- Match all the F_a entities in the morphological knowledge base with the body string.

For each match with an F_a entity having code *f*:

- Get the corresponding R_a root with code *r*.
- Add to L_{rf} the pair (*t* = Arabized: *r*, *f*) and a pointer to the body in L_{b1}.

- If L_{rf} is empty, exit declaring the input word as an invalid Arabic word.

After applying step 6 to our example, the contents of L_{rf} are as follows:

L_{rf} array index	The root-form pairs *(t: r, f)*	Indeces of the corresponding bodies in L_{b1}
0	(RegularDerivative: 3806 (م و ل), 41 (x_3 y_2 x_1 تُ سْ إ))	2
1	(RegularDerivative: 3806 (م و ل), 41 (x_3 y_2 x_1 تُ سْ ا))	8
2	(RegularDerivative: 3817 (م ي ل), 42 (x_3 y_2 x_1 تُ سْ إ))	2
3	(RegularDerivative: 3817 (م ي ل), 42 (x_3 y_2 x_1 تُ سْ ا))	8
4	(RegularDerivative: 3806 (م و ل), 43 (x_3 y_2 x_1 تَ سْ إ))	2
5	(RegularDerivative: 3806 (م و ل), 43 (x_3 y_2 x_1 تَ سْ ا))	8
6	(RegularDerivative: 3817 (م ي ل), 44 (x_3 y_2 x_1 تَ سْ إ))	2
7	(RegularDerivative: 3817 (م ي ل), 44 (x_3 y_2 x_1 تَ سْ ا))	8

6.7 Match the Forms with The Corresponding Prefix-Suffix Pairs

- Prepare an empty list L_q.

For each item in L_{rf}:

- Using pointers, go to the corresponding item in L_{b1} and whence go to the corresponding prefix-suffix code pair (p, s) in L_{ps}.
- Using the relations of the form and the two IDP vectors of the prefix and the suffix, test whether the prefix and the suffix match the form. The condition of the prefix-body-suffix match or *pbsMatch* expressed verbally in section[5.4], while explaining the relations, can be expressed formally by the following Boolean logic expression:

  Defining:

$$x_1 + x_2 := x_1 \; OR \; x_2$$

$$x_1 \cdot x_2 := x_1 \; AND \; x_2$$

$$\sim x := NOT \; x$$

  where x_1 and x_2 are Boolean logic variables, and:

$$SonOf(\mathrm{IDP}_1, \mathrm{IDP}_2) := \begin{cases} TRUE \text{ or } 1 \text{ if } \mathrm{IDP}_1 \text{ is a descendant of } \mathrm{IDP}_2 \\ \text{in the prefix or suffix IDP's tree.} \\ FALSE \text{ or } 0 \text{ if } \mathrm{IDP}_1 \text{ is not a descendant of} \\ \mathrm{IDP}_2 \text{ in the prefix or suffix IDP's tree.} \end{cases}$$

  For a prefix having an IDP vector *pIDPvec* with length I_p, a suffix having an IDP vector *sIDPvec* with length I_s, and a form having R relations; let

$$pMustMatch(r, pIDPvec) = \sum_{i=1}^{I_p} SonOf(pIDPvec[i], pMust[r])$$

$$pMayMatch(r, pIDPvec) = \sim \sum_{i=1}^{I_p} \sum_{j=1}^{J_p[r]} \sim \begin{pmatrix} (SonOf(pIDPvec[i], pMayIDPvec_r[j]) + \\ SonOf(pMayIDPvec_r[j], pIDPvec[i])) \end{pmatrix}$$

$$sMustMatch(r, sIDPvec) = \sum_{i=1}^{I_s} SonOf(sIDPvec[i], sMust[r])$$

$$sMayMatch(r, sIDPvec) = \sim \sum_{i=1}^{I_s} \sum_{j=1}^{J_s[r]} \sim \begin{pmatrix} (SonOf(sIDPvec[i], sMayIDPvec_r[j]) + \\ SonOf(sMayIDPvec_r[j], sIDPvec[i])) \end{pmatrix}$$

  therefore:

$$pbsMatch = \sum_{r=1}^{R} \begin{pmatrix} pMustMatch(r, pIDPvec) \cdot pMayMatch(r, pIDPvec) \\ \cdot sMustMatch(r, sIDPvec) \cdot sMayMatch(r, sIDPvec) \end{pmatrix} \qquad [6.3]$$

- If the *pbsMatch* is TRUE, add the quadruple (t: p, r, f, s) to L_q.

- If L_q is empty, exit declaring the input word as an invalid Arabic word. Else, the contents of L_q are all the possible quadruples that can represent W_s.

After applying step 7—the last step, to our example, the contents of L_q are as follows:

L_q **array index**	**Quadruple**
0	(*t*=RegularDerivative: *P*=0 (لا سابقَ), *r*=3806 (م و ل), *f*=41 (x_3 y_2 x_1 إ س ْ ت ُ), *S*=253 (واوُ الجَمَاعةِ مَعَ الماضِي))
1	(*t*=RegularDerivative: *P*=0 (لا سابقَ), *r*=3817 (م ي ل), *f*=42 (x_3 y_2 x_1 إ س ْ ت ُ), *S*=253 (واوُ الجَمَاعةِ مَعَ الماضِي))
2	(*t*=RegularDerivative: *P*=82 (أَلِفُ الأَمْرِ), *r*=3806 (م و ل), *f*=43 (x_3 y_2 x_1 ا س ْ ت َ), *S*=377 (واوُ الجَمَاعةِ مَعَ المُضارِعِ المنصوبِ أَو المجزومِ))
3	(*t*=RegularDerivative: *P*=82 (أَلِفُ الأَمْرِ), *r*=3817 (م ي ل), *f*=44 (x_3 y_2 x_1 ا س ْ ت َ), *S*=377 (واوُ الجَمَاعةِ مَعَ المُضارِعِ المنصوبِ أَو المجزومِ))

Chapter 7

Morphological Synthesis

7.1 The Purpose of Morphological Synthesis

By the end of the previous chapter, we explained how an input word *W* is analysed to get all the quadruples, if any, that can represent only its spelling part W_s. The next phase is to resynthesize these quadruples and compare the resulting half diacritized words[1] to *W*. This process enables the morphological system to achieve the following two goals:

1. **Ambiguity reduction**: If *W* is partially diacritized[2], we reject those quadruples which represent words that do not match the partial diacaritization of *W*, hence the morphological ambiguity of *W* is reduced. Utilizing the partial diacritization, if any, of *W* for this purpose is governed by the following two rules:
   i) The diacritics, if any, of *W* are not considered if *W* contains a common near-miss spelling mistake, a shadda-state mistake or a diacritics mistake.
   ii) The shadda states, if any, of *W* are not considered if *W* contains a common near-miss spelling mistake or a shadda-state mistake.
2. **Automatic half diacritization, and correction of near-miss spelling mistakes:** When a single quadruple q_{best} is elected after the disambiguation phase[3], the input word *W* is replaced by the half-diacritized word W_{best} synthesized from q_{best}. Note also that as W_{best} is synthesized from q_{best} using the correct spelling of each entity in q_{best}, W_{best} is void of any common near-miss spelling mistakes.

7.2 The Morphological Synthesis Algorithm

1. Make an empty list L_w.

For each quadruple $L_q[i]$ in the L_q list:

2. According to formula[2.2], synthesize the diacritized word $L_w[i]$ corresponding to $L_q[i]$.
3. Compare the spelling of $L_w[i]$ with the spelling of *W*. If they do not match[4], mark $L_q[i]$ as Spell_Match and jump to step 7.
4. Compare the shadda states[1] of $L_w[i]$ with those of *W*. If they do not match, mark $L_q[i]$ as Spell_Match and jump to step 7.

---

1 Review section[2.3] for the definition of half diacritization.

2 Review section[2.3] for the definition of partial diacritization.

3 Disambiguation, which is statistical in this thesis, is the last phase of the morphological processing of an input word. The next chapter is devoted to discuss Statistical Disambiguation.

4 This mis-match occurs when *W* contains one or more common near-miss spelling mistakes.

5. Compare the diacritics[2] of $L_w[i]$ with those of W. If they do not match, mark $L_q[i]$ as Shadda_Match and jump to step 7.
6. Mark $L_q[i]$ as Diacritics_Match.

7. If one or more quadruples in L_q are marked Diacritics_Match, then:
   - Only the quadruples marked Diacritics_Match are the possible analyses of W.
   - Other quadruples are rejected from L_q.
   - Exit.
8. If one or more quadruples in L_q are marked Shadda_Match, then:
   - Only the quadruples marked Shadda_Match are the possible analyses of W.
   - Other quadruples are rejected from L_q.
   - Exit.
9. All the quadruples in L_q are possible analyses of W. Exit.

After applying the algorithm above to the L_q list obtained in section[6.7] for example[6.1], the possible analyses of W are:

i	$L_q[i]$	$L_w[i]$
0	(Regular-Derivative: 0, 3806, 41, 253)	اِسْتُمِيْلُوْا
1	(Regular-Derivative: 0, 3817, 42, 253)	اِسْتُمِيْلُوْا
2	(Regular-Derivative: 82, 3806, 43, 377)	اِسْتَمِيْلُوْا
3	(Regular-Derivative: 82, 3806, 44, 377)	اِسْتَمِيْلُوْا

Note how the common near-miss spelling mistake "إ" is corrected to "ا" in each of $L_w[i]$'s above.

[1] If the shadda state of some character in W is not supplied, it is assumed in match with the corresponding shadda state of $L_w[i]$.

[2] If the diacritic of some character in W is not supplied, it is assumed in match with the corresponding diacritic of $L_w[i]$.

Chapter 8

Statistical Disambiguation

8.1. The Disambiguation Scheme

Given an input word string *W*, we concluded the previous chapter with all its possible morphological analyses, quadruples, put in a list L_q. In this chapter, our task is to disambiguate these quadruples; i.e., choose only one; the most likely one from them. Our general scheme for disambiguation is:

- Prepare an empty scores list L_{sc}.
- Consulting some morphological statistics[1] and according to some statistical criteria, assign each quadruple $L_q[j]$ a score $L_{sc}[j]$. (Review section[4.3] for why the statistical approach is chosen.)
- Get the maximum score in the L_{sc} array;

$$S_{\max} = \max_j \left(L_{sc}[j] \right) = L_{sc}[j_{\max}]$$

  - Get

$$S_{\max}\% = \frac{S_{\max}}{\sum_j \left(L_{sc}[j] \right)}$$

- If ($S_{\max} \geq$ *Trust threshold*) AND ($S_{\max}\% \geq$ *AutoSelect threshold*), declare $L_q[j_{\max}]$ disambigeously as the best analysis of the input word. Else, ask a human operator to select the right quadruple itself.

Whereas comparing with the *Trust threshold* detects statistically feeble best quadruples, comparing with the *AutoSelect threshold* detects the statistical ambiguity among the top scoring quadruples[2].

The rest of this chapter discusses how to build the morphological statistics as well as the statistical scoring criteria of the quadruples in L_q.

---

[1] These statistics are built from a morphologically analyzed and disambiguated large-enough text corpus. As the untrained system then is still unable to disambiguate automatically, the disambiguation of the training text corpus cannot but be made manually. Having a care-fully diacritized text corpus, as was fortunately the case with this system, limits the number of possible quadruples per word hence considerably simplifies such a manual task.

[2] For a delicate text the AutoSelect threshold should be set high, say 0.99, which raises the ratio of the words that need manual checking. A less delicate text, on the other hand, may be disambiguated with a lower threshold, say 0.97, which lowers the ratio of the words that need manual checking.

8.2 The Correlation Neighbourhood

In any language (a natural language, a computer language, Mourse's signals, ..., etc.), words (the units of the language) are not occurring randomly if meaningful context is to be expressed. This non-random occurrence phenomenon is responsible of what is called the *correlation* between the neighbouring words in a sentence; i.e., the likeliness of some word to occur gets higher when some certain words occur as neighbours and gets lower when some other words occur as neighbours[1]. Example[8.1] illustrates the idea with two Arabic sentences.

Example [8.1]

Sentence 1*:* قال الله تعالى "... قرآن كريم ..."

*The occurrence of (*قال الله*) implies that (*تَعَالَى*) is the most likely solution of (*تعالى*).*

Sentence 2*:* فقلت لها تعالى إلى جواري

*The occurrence of (*فَقُلْتُ لَهَا*) implies that (*تَعَالَيْ*) is the most likely solution of (*تعالى*).*

Assume our word of interest, being disambiguated, W_0 occurs within a sentence (... W_{-2} W_{-1} W_0 W_{+1} W_{+2} ...); where W_{-i}'s are the previous neighbours and W_{+i}'s are the subsequent ones. Now, W_0 is correlated to both its previous neighbours as well as to its subsequent ones. If one is going, naturally, to scan the text in order (from the start of the text toward its end), then the disambiguated analyses Q_{+i}'s of W_{+i}'s are not available while W_0 is being examined, and we have to consider the correlation with Q_{-i}'s only. How can a possible quadruple $L_q[j]$ of W_0 be scored based on the correlation of $L_q[j]$ with Q_{-i}'s?

The well established answer used in automatic recognition systems, especially Speech Recognition, is to use the *m*-gram conditional probability for expressing correlation[2]. So the score $L_{sc}[j]$ given to a possible quadruple $L_q[j]$ of W_0 is given by the following expression:

$$L_{sc}[j] = P\left(L_q[j] \,\middle|\, Q_{-(m-1)}, Q_{-(m-2)}, ..., Q_{-2}, Q_{-1}\right) \qquad [8.1]$$

where *m* is called the width of the Correlation Neighbourhood. As the correlation of $L_q[j]$ to Q_{-i} decreases quickly with *i* increasing, *m* needs not be large. In fact, the most typical values of *m* is 2, 3 and 4.

---

[1] See Ref. 19, chapt.5, pp. 411 through pp. 435 for a discussion of the correlation of signals in general. With respect to correlation, a text stream (sequence of words) may be regarded as a digital signal.

[2] In automatic recognition systems, *m*-gram conditional probabilitiy is just an auxiliary scoring mechanism that supports the main recognition technique which may be Hidden Markov Models or Neural Nets for example.

In short hand notation, Eq.[8.1] is rewritten as:

$$L_{sc}[j] = P\left(L_q[j] \,\middle|\, Q^{-1}_{-(m-1)}\right) \qquad [8.2]$$

8.3 Two Hard Difficulties

Two important statistical features of any real text corpus are important to mention here:

1. Any finite-size text corpus, whatever large, is sparse. Sparseness means that; from all the possible *m*-gram combination of the vocabulary words, a lot, in fact most, of these combinations occur rarely or do not occur at all within the text corpus.
2. Sparseness increases as *m* gets larger.

So, if a direct, and naive, *m*-dimensional array is devoted to accomodate the occurrences of each of the possible *m*-grams in a training text corpus, the following two tough problems are encountered:

1. The needed storage for such an array is proportional to V^m; where V is the vocabulary size. If, for example, V=10000 (which is typically considered a small vocabulary size) and m=3, then the needed storage is prohibtively, and wastefully, large.
2. Due to sparseness, most of the elements of such an array, if ever implemented, will be either zeroes or unusefully very small. The minority of the elements which register considerable occurrences (neither zeroes nor very small numbers) can be regarded as reliabale estimates of the actual frequency of the corresponding *m*-grams, whereas the majority zero or very small elements can not be regarded as reliable (See ref.[21]). As computing *m*-gram probabilities directly relies on the frequency estimation, these estimates must be reliable enough.

   How can these two problems be overcome?

Amongst several good techniques designed to economically and reliably compute the *m*-gram conditional probabilities, the one employed in the IBM's speech recognizer is selected here (See ref.[21]). This technique, which will be called here *The Bayes Turing-discount Back-off technique*, works in general as follows:

1. When the occurrence of the required *m*-gram is high enough, the Bayes' rule[1] is applied directly for reliably estimating the *m*-gram conditional probability.
2. When the occurrence of the required *m*-gram is rare, the conditional probability is reliably estimated via the Turing-Good discount procedure[1].

---

[1] The Bayes' rule is the very basic law of conditional probability. It states that $P(x_2 | x_1) = P(x_1, x_2)/P(x_1)$. For *m*-grams it is written as $P(W_m | W_1^{m-1}) = P(W_1^m)/P(W_1^{m-1})$.

3. When the required *m*-gram occurs once or does not occur at all, the conditional probability is estimated recursively via the Back-off procedure using the occurrence statistics of (*m*-1)-grams.

As the Back-off procedure compensates for the unseen and seen-once *m*-grams, this technique needs only store the occurrences of those *m*-grams which occurred more than once. Hence the storage requirements are vastly reduced. Also, this technique switches (according to the frequency of the *m*-gram) among three procedures, or modes, for estimating the conditional probability as reliable as possible.

8.4 The Bayes Turing-Discount Back-off Technique

The mathematical derivation of this technique is presented in ref.[18].

The run-time phase (Estimating the *m*-gram conditional probabilities)

if $c(w_1^m) > k_1$,
Apply Bayes' rule
{

if $m>1,\ P\left(w_m \middle| w_1^{m-1}\right) = \dfrac{c(w_1^m)}{c(w_1^{m-1})}$

if $m=1,\ P\left(w_m \middle| w_1^{m-1}\right) = \dfrac{c(w_1^1)}{N}$

} Eq. Set[8.3.1]

if $\left(m=1 \text{ AND } c(w_1^m) \le k_1\right) \text{OR} \left(m>1 \text{ AND } \mathrm{k}_2 < c(w_1^m) \le k_1\right)$,
Apply Turing-Good discount
{

$$c^*(w_1^m) = \frac{\left(c(w_1^m)+1\right)\cdot n_{c(w_1^m)+1,m}}{n_{c(w_1^m),m}}$$

$$d_{c(w_1^m)} = \frac{\frac{c^*(w_1^m)}{c(w_1^m)} - \delta_m}{1-\delta_m}$$

if $m>1,\ P\left(w_m \middle| w_1^{m-1}\right) = d_{c(w_1^m)} \cdot \dfrac{c(w_1^m)}{c(w_1^{m-1})}$

if $m=1$
{

if $c(w_1^m) \ge k_2,\ P\left(w_m \middle| w_1^{m-1}\right) = d_{c(w_1^1)} \cdot \dfrac{c(w_1^1)}{N}$

if $c(w_1^1) < k_2,\ P\left(w_m \middle| w_1^{m-1}\right) = \dfrac{n_{1,1} \diagup n_{0,1}}{N}$

[1] For a thorough mathematical derivation and discussion of Turing-Good discount procedure, see ref. 21.

}
}

Eq. Set[8.3.2]

if $\left(c(w_1^m) \le k_2\right)\text{AND}\left(m>1\right)$,
Apply the back-off recursive procedure
{
if $c(w_1^{m-1}) > k_2$
{
if $m>2,\ P\left(w_m \middle| w_1^{m-1}\right) = \alpha\left(w_1^{m-1}\right)\cdot P\left(w_m \middle| w_2^{m-1}\right)$
if $m=2,\ P\left(w_m \middle| w_1^{m-1}\right) = \alpha\left(w_1^{1}\right)\cdot P\left(w_2\right)$
}
if $c(w_1^{m-1}) \le k_2$
{
if $m>2,\ P\left(w_m \middle| w_1^{m-1}\right) = P\left(w_m \middle| w_2^{m-1}\right)$
if $m=2,\ P\left(w_m \middle| w_1^{m-1}\right) = P\left(w_2\right)$
}
}

Eq. Set[8.3.3]

k_1 & k_2 are constants with the typical values k_1=5 & k_2=1.

N=Total number of the occurrences of monograms.

$n_{r,m}$=number of m-grams that occured exactly r times.

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 $1 \le m \le M$.
2. Get the counts $n_{r,m}$ for all $1 \le r \le k_1+1$ and $1 \le m \le M$.
3. Compute $\delta_m = \dfrac{(k_1+1)\cdot n_{k_1+1,m}}{n_{1,m}}$; $1 \le m \le M$.
4. Sort all the m-grams ascendingly using the *quick-sort* algorithm for later fast access using *binary search*.
5. Compute the α parameters as:

$$\alpha(w_1^{m-1}) = \frac{1 - \sum_{w_m : c(w_1^m)>0} P\left(w_m \middle| w_1^{m-1}\right)}{1 - \sum_{w_m : c(w_1^m)>0} P\left(w_m \middle| w_2^{m-1}\right)};\ m>1$$

6. Discard w_1^m and the corresponding $\alpha(w_1^{m-1})$ for which $c(w_1^m) \le k_2$.

Eq. Set[8.4]

8.5 Entity 4*m*-grams Instead of Word *m*-grams

To simplify the discussion in sections[8.2] through [8.4], we assumed the word as the basic unit of text. To consider the internal structure of the word as *W=Q=(t: p, r, f, s)*, entity 4*m*-grams are built instead of word *m*-grams and the following conditional probability is computed:

$$P\left(Q_m \middle| Q_1^{m-1}\right) = P\left((e_{4m-3}, e_{4m-2}, e_{4m-1}, e_{4m}) \middle| (e_{4m-7}, e_{4m-6}, e_{4m-5}, e_{4m-4}), \ldots, (e_1, e_2, e_3, e_4)\right)$$

$$= P\left(e_{4(m-1)+1}^{4m} \middle| e_1^{4(m-1)}\right) = \frac{P\left(e_1^{4m}\right)}{P\left(e_1^{4(m-1)}\right)} = \frac{\prod_{j=1}^{4m} P\left(e_j \middle| e_1^{j-1}\right)}{\prod_{j=1}^{4(m-1)} P\left(e_j \middle| e_1^{j-1}\right)} \qquad [8.5]$$

$$= P\left(e_{4m} \middle| e_1^{4m-1}\right) \cdot P\left(e_{4m-1} \middle| e_1^{4m-2}\right) \cdot P\left(e_{4m-2} \middle| e_1^{4m-3}\right) \cdot P\left(e_{4m-3} \middle| e_1^{4m-4}\right)$$

Note: To make each entity have unique ID, we set:

$$\textit{Any type : } p \Rightarrow e=p;\ e_{pMax}=P_{Max}$$

$$\textit{Regular\_Derivative : } r \Rightarrow e=e_{pMax}+r;\ e_{rrMax}=e_{pMax}+R_{rMax}$$

$$\textit{Regular\_Derivative : } f \Rightarrow e=e_{rrMax}+f;\ e_{frMax}=e_{rrMax}+F_{rMax}$$

$$\textit{Irregular\_Derivative : } f \Rightarrow e=e_{frMax}+f;\ e_{irMax}=e_{frMax}+F_{iMax}$$

⋮

Example[8.2] shows the statistically disambiguated morphological analyses of the word of example[6.1] at two different contexts. The highest *m*-gram statistics used for this example is bigram; i.e., *M*=2.

Example [8.2]: i- Disambiguating . . . لَقَدْ إستميلوا

j	$L_q[j]$	$L_w[j]$	$L_{sc}[j]$%
0	(Regular_Derivative: 0, 3817, 42, 253)	اِسْتُمِيْلُواْ	90.34
1	(Regular_Derivative: 82, 3817, 44, 377)	اِسْتَمِيْلُواْ	8.20
2	(Regular_Derivative: 82, 3806, 43, 377)	اِسْتَمِيْلُواْ	1.21
3	(Regular_Derivative: 0, 3806, 41, 253)	اِسْتُمِيْلُواْ	0.25

ii- Disambiguating . . . إستميلوا عقله

j	$L_q[j]$	$L_w[j]$	$L_{sc}[j]$%
0	(Regular_Derivative: 82, 3817, 44, 377)	اِسْتُمِيْلُواْ	49.67
1	(Regular_Derivative: 0, 3817, 42, 253)	اِسْتَمِيلُواْ	40.30
2	(Regular_Derivative: 82, 3806, 43, 377)	اِسْتَمِيلُواْ	8.41
3	(Regular_Derivative: 0, 3806, 41, 253)	اِسْتُمِيْلُواْ	1.62

Chapter 9

Conclusions and Suggestions for Future Work

9.1 Conclusions

As stated in chapter 4, the main three challenges before a successful computational Arabic morphological analyzer are coverage, tolerability and ambiguity. While the tolerability of an analyzer is somewhat qualitative, equations[4.2] and [4.4] define quantitatively both the approximate coverage $c^{\sim}$ and the disambiguation *d* respectively. For our analyzer Morpho3, the following table shows $c^{\sim}$ and *d* after two experiments:

Table [9.1]

Experiment	Size of the training text corpus	Coverage			Disambiguation trust threshold=99%		
		M_T	M	$c^{\sim}=M/M$	M_T	M	d=M/M
1	15,000 words	5,000	4,900	98%	5,000	3511	70.21%
2	50,000 words	10,000	9,873	98.73%	10,000	7934	79.34%

Although training Morpho3 on larger text corpus and trying it on other large text samples are necessary for a more reliable evaluation, the table above suggests that Morpho3 is promising. Based upon table[9.1], we may expect, or "hope", the following results after larger experiments:

Table [9.2]

Experiment	Size of the training text corpus	Coverage			Disambiguation trust threshold=99%		
		M_T	M	$c^{\sim}$	M_T	M	d
1	250,000 words	10,000	9,890	98.9%	10,000	8800	88%
2	500,000 words	10,000	9,905	99.05%	10,000	9430	94.3%

Besides producing Morpho3, working on the computational analysis of the Arabic morphology founded three essential principles that we think invaluable while working on natural language processing in general:

1. Modeling the linguistic phenomenon is crucial to the success of its computational analysis. The model has to own the following properties:
   - The model has to be general; i.e., it has to cover all, or the sweeping majority of, the cases of the phenomenon. Otherwise ad hoc violations of the model will be inevitable which in turn will drive the system towards uncontrollability.
   - The model has to be clear and simple enough. This allows developing concepts and mental images around the model and hence producing a reliable and efficient implementation of the system.

2. Dealing with the basic entities of the linguistic phenomenon is a primary key towards producing a clean, compact, economic, extensible, deep and almost complete solution. Dealing with the combinations of these basic entities, if ever feasible, will most probably produce a dirty, fat, limited, superficial and partial system.
3. Marrying rule-oriented methods and statistical methods is a promising approach towards solving problems in computational linguistics where neither methodology alone can produce satisfactory performance.

9.2 Suggestions for Future Work

Apart from working on other direct applications on morphological analysis than the ones mentioned in chapter 3, we foresee the future work of this thesis as follows:

1. Although our statistical disambiguation technique based on *m*-grams works alright, a more involved technique based on *maximum entropy models*[1] has the following important added feature:
   - It can utilize simple syntactic or/and semantic constraints besides the mere statistical entity correlations for disambiguating the multiple morphological analyses of a given word within some context.

   Hence, the employment of such a disambiguation technique will result in two important benefits:
   i. The disambiguation *d* of equation[4.4] will be much better.
   ii. Gradually merging more and more simple syntactic or/and semantic constraints with statistical entity correlations may at some point cover a considerable area of the huge syntactic or/and the semantic analysis problem.
2. A computational processor of Arabic morphology may be used to easily and quickly develop some useful task-oriented systems belonging to higher linguistic layers. One such system belonging to the semantic layer is proposed here. Given an Arabic word, the task of this system is to search some text for those words whose meaning is *synonymous*, o*pposite* or irrelevant to that of the given word. Inspired by ref.[2], we claim that each entity has a general lexical semantic (or a small set of possible lexical semantics[2]) and that the number of unrepeated general lexical semantics for some

[1] See "Maximum Entropy Models for Natural Language Ambiguity Resolution", By Adwait Ratnaparkhi, 1998, Ph.D. thesis in Computer and Information Science, University of Pennsylvania.

[2] Similar to ambiguous morphological entities, the ambiguous semantics may also be statistically disambiguated.

type of entities is small[1]. Hence, manually devising a "semantic similarity score" table for each type of entities is a feasible task. If some way is found to satisfactorily combine the similarity scores of the corresponding pairs of entities of two words and get the similarity score of the two whole words, semantic search can be readily implemented.

[1] For our 4500 R_{rd} entities, we can claim that there are at most 500 general lexical semantics.

References

قائمةُ المَراجِعِ العَرَبيَّةِ

(١) "دِيوانُ الأَدَبِ"، أَوَّلُ مُعْجَمٍ عَرَبِيٍّ مُرَتَّبٍ حَسْبَ الصِّيَغِ الصَّرْفِيَّةِ، أَبُو إبْراهيمَ إسْحاقَ ابْنُ إبراهيمَ الفارابِيّ، تحقيقُ د/أَحْمَد مختار عُمَر، ١٩٧٤.

(٢) "مَقاييسُ اللُّغَةِ"، أَبُو الحُسَيْنِ أحْمَدُ ابْنُ فارِسٍ ابْنُ زَكَرِيَّا، تحقيقُ أ/عَبْدالسَّلَامِ محمَّد هارون، الطَّبْعَةُ الثَّانِيَـــةُ، ١٩٦٩.

(٣) "لِسانُ العَرَبِ"، أَبُو الفَضْلِ جَمَالُ الدِّين محمَّدٌ ابْنُ منظور، دارُ فاضل، بَيْروت.

(٤) "مُعْجَمُ مَتْنِ اللُّغَةِ"، أَحْمَد رِضَا، مَكْتَبَةُ الحَيَاةِ، بيْروت، ١٩٦٠.

(٥) "مُعْجَمُ قَواعِدِ العَرَبيَّةِ العالَمِيَّةِ"، أنْطُوانُ الدَّحْداح، الطَّبْعة الأُولَى، ١٩٩٠، مَكْتَبة لُبْنان.

(٦) "القاموسُ الطَّويلُ لِلْقُرْآنِ الكَرِيمِ"، إبراهيمُ أَحْمَد عَبْد الفَتَّاح، مَجْمَعُ البُحوثِ الإسلاميَّةِ، ١٩٨٣.

(٧) "فِي النَّصِّ الأَدَبِيِّ"، دِراسةٌ أُسْلوبيَّةٌ إحصائيَّةٌ، د/سَعْد مصلوح، النَّادِي الأَدَبِيُّ الثَّقَافِيُّ جِدَّة، الطَّبْعةُ الأُولَى، ١٩٩١.

(٨) "الحُقولُ الدَّلَالِيَّةُ الصَّرْفِيَّةُ لِلأَفْعالِ العَرَبيَّةِ"، سُلَيْمان فَيَّاض، دارُ المَرِّيخِ بالرِّياضِ، ١٩٩٠.

(٩) "التَّطبيقُ الصَّرْفِيُّ"، د/عَبْدُهُ الرَّاجحيّ، دارُ المَعْرِفةِ الجامِعِيَّةِ، الإسْكَنْدريَّةُ، ١٩٩٣.

(١٠) "المُعْجَمُ الوَسيطُ"، مَجْمَعُ اللُّغَةِ العَرَبيَّةِ بالقاهِرةِ، الطَبْعةُ الثَّالِثَةُ، ١٩٨٥.

(١١) "مُعْجَمُ أَلْفاظِ القُرْآنِ الكَرِيمِ"، مَجْمَعُ اللُّغَةِ العَرَبيَّةِ بالقاهِرةِ، طَبْعةٌ مُنَقَّحةٌ، ١٩٩٠.

(١٢) ما تَمَّ مِنْ مشروعِ "المُعْجَمِ الكَبِيرِ"، مَجْمَعُ اللُّغَةِ العَرَبيَّةِ بالقاهِرةِ، حَرْفُ الأَلِفِ، الطَّبْعَةُ الأُولَى، ١٩٩٠، حَرْفُ الباءِ، الطَّبْعَةُ الأُولَى، ١٩٩٠، حَرْفا التَّاءِ والثَّاءِ، الطَّبْعَةُ الأُولَى، ١٩٩٢.

(١٣) "مُختارُ الصِّحاحِ"، مُحَمَّدُ ابْنُ أَبِي بَكْرٍ ابْنُ عَبْدِ القادِرِ الرَّازِيّ، دارُ الجِيلِ، بَيْروت، طَبْعةٌ حَديثةٌ مُنَقَّحةٌ.

(١٤) "المُغْنِي فِي تصريفِ الأَفْعالِ"، مُحمَّد عَبْدُ الخالِقِ عُضَيْمة، دارُ الحَديثِ.

(١٥) "تاجُ العَروسِ مِنْ جَواهِرِ القاموسِ"، السَّيِّد/مُحمَّدُ مُرْتَضَى الحُسَيْنِيُّ الزَّبيديُّ، طَبْعةُ وَزَارةِ الإعلامِ الكُوَيْتيَّةِ.

The list of English references

(16) Grosz, B.J., Jones, K.S., and Webber, B.L., "Readings in Natural Language Processing", Morgan Kauffman publishers, 1986.

(17) Kapur, J.N., Saxena, .H.C., "Mathematical Statistics", 7th edition, S. Chand & Co. (Pvt.) LTD, 1972.

(18) Katz, S.M., "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.

(19) Lathi, B.P., "Modern Digital and Analog Communication systems", 2nd edition, Holt, Rinehart and Winston Inc.

(20) Mazen Al-Wa'er, "Toward a Modern Theory of Basic Structures in Arabic", Ed. Arab School of Science and Technology, Damascus, Syria, 1983.

(21) Nadas, A., "On Turing's Formula for Word Probabilities", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-33 no. 6, December 1985.

(22) Nagy Fatehy Mohammad, "An Integrated Morphological and Syntactic Arabic Language Processor Based on a Novel Lexicon Search Technique", master thesis, Faculty of Engineering, Cairo University, 1995.

(23) Rich, E., Knight, K., "Artificial Intelligence" 2nd edition, McGraw-Hill, 1991.

(24) Schatz, B., Mischo, W.H., Cole, T.W, Hardin, J.B, Bishop, A.P., Chen, H., "Federating Diverse Collections of Scientific Literature", IEEE Computer Magazine, May 1996, pp. 28 to pp. 36.

(25) Winston, P.H., "Artificial Intelligence" 3rd edition, Addison Wesley, 1992.

ملخص الرسالة

يُعْتَبَرُ الْمُسْتَوَى الصَّرْفِيُّ بِمَثَابَةِ الأَسَاسِ الَّـذِي يُبْنَـى عَلَيْـهِ كُـلٌّ مِـنَ الْمُسْـتَوَيَيْنِ النَّحْـوِيِّ والـدِّلالِيِّ فِـي بِنْيَـةِ أَيَّةِ لُغَةٍ حَيَّةٍ. وَلِـذَلِكَ فَـإِنَّ مُعْظَـمَ الـنُّظُمِ الْبَرْمَجِيَّـةِ الْمُتَعَلِّقَـةِ بَالاتِّصَـالِ بَـيْنَ الإِنْسَـانِ وَالْحَاسُـوبِ عَبْـرَ الـنَّصِّ الْمَكْتُـوبِ تَحْتَـاجُ بِشَـكْلٍ مُبَاشِـرٍ أَوْ غَيْـرِ مُبَاشِـرٍ لِمُعَالِجَـاتٍ صَـرْفِيَّةٍ آلِيَّـةٍ ذَاتِ اعْتِمَادِيَّـةٍ وَكَفَاءَةٍ، وَلَعَلَّنَـا لا نُبَـالِغُ إِذَا اعْتَبَرْنَـا أَنَّ اللُّغَـةَ الْعَرَبِيَّـةَ مِـنْ بَـيْنِ جَمِيـعِ اللُّغَـاتِ الحَيَّـةِ الرَّئِيسِـيَّةِ تَتَمَتَّـعُ بِنِظَامٍ صَرْفِيٍّ هُوَ الأَكْثَرُ تَفْصِـيلاً وَلَكِنَّـهُ فِـي الْوَقْـتِ نَفْسِـهِ الأَكْثَـرُ اطِّـرَادًا، وَلِـذَلِكَ فَـإِنَّ احْتِـوَاءِهِ فِـي نُمُوذَجٍ رِيَاضِيٍّ هُوَ أَمْـرٌ مُمْكِـنٌ بَـلْ وَضَـرُورِيٌّ لأَيَّـةِ مُحَاوَلَـةٍ جَـادَّةٍ لِلْمُعَالجَـةِ الحَاسُـوبِيَّةِ لِلُّغَـةِ الْعَرَبِيَّـةِ بِكُلِّ مُسْتَوَيَاتِهَا.

وَقَـدْ تَـمَّ - بِفَضْـلِ اللهِ - إِنْجَـازُ مُعَـالِجٍ حَاسُـوبِيٍّ عَـالِيَ الأَدَاءِ لِلصَّـرْفِ الْعَرَبِـيِّ فَضْـلاً عَـنْ طَائِفَةٍ مِنَ التَّطْبِيقَاتِ الْمُعْتَمِـدَةِ عَلَيْـهِ وَالْمُكَمِّـلاتِ الْمُتَعَلِّقَـةِ بِـهِ. وَعَلَـى ذَلِـكَ فَـإِنَّ هَـذِهِ الرِّسَـالَةَ تَعْـرِضُ لِلأُسُسِ النَّظَرِيَّةِ الَّتِي بُنِـيَ عَلَيْهَـا هَـذَا الْمُعَـالِجُ الصَّـرْفِيُّ كَمَـا تَعْـرِضُ كَـذَلِكَ لِمَجَـالاتِ تَطْبِيقِـهِ. وَقَـدْ كُرِّسَتْ عِنَايَةٌ خَاصَّةٌ لِشَـرْحِ كَيْفِيَّـةِ مُعَالَجَـةِ التَّحَـدِّيَاتِ الأَسَاسِـيَّةِ الَّتِـي تُوَاجِـهُ أَيَّ نِظَـامٍ آلِـيٍّ لِلصَّـرْفِ الْعَرَبِيِّ أَلا وَهِيَ الشُّمُولُ وَالسَّمَاحِيَةِ وَالالْتِبَاسِ.

وَأَخِيرًا فَإِنَّنَا نَلْفِتُ الانْتِبَـاهَ إِلَـى أَنَّ مُعَالِجُنَـا الصَّـرْفِيُّ هَـذَا يُمْكِـنُ أَنْ يَكُـونَ مِثَـالاً عَلَـى كَيْفِيَّـةِ الْمُزَاوَجَةِ بَيْنَ إِطَارٍ مَعْرِفِيٍّ مُكَـوَّنٍ مِـنْ قَوَاعِـدَ مُحْكَمَـةِ التَّعْرِيـفِ وَبَـيْنَ إِطَـارٍ مَعْرِفِـيٍّ إِحْصَـائِيٍّ بِهَـدَفِ إِيجَادِ حُلُولٍ لِمَسَائِلِ اللُّغَوِيَّاتِ الْحَاسُوبِيَّةِ.

بِسْمِ اللهِ الرَّحْمنِ الرَّحيمِ

{الحَمْدُ لِلّهِ الَّذِي هَدَانا لِهذَا وَمَا كُنَّا لِنَهْتَدِيَ لَوْلا أَنْ هَدَانَا اللّهُ}

صَدَقَ اللهُ العَظيمُ

عَرْضٌ لِمُعالِجٍ آلِيٍّ شامِلٍ لِلصَّرْفِ العَرَبِيِّ، وتطبيقاتِهِ

إعداد

محمَّد عَطِيَّة محمَّد العَرَبيّ أحمد

رِسالةٌ مقدَّمةٌ إلَى كُلِّيَّةِ الهَنْدَسةِ، جامِعة القاهِرة
كجُزْءٍ مِنْ مُتطلَّباتِ الحُصول عَلَى دَرَجةِ الماجستير
فِي هَنْدَسةِ الحاسِباتِ

كُلِّيَّة الهَنْدَسةِ، جامِعة القاهِرة
الجِيزة، جُمهوريَّةُ مِصْرَ العَرَبيَّة
يناير ٢٠٠٠

عَرْضٌ لِمُعالِجٍ آلِيٍّ شامِلٍ لِلصَّرْفِ العَرَبِيِّ، وتطبيقاتِهِ

إعداد

محمَّد عَطِيَّة محمَّد العَرَبيّ أحمد

رِسالةٌ مقدَّمةٌ إلَى كُلِّيَّةِ الهَنْدَسةِ، جامِعة القاهِرة
كجُزْءٍ مِنْ مُتطلَّباتِ الحُصول عَلَى دَرَجةِ الماجستير
فِي هَنْدَسةِ الحاسِباتِ

تَحْتَ إشراف

أُستاذ دكتور

عَلِيّ حَسَن فَهْميّ

كُلِّيَّة الهَنْدَسةِ، جامِعة القاهِرة

أُستاذ مُساعِد، دكتور

مُحْسِن عَبْد الرَّازق رَشْوان

كُلِّيَّة الهَنْدَسةِ، جامِعة القاهِرة

كُلِّيَّة الهَنْدَسةِ، جامِعة القاهِرة
الجِيزة، جُمهوريَّةُ مِصْرَ العَرَبيَّة
يناير ٢٠٠٠

عَرْضٌ لِمُعالِجٍ آلِيٍّ شامِلٍ لِلصَّرْفِ العَرَبِيِّ، وتطبيقاتِهِ

إعداد

محمَّد عَطِيَّة محمَّد العَرَبيّ أحمد

رِسالةٌ مقدَّمةٌ إلَى كُلِّيَّةِ الهَنْدَسةِ، جامِعة القاهِرة
كجُزْءٍ مِنْ مُتطلَّباتِ الحُصول عَلَى دَرَجةِ الماجستير
فِي هَنْدَسةِ الحاسِباتِ

يعتمد من لجنة الممتحنين

أُستاذ دكتور إبراهيم فَرَج

أستاذ دكتور محمَّد جَمَال درويش

أُستاذ دكتور عَلِيّ حَسَن فَهْميّ — المُشْرِفُ الرَّئيسيّ

دكتور مُحْسِن عَبْد الرَّازق رَشْوان — المُشْرِفُ الرَّئيسيّ

كُلِّيَّة الهَنْدَسةِ، جامِعة القاهِرة
الجِيزة، جُمهوريَّةُ مِصْرَ العَرَبيَّة
ديسمبر ١٩٩٩