University of Sussex

Unsupervised Learning of Arabic Non-Concatenative Morphology

Bilal Khaliq

Submitted for the degree of Doctor of Philosophy
University of Sussex
May 2015

Declaration

I hereby declare that this thesis has not been and will not be, submitted in whole or in part to another University for the award of any other degree.

Signature:

Bilal Khaliq

Summary

Unsupervised approaches to learning the morphology of a language play an important role in computer processing of language from a practical and theoretical perspective, due their minimal reliance on manually produced linguistic resources and human annotation. Such approaches have been widely researched for the problem of concatenative affixation, but less attention has been paid to the intercalated (non-concatenative) morphology exhibited by Arabic and other Semitic languages.

The aim of this research is to learn the root and pattern morphology of Arabic, with accuracy comparable to manually built morphological analysis systems. The approach is kept free from human supervision or manual parameter settings, assuming only that roots and patterns intertwine to form a word.

Promising results were obtained by applying a technique adapted from previous work in concatenative morphology learning, which uses machine learning to determine relatedness between words. The output, with probabilistic relatedness values between words, was then used to rank all possible roots and patterns to form a lexicon. Analysis using trilateral roots resulted in correct root identification accuracy of approximately 86% for inflected words.

Although the machine learning-based approach is effective, it is conceptually complex. So an alternative, simpler and computationally efficient approach was then devised to obtain morpheme scores based on comparative counts of roots and patterns. In this approach, root and pattern scores are defined in terms of each other in a mutually recursive relationship, converging to an optimized morpheme ranking. This technique gives slightly better accuracy while being conceptually simpler and more efficient.

The approach, after further enhancements, was evaluated on a version of the Quranic Arabic Corpus, attaining a final accuracy of approximately 93%. A comparative evaluation shows this to be superior to two existing, well used manually built Arabic stemmers, thus demonstrating the practical feasibility of unsupervised learning of non-concatenative morphology.

Acknowledgements

الحمد لله
(All Praise is due to Allah)

ولا حول ولا قوة الا بالله
(And there is no power nor strength except with Allah)

The one thing that brought me to a research degree at Sussex was my supervisor, Professor John Carroll with whom I got along comfortably from the first day. He always kept a positive attitude towards my work and my circumstances. He has been understanding with respect to my personal commitments and in difficult times he has been supportive and always giving encouragement. I couldn't imagine somebody better than John as supervisor to whom I am greatly indebted.

Dr. Bill Keller, as my second supervisor, has also been immensely supportive and helpful. He always kept a keen interest in my work I always found him available to offer guidance in any aspect I needed. I truly appreciate his help and contribution in this work.

I'd like to specially thank Professor David Weir for providing useful comments and feedback in the annual review meeting which helped steer my work in the right direction. Simon, Hamish, Jeremy and others in the NLP group have been always helpful whenever I needed to turn to them for any problem. My former and current room-mates, Rob Koeling, El-Tayyab and Raphael have been great companions with whom I enjoyed working and always had interesting discussions with, often to relieve the stress from work.

I owe endless gratitude to my parents and parents-in-law who gave me the opportunity to pursue my PhD, supporting me in every way. They provided me with the maximum comfort and ease to help me focus on my work without letting me get distracted. The same for my sisters who have shown equal support. They all have been patiently persevering through the duration of my study.

The closest one to feel all the ups and downs during my study has been my family who has been key players in the accomplishment of my work. My wife has stood by my side all along. She has been there patiently bearing along with me, sharing my emotions in happy and stressful times. The children have been a pleasure to see and play with after a day's work.

Preface

Some of the research presented in this thesis has been published in peer-reviewed conference proceedings as follows.

Chapter 3

- Khaliq, B., Carroll, J. (2013). Unsupervised morphology learning using the Quranic Arabic Corpus. In *Proceedings of the Second Workshop on Arabic Corpus Linguistics (WACL'2)*, Lancaster, UK.

- Khaliq, B., Carroll, J. (2013). Unsupervised induction of Arabic root and pattern lexicons using machine learning. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, Hissar, Bulgaria. 350-356.

Chapter 4

- Khaliq, B., Carroll, J. (2013). Induction of root and pattern lexicon for unsupervised morphological analysis of Arabic. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*, Nagoya, Japan. 1012-1016.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

1.1 Introduction

1.1.1 Problem Definition

The field of natural language processing has over the passing years seen a significant growth in the level of automation in building and devising tools and resources which rely only minimally or not at all on the expertise of a linguist. Current sophisticated empirical and machine learning methods typically apply supervised learning techniques in conjunction with labelled data to make predictions about the desired task which approach the performance of linguistic experts. The larger and more accurate the annotated database is, the better the model learnt for prediction. Yet, there are certain situations for which labelled data may be absent or insufficient to produce an effective system. For such tasks a more unsupervised approach is needed which is able to find the hidden structure in the unlabelled data. One such field of research which requires such an unsupervised approach is the learning of morphology, especially for morphologically rich languages with limited linguistic resources.

The number and diversity of human languages makes it impractical to manually craft lexicons and morphological processors for more than a very small proportion of them. Further challenges are posed by the need to deal with dialects and colloquial forms of languages. This has motivated recent increased interest in approaches to morphological analysis based on unsupervised learning. Inspired by competitions such as the Morpho Challenge[1], many techniques have been proposed for unsupervised morphology learning.

---

[1] Website http://research.ics.aalto.fi/events/morphochallenge/ accessed 3rd May 2014

Although these techniques are often intended to be language independent, they are often directed to a specific group of languages. Most work has aimed at sequential separation or segmentation of morphemes concatenated together in a surface word form. This type of analysis, outputting stems and appended morphemes aims to identify some kind of border between the different morphemes. However, another type of word formation consists of the interdigitation of a root morpheme with an affix or pattern template; in this case there is no boundary between morphemes, since they are rather intertwined with each other. This type of non-concatenative morphology, which is characteristic of the Semitic group of languages, has attracted far less interest for unsupervised learning.

In this research I present an approach to learning the non-concatenative morphology of Arabic, given unannotated data as found in naturally written texts, while minimising supervision and manual setting of parameters.

1.1.2 Research Question

This research tackles the following research questions:

> *Can the non-concatenative morphology of Arabic be learnt effectively with performance reasonably close to that of linguistic resources and tools? To what extent can the devised approach be independent of manual settings and language specific parameters?*

1.1.3 Chapter Organization

In this chapter, I first give a brief description of the background and characteristics of the Semitic languages detailing the development of Arabic language and its dialect (section 1.2). Thereafter, I define unsupervised learning in general and in the context of morphology learning (section 1.3). This includes specifying the inputs to the system, various layers of details that are output, and the justification for using unsupervised methods to learn morphology. Next, I introduce briefly the morphology of Arabic to the level needed to understand the problem of morphological processing in this work (section 1.4). The section covers the special challenge and justification for learning the

rich morphology of Arabic with reference to the two types of morphologies, concatenative and non-concatenative. This is followed by a formal definition of a model for unsupervised learning of Arabic non-concatenative morphology specifying the input and outputs, along with the techniques for unsupervised morphology induction (section 1.5). Finally, section 1.6 outlines the thesis organization.

1.2 Language Preliminaries

1.2.1 Arabic and the Semitic Language Group

Arabic belongs to the Semitic group of languages which, originating in the Near East, are currently spoken in the regions of West Asia (the Arab peninsula), North Africa and parts of the African Horn, and also expatriate communities in the North American and European continents.

Arabic dominates the Semitic language family, being an official language, solely and jointly, of almost 20 countries in the region stretching from West Asia to North Africa. Out of the Semitic language group's (approximate) 500 million speakers, Arabic is spoken by nearly 300 million (Thompson & Phillips, 2013). The most prominent languages in this group are shown in Table 1.1 along with numbers of speakers.

Language	**Speakers**
Arabic	300 million
Amharic	22 million
Hebrew	7 million
Tigrinya	6.7 million
Silt'e	0.8 million
…	…

Table 1.1: Most spoken Semitic languages

1.2.2 Common Characteristics of Semitic Languages

Nearly all languages in the Semitic family share common characteristics in terms phonology, morphology and syntax which make them quite distinct from languages of other regions. They exhibit a kind of engineered structure showing remarkable organisation and arrangement with rich content expressed very concisely. Although having very different scripts, languages in the Semitic family share certain orthographic conventions. The most common is the use of optional diacritic markers to indicate short vowels and consonantal germination; the omission of these markers can lead to ambiguity in the analysis of words. The script for Maltese is the least ambiguous, with alphabetic spelling conventions resulting in a one-to-one mapping from grapheme to phoneme. Thereafter, Amharic with a syllabic writing system is arguably less ambiguous than Arabic, Hebrew and Syriac, which have the most ambiguity due to diacritic omission (Fabri *et al*, 2014)

In terms of phonology, Semitic languages are marked by a dearth of vocalic sounds, while having a rich consonantal system (Watson, 2002). There are only three basic vowels *a*, *I*, *u*, which are realized in their short and long forms. The consonant collection is rich in guttural sounds. The consonantal phonemes of the language group are categorized as voiced, voiceless, and 'emphatic', thus constituting a triad in what is a subset of the coronal set. The emphatic phonemes may be realized as pharyngealized, velarized, ejective, or plain voiced or voiceless consonants.

A core characteristic of Semitic languages is their root-and pattern morphology. The root consists of 2, 3 or sometimes 4 letter literals denoting a broad meaning or concept, onto which a template (or pattern) is applied to form a derived word. Typically gender in such languages is expressed both in nouns and verbs. Plurality is also expressed in nouns, which besides singular and plural forms have a third type, *dual*, though this is seldom used in contemporary dialects. In terms of verb aspects and tense, there are two distinct types of markings which are common to almost all Semitic languages: suffix conjugations for past tense, and prefix-suffix conjugations for non-past tense. The former marks the verb for gender, number and person, while in the case of the latter the prefix primarily indicates person, and the suffix indicates number and gender whenever

the prefix does not indicate these. A more detailed account of Arabic morphology is given in section 1.4.1.

Typical word order in Proto-Semitic languages such Arabic and Hebrew is the head-first order V(erb) S(ubject) (O)bject. This is in contrast to the distinct Ethiopian language Amharic which has the head final order S-O-V, with nominal phrases being Adjective-Noun. The emphatic V-S-O order is giving way to S-V-O in modern Semitic language usage, especially for Arabic dialects and Hebrew, under the influence of English and other European languages. In some dialects, particularly Bedouin, word order can be dependent on factors such as the main verb type (dynamic or stative), the type of text (distinct event narrative would tend to have head first clauses) and the tonal style or *stylistics* (Holes, 1995; Dahlgren, 1998).

1.2.3 Origins and Growth of the Arabic Language

Arabic has been the language of the people of the Arabian Peninsula since time immemorial. The language received significant impetus and spread with the coming of the religion of Islam. Over the 100 years after its emergence in the sixth century CE, the religion spread rapidly in the Arabian Peninsula reaching northern parts up to modern day Syria and Turkey; east into Iraq and western Iran; and west into Northern Africa. In the centuries to follow, the frontiers of the new faith reached far and wide, extending to Spain, Africa and Asian regions of India, Turkestan, China and further into Indonesia.

The Arabic language is the language of the Holy Book, the Quran. Islam brought not only religious and cultural change but also promoted the language through which believers could better comprehend the divine literature and teachings of the Prophet (ص). It became either the vernacular language of the regions to which Islam spread or was adopted alongside their native language.

1.2.4 Standard Arabic

While diversity in the Arabic language existed in pre-Islamic days, a formal standard form of the language began to emerge in the sixth century CE, before the advent of Islam. Poets started to use a Proto-Classical Arabic, taken predominantly from the Hejaz dialect and also other archaic dialects, to recite their poetry which was very

different from their own dialects (Lipinski, 1997). This Classical language was then codified by the revelation of the Quran adding richness to the grammatical forms. The richness of the grammar of Arabic was formalized by grammarians in the eighth century CE providing a standard for scholarly work and formal education and usage until today; this language is called Classical Arabic.

Modern Standard Arabic (MSA) has emerged as the norm for present-day formal usage, keeping largely the same syntax and morphology as Classical Arabic while differing considerably in lexis and stylistics. Standard Arabic is mostly taught in educational institutions and used for formal discourses. It is mostly written and seldom spoken, while the regional variety is the primary mode of oral communication. This standard has helped to unite the Arab speaking nations with a common means of communication. There is continued effort to preserve and promote the standard language keeping its link with the classical form for literary understanding of traditional resources.

1.2.5 Dialects

As the Arabic language spread to the various nations around the Arab peninsula, regional influences of other Semitic and non-Semitic languages began to influence the original classical language over the centuries. For example in North Africa, the Arabic language of the region has been influenced considerably by Berber and the French language. There many dialects and also colloquial forms of the Arabic language in use today. Some of the varieties resemble each other while others are quite different and largely incomprehensible to speakers from other regions. Although variation in a language occurs along different dimensions, some geographically defined variants are recognized as: *Hejaz* and *Najd* Arabic of the Western and Central (Saudi) Arabia, respectively; *Maghrebi* Arabic of Morocco, Algeria, Tunisia and Libya; *Egyptian* Arabic; *Levantine* Arabic of Lebanon, Syria, Jordan and Palestine; *Gulf* Arabic spoken in Kuwait, Bahrain, Qatar, the U.A.E. and Oman; etc.

Another important dimension of variation is social, according to the class hierarchy of a region. So the urban dialect of the affluent would be different from that of rural less affluent people and also that of the poor Bedouin class (Habash, 2010). Urban dialects

are more prone to evolution due to intermingling of speakers from diverse origins; on the other hand the dialect of the Bedouin is considered less prestigious, more rough, yet bearing more resemblance to the original Classical Arabic due to social isolation. These class-based variations in language are more pronounced in North Africa than they are in the Eastern region and Arab Peninsula. Speakers also have the tendency to switch dialects according the formality of the situation or when needing to communicate with people of other classes.

The diglossia of using a vernacular regional variety alongside standard Arabic for formal situations has continued to exist over the centuries to this day. People mostly learn their regional variety as their mother tongue while using MSA in formal environments (Watson, 2002). Speakers tend not to distinguish the two forms as separate languages, using them interchangeably according to situation. Each region feels their vernacular variant to be the one that mostly closely resembles the Standard/Classical form.

1.3 Unsupervised Learning

Unsupervised Learning aims to identify an underlying structure of some input data revealed by the distributional patterns of the key features in the data. Unlike in supervised learning or reinforcement learning there is no example knowledge to affirm the choice of a particular solution. It has similarities to the problem of *density estimation* in statistics, which is the most basic task of unsupervised learning but also encompasses other procedures that aim to explain and summarize important aspects of the data.

Studies show the existence of such learning taking place in the natural environment. The human mind, for example, processes information of visual images in an unsupervised manner. Clustering has been used in an unsupervised way in simulations to process the photoreceptor activities to capture the images of objects characterized by a low dimensional cluster having fewer degrees of variation (Dayan, 1999).

Unsupervised learning is widely used in scientific research where some of the common approaches include clustering, self-organizing map (SOM) from neural networks, hidden Markov models, principal component analysis etc. For example, SOM is used for certain pattern recognition tasks such as automatic target recognition, which is an element in robotic warfare (Ohno *et al*, 2013).

In natural language processing, unsupervised learning has been used in a variety of tasks such as grammar and lexicon induction, part-of-speech tagging etc. In grammar induction, for example, the underlying syntactic structure of a grammatical component is recognized for use in further NLP tasks (Klein & Manning, 2002; Clark & Lappin, 2010). One of the main reasons for the popularity of using this approach in NLP is the advantage of not requiring labelled datasets, which may be expensive to produce.

1.3.1 Unsupervised Learning of Morphology (ULM)

Unsupervised learning of morphology is a general expression referring to the problem of analysing text in the absence of annotation to reveal the required levels of description of how morphemes have been combined to form words, in a particular language. There are expressions used to refer to this problem, including (unsupervised) morphology induction, automatic word segmentation, and stemming.

The various aspects of the problem of unsupervised morphology learning are discussed below.

1.3.1.1 Input

As the aim is to process a language without making use of any linguistic aids and tools, the input is simply the written text of the language without other knowledge or cues to describe the text except the words themselves. Hence, an important consideration is the size and composition of the dataset. Thus, standard, edited text, which is less likely to contain inaccuracies, while also being rich with morphed word types, is preferable in order to produce a sound analysis. System accuracy would be dependent on the ratio of inflected word frequency to non-inflected word types such as proper nouns. As the techniques are based on statistical counts of morphemes, uninflected word types would

add to noise in the data. It may be advantageous to work on smaller rather than larger datasets, particularly if the complexity of the learning algorithm is much worse than linear in the number of input word tokens that are input to the system.

The structure of input text is another consideration for ULM systems. It is assumed that a sentence of the text data is broken down into word tokens, rather than a complete utterance as in the case of Chinese and Japanese where sentences but not word tokens are delimited. Such languages would first require segmentation of the sentence into its component words, as done by Xue (2003) and others, before input into the ULM system for morphological analysis. Word context may be of some use for the ULM problem but most systems in the literature base their processing on just the vocabulary of the dataset to produce the desired output. Hence for such systems, functioning on orthographic tokens, the input would be just a bag of words.

1.3.1.2 Output

The output of ULM varies significantly between researchers, ranging from the simplest task of affix induction to the more complex identification of paradigms for stems. Hammarström (2009) presents an ‘implication hierarchy’ to show the different types of analysis that ULM systems may output, illustrated in figure Figure 1.1.

The lower levels in the hierarchy usually imply the higher level solutions, which are trivially obtainable. For example, it is possible to easily make same-stem decisions given a segmentation of the words. But the converse is not true. A segmentation of all words is not possible if the output is simply a same-stem decision.

List of affixes	List, possibly ranked, of either prefixes, suffixes or patterns
↑	
Same-stem decisions	To decide whether two stems have the same affix
↑	
Segmentation	Segment the words into constituent morphemes
↑	
List of paradigms	Paradigm are affixes subsets that occur with certain stem types.
↑	
Lexicon+Paradigm	List of stems with links to the paradigm that each stem is associated with.

Figure 1.1: Possible levels of outputs from a ULM system

1.3.1.3 Modelling

Depending on the type of morphology to be learnt, e.g. either concatenative or non-concatenative, an appropriate model is chosen to represent the problem for learning the morphology. Usually this model, although built with a specific language in perspective, is generic enough to be applied to other languages exhibiting the same characteristics of morphology represented by the chosen model.

There are some common assumptions that guide all models designed for unsupervised morphology learning, such as, affix strings generally have higher occurrence counts than the remaining stem/root which has a relatively lower frequency of occurrence. Other assumptions might be specific to type of morphology being modelled.

1.3.1.4 Supervision

The aim is to build into the unsupervised learning technique as few language specific assumptions as possible. In order to keep the technique purely unsupervised, there should be no parameters or thresholds that require to be set by a human.

1.3.1.5 ULM Problem

The ULM problem can be visualized as illustrated in Figure 1.2.

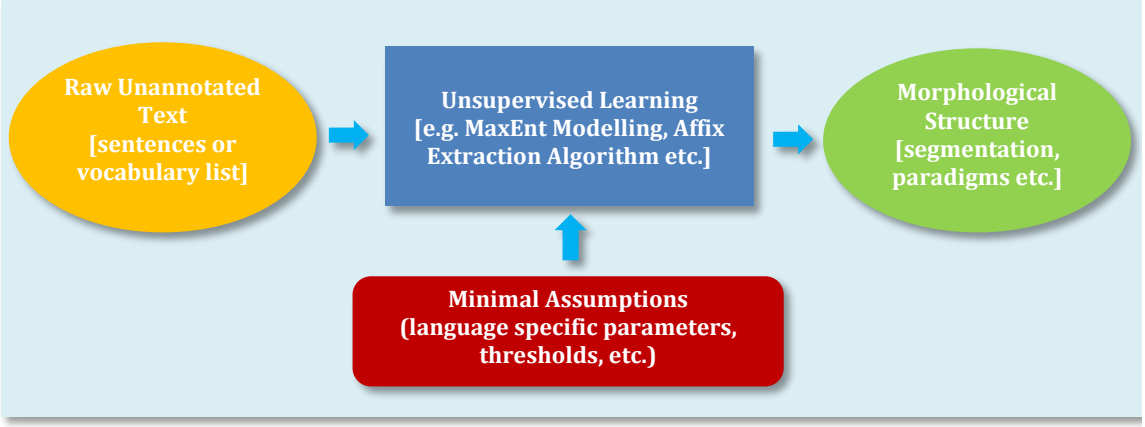


Figure 1.2: General processing steps for unsupervised learning

1.3.2 Justification for ULM

Research in ULM has a long history. Researchers have been motivated by a variety of reasons for developing morphology induction techniques. As early as the 1950s, some researchers were interested in ULM from a theoretical perspective (Bloomfield, 1963), studying the morphological structure of language based on distribution rather than semantics (Andreev, 1963, 1965). Some researchers such as Clark (2002) are interested in the modelling of the human language acquisition process which is largely unsupervised in nature. Another reason has been the difficulty of accommodating large dictionaries in the limited memory and storage systems of the past (Wothke, 1986; Klenk, 1985).

Why would researchers be interested in investigating unsupervised learning approaches to morphology when advances have made it possible to build resources such as large manually encoded dictionaries, finite-state approaches with hand written rules, and techniques for supervised learning that are known to give high performance? The core reason has been to counter the cost of manual labour required in building lexical resources for use in advanced natural language applications. For very many languages and dialects there are no existing linguistic resources. In supervised learning, the labelled data may be difficult and expensive to obtain requiring intensive manual labour and standardization. Even for resource-rich languages there is a constant flux in vocabulary with new word usages and adaptations, thus requiring constant updating which itself is an overhead. On the other hand, unsupervised morphology learning offers the possibility to acquire the morphology of a language without incurring much expense and manual labour and can be applied to a diverse set of languages.

1.4 **Morphology**

As the term implies literally, morphology (from ancient Greek, *morphe + logos*), is the study/discourse (*logos*) of changes in form (*morphe*). In the linguistic context it mostly refers to changes in the form of words of a language. In the linguistic context it mostly refers to changes in the form of words of a language. Words are the fundamental building blocks of language. The surface forms of words can vary from simple, single meaning bearing units, to complex units, the meaning of a complete sentence or proposition. Nearly all languages combine one or more grammatical units, called morphemes, to a base form in order to convey a different meaning to the base meaning. These morpheme combinations occur in a variety of manners with different levels of complexity. The study of rules for forming the words is given much emphasis by researchers as the correct unit chosen for building the syntactic or semantic structure of the system is of fundamental importance. Just as continued research in building better language processing systems in terms of speed, efficiency, cost, robustness and applicability while catering to a diverse set of languages is a requirement, a parallel effort is needed to develop dynamic ways to build suitable morpheme word bases on which the other language structures are built.

The ways in which morphemes combine can be categorized differently. One type of categorization relevant to the computational processing of morphology is referred to as concatenative as opposed to non-concatenative morphology. When surface forms are built using morphemes that append to the beginning or the end of a word, this type of process is called concatenative morphology. The appended morpheme at the beginning is called a prefix (such as *re-* in *rewrite*) and the end is called a suffix (such as *-s* in *writes*) while some languages have circumfixes consisting of a beginning and end element (e.g. in Malay a circumfix, *ke..an* added to *adil* "fair", gives *keadilan* "fairness"). In non-concatenative morphology there is a different kind of word formation which is more complex than simple end attachments. Usually an intermingling of morphemes is seen; for example, in the Philippine language Tagalog, the affix *um* "to do something" is infixed to the stem *hingi* "ask" to form *humingi* meaning "to ask for".

1.4.1 Arabic Morphology

Arabic uses both concatenative as well as non-concatenative morphological processes.

There are two types of concatenations that take place: firstly, affixation by means of prefixes or suffixes, including inflectional morphemes marking gender, plurality and/or tense. Secondly, a final layer of clitics may attach to a word, including a subset of prepositions, conjunctions, determiners and pronouns; these appear at the beginning (proclitics) or end (enclitics) of a word.

The core of the Arabic word formation process is non-concatenative, that is, it does not consist of sequential appending together of morphemes. This type of word formation is sometimes called *templatic morphology* or *root-and-pattern morphology*, where a root and a pattern template intertwine to form a base word, which can be a noun, verb or adjective, all of which are semantically related to the root.

In the case of Arabic, a further complication is that text is usually written without diacritics or short vowels which means that most of the template letters are missing in the final word, thus adding to the ambiguity of the analysis. Multiple analyses of a word are thus possible.

The following subsections describe each important aspect of Arabic morphology. In the rest of the chapter, the Arabic script will be accompanied with a Buckwalter transliteration (see Appendix A for details).

1.4.1.1 Concatenative Morphology of Arabic

Most affixes and clitics append to the beginning and end of nouns and verbs (and sometimes particles). A few of the affixes are infixes appearing in the middle of the word, which will be discussed later when discussing non-concatenative structure (section 1.3.1.2). These affixes may be pronouns, prepositions, conjunctions or case endings. Some affixes attach to any word, noun, verb or particle while others are specific to either nouns or verbs. In this section, a brief description is given of the different types of these morphemes that attach to nouns, verbs and particles.

Amongst the common affixes, و (*w*) and ف (*f*) are clitics that appear as prefixes of any word. و (*w*) is a long vowel whose tendency is to appear in many weak root words hence is often confused between an affix and a word. The letter ل (*l*) has multiple roles in different contexts. In a noun it appears as a prepositional clitic, meaning *to* or *for*. In verbs, it is used periphrastically for emphasis in first and third person imperative, such as *let/will certainly (e.g.,* لِيَذهبُوا = "Let them go" or "They will go"). With particles, the ل (*l*) appears again as prepositional clitic. The interrogative marker أ (>) is sometimes attached to any first word of an interrogative sentence. Besides these common prefixes, some pronouns are suffixed to nouns, verbs and certain particles. Except for the 2nd person and 1st person singular the rest of the pronouns are the same for nouns and verb. Table 1.1 shows some common affixes including pronouns for the 1st, 2nd and 3rd person singular, masculine, with example usages with nouns, verbs and particles.

	Some Prefixes			**Suffixed Pronouns (Masculine, Singular)**		
	و (*w*)	**ف (*f*)**	**ل (*l*)**	**1st**	**2nd**	**3rd**
Noun	وَكُتُب (wakutub) *and books*	فَكُتُب (fakutub) *so books*	لِكُتُب (likutub) *for books*	كُتُبِي (kutubiy *my books*	كُتُبُكَ (kutubiy) *your books*	كُتُبُهُ (kutubuhu) *his books*
Verb	وَيَكْتُب (wayakutub) *and he writes*	فَيَكْتُب (fayakutub) *then he writes*	لِيَكْتُب (liyakutub) *so he writes*	سَمِعَنِي (samiEaniy) *he heard me*	سَمِعَكَ (samiEaka) *he heard you*	سَمِعَهُ (samiEahu) *he heard him*
Particle	وَفِي (wafi) *and in*	فَفِي (fafi) *so in*	لَفِي (lafi) *certainly in*	لِي (liy) *for me*	لَكَ (laka) *for you*	لَهُ (lahu) *for him*

Table 1.2: Example usages of common prefixes and suffixes

In the case of nouns, the most common affix is the determiner ال (*Al)*, which appears as a prefix, corresponding to the English determiner *the*. It has other variants, such as وال (*wAl)*, meaning *and the* and لل (*ll)*, meaning *for the*, due to the preceding conjunctions, و (*w)* and ل (*l)*. In Arabic some prepositions that attach to the noun are inseparable. The five prefixed prepositions are ب (*b*) (meaning *by/with*), ك (*k*) (meaning *as*), ل (*l*) (meaning *for*), and و (*w*) (meaning (swearing) *by the*). Nouns often gets feminized by attaching the feminine marker ت (*t*) at the end of the word. The masculine and feminine sound plurals end with ون (*wn*) and ات (*At*) respectively, and the dual attachment for masculine/feminine is ان (*An*). In the case of feminized nouns, the ة (*p*) is replaced by ت (*t*) for dual when attaching the ان (*An*). In the case of plurals the ة (*p*) is dropped, and ات (*At*) added. A list of selected prefixes and suffixes specific to nouns is shown with examples in Table 1.2.

	Noun Prefixing		Number Marking		
	ال (*Al*)	**ب (*b*)**	**Singular**	**Dual**	**Plural**
Masculine	الْمُعَلِّم (AlmuEal~im) the teacher	بِمُعَلِّم (bimuEal~im) by a teacher	مُعَلِّم (muEal~im) a teacher	مُعَلِّمَان (muEal~imAn) two teachers	مُعَلِّمُون (muEal~imwn) teachers
Feminine	الْمُعَلِّمَة (AlmuEal~imap) the teacher (*f*)	بِمُعَلِّمَة (bimuEal~imap) by a teacher (*f*)	مُعَلِّمَة (muEal~imap) a teacher (*f*)	مُعَلِّمَتان (muEal~imatAn) two teachers (*f*)	مُعَلِّمَات (muEal~imAt) teachers (*f*)

Table 1.3: Example usages of noun prefixes and suffixes

Besides the common affixes for nouns and verbs discussed above there is a class of prefixes and suffixes that is specific to verbs. Prefixes are added to represent the present-tense verb with different realizations for 1st, 2nd and 3rd person masculine and feminine. In the case of 2nd person feminine a suffix ي (*y*) is also added besides the prefix ت (t). The first three columns of Table 1.3 exemplify these different person forms for the singular. In order to put the same words into the future tense, the same prefixes are used for each respective form with the addition of the letter س (*s*) which is prefixed as a second layer on the present-tense prefix layer. An example of this is seen in column four of Table 1.3. Certain pronouns are excluded from the common pronoun suffixes. These pronouns are specific to verbs are listed in the table along with gender and person. Besides these prefixes, a suffix that may occur with 1st, 2nd and 3rd person singular feminine past-tense verbs is the attached 3rd person pronoun; for example the masculine ه (*h*) or feminine ها (*hA*) is attached as an object, making the word a sentence. This is shown in the last column of Table 1.3. Finally, the suffix, وا (*wA*) is added to the imperfect, present and past tense referring to either second person or third person.

	Present Tense Prefixes (Masculine, Singular)			**Future tense marker**	**3rd person pronoun suffix**
	1st	**2nd**	**3rd**	**س (*s*)**	**ه (*h*) /ها (*hA*)**
Masculine	أَعْلَم (>aElam) *I know*	تَعْلَم (taElam) *you know*	يَعْلَم (yaElam) *he knows*	سَيَعْلَم (sayaElam) *he will know*	سَيَعْلَمُهُ (sayaElamuhu) *he will know him*
Feminine	أَعْلَم (>aElam) I (*f*) know	تَعْلَمِي (taElamiy) *you (f) know*	تَعْلَم (taElam) *she knows*	سَتَعْلَم (sataElam) *she will know*	سَيَعْلَمُهَا (sayaElamuhaA) *he will know her*

Table 1.4: Example usage of mostly verb prefixes and suffixes

Other than these types of concatenative attachments there are some clitics such as ما (*mA*), لا (*lA*), and يا (*yA*) which may appear as proclitics of some words.

1.4.1.2 Non-Concatenative or Templatic Morphology of Arabic

As stated earlier templatic morphology is the process of word formation in which the base root letters, having a semantic meaning, intertwine with the pattern templates encoding syntactic information to obtain the derived stem word or lemma. Most Arabic roots are triliteral (3-letter), while some are quadraliteral (4-letter) and there are a few 5-letter roots. There are approximately 9000 roots listed in the famous Arabic Dictionary, *Lisan ul Araby* (Moukdad, 2006) of which 5000 roots are in usage in Modern Standard Arabic (MSA) (Beesley, 1996). Attia *et al* (2011) have compiled up to 549 patterns in Arabic, of which most patterns are rarely used. The different templatic formations can be categorized into three types: verb patterns, derivational patterns and nominal broken plural patterns.

Verbs have a several patterns of which 12 basic patterns are the most important. A few additional patterns are not used frequently. Ten of the twelve patterns occur with 3-letter

roots and the remaining two occur with 4-letter roots. Verb patterns are usually represented using three abstract letters ف(*f*), ع(*E*), and ل(*l*). The most basic pattern, which just used to represent the three and four letter root, is ف - ع - ل (*f-E-l)* and ف - ع - ل - ل (*f-E-l-l*) . This pattern representation is known as the *scale* or *form*. Each individual pattern on a particular scale has its own meaning but some of these scales are semantically related, one having been derived from the other; for example, the form II is the causative of form I. Some of the verb scales or forms are standardly denoted by Roman numerals in MSA; Table 1.4 shows a sample, with their respective meanings.

Form		**Transliteration**	**Meaning**	**Example**
I	فَعَلَ	f-a-E-a-l-a	The simplest, basic form of 3-letter root in past tense verb	كَتَبَ (kataba) he wrote
II	فَعَّلَ	f-a-EE-a-l-a	Causative: to make someone do an action	عَلَّمَ (Eallama) *he taught*
III	تَفَعَّلَ	t-a-f-a-EE-a-l-a	Reflexive of form I-II: this form acts as the object receiving the action of Form I-II	تَذَكَّرَ (ta*akkara) *he received the reminder*
…	…	…	…	…
QI	فَعْلَلَ	f-a-E-l-a-l-a	Basic form of 4-letter root	وَسْوَسَ (waswasa) *he whispered*
QII	تَفَعْلَلَ	t-a-f-a-E-l-a-l-a	Reflexive or reflexive causative of II-I, like form I-III	تَوَسْوَسَ (tawaswasa) he was whispered to
…	…	…	…	…

Table 1.5: Some example patterns for 3 and 4 letter rooted verbs along with their meanings and examples

Some words are derived from other words; the most common occurrence of derivation occurs where a noun derives from a verb form. Most of the derivational changes involve a change in pattern while sometimes affixes are appended. Sometimes there is a particular pattern that is applied to a particular verb form while elsewhere there is

considerable variety in the types of patterns that may be applied. For instance, a deverbal noun is obtained from a form I verb using a variety of patterns while the derivation from all other forms is obtained using a single pattern. The active participle and passive particle derive from verbs of different forms. Likewise, nouns of place, time and denoting instruments are also derivable from verbs. Some example derivations are shown in Table 1.5.

Form	**Transliter-ation**	**Meaning**	**Example**
فاَعِل	f-a-A-E-i-l	Active participle of form I-I	كاَتِبَ (kaAtib) *writer*
مَفْعُول	m-a-f-E-u-w-l	Passive participle of form I-I	مَكْتُوب (maktuwb) *written*
مَفْعَل	m-a-f-E-a-l	This form is used to indicate noun of place and time	مَكْتَب (maktab) *office*
مِفْعَال	m-i-f-E-a-A-l	A nominal pattern to denote instrument	مِفْتَاح (miktaAb) *key (one that opens)*
…	…	…	…

Table 1.6: Example patterns for derivational morphology

Another place where intercalated morphology is apparent is in the case of the Arabic broken (irregular) plural, where a singular word undergoes pattern changes, instead of the regular appending of a plural marker as seen earlier. Irregular plurals occur just as frequently as regular plurals. Patterns for the broken plural are sometimes the same as patterns for other derived words. For instance, the singular word كِتَاب (*kitAb*) and the plural word رِجَال (*rijAl*), share the same common pattern فِعَال (*fiEAl*). There is the possibility of multiple plurals for a word, which may be all broken or some broken

while others being regular plurals. Table 1.6 shows a few example patterns that are used to pluralize words.

Pattern		Example	
Arabic	**Transliteration**	**Singular**	**Plural**
أفْعَال	>-a-f-E-a-A-l	خَبَر (xabar) *news*	أخْبَار (>xbaAr) *news*
فُعُول	f-u-E-u-w-l	بُرج (burj) *tower*	بُرُوج (buruwj) *towers*
فِعَال	f-i-E-a-A-l	رَجُل (rajul) *man*	رِجَال (rijaAl) *men*
فُعُل	f-u-E-u-l	كِتَاب (kitaAb) *book*	كُتُب (kutub) *books*

Table 1.7: Example patterns for the broken plural

1.4.2 Special Issues

Below are discussed a few special issues that are of particular importance to the problem of Arabic morphology learning in this research.

1.4.2.1 Missing Diacritics

Diacritics (sometimes referred to as short vowels) in Arabic are symbols used to indicate vowels, definiteness, consonant doubling etc. These symbols, as opposed to letters, are considered optional and are omitted in most kinds of writing. The few places where text may be diacritized include religious text, especially the Quran, children's literature, and poetry (Dukes & Habash, 2010). Text is typically written without diacritics except in some places they may be placed by the author in order to disambiguate a certain meaning of a word. In the absence of diacritics, the same orthographic word form may indicate a variety of meanings. For example, the word كتب

(*ktb*), without the short vowels, could be interpreted as كَتَبَ (*kataba*) *he wrote* (3rd person past tense), كُتُب (*kutub*) *books*, and كُتِبَ (*kutiba*) *it was written* (past passive verb), amongst other meanings.

In the absence of short vowels, the number of distinct patterns also reduces resulting in fewer word formations. Of the 590 patterns identified by Attia *et al* (2011) some of these patterns are orthographically overlapping, having the same form but with distinct idiosyncratic meanings. Of these patterns, 306 patterns are orthographically non-overlapping types. But in the absence of short vowels these patterns further conflate to 180 types (see Appendix B) with a significant degree of overlap: an average of 3.2 grammatical patterns are represented by a single undiacritized template.

1.4.2.2 Morphophonemic Adjustments

The process of intercalation and concatenation of a root morpheme with templates and affixations may not be a straightforward agglutination of morphemes; sometimes the resulting word form undergoes changes, which make it quite different from its constituents (Holes, 2004). These changes follow certain morphophonemic rules applied to the components in the interdigitation and concatenation process. One particular rule which is especially relevant to the morphology learning problem is known as the weak root radical rule.

Weak roots are roots that contain one of the three long vowels *w* (*wāw*), *y* (*yā*) or *A (hamzah).* Such types of root undergo changes to the weak radical containing the long vowel, to adapt to vocalic harmony, sometimes with the vowel being switched or being completely dropped from the final word. There are rare occasions where a root may contain a weak radical which behaves like a regular consonant, and thus does not undergo any morphophonemic changes. An example of these changes is in the case of the root letters ق-و-ل (*q-w-l*) from the various derivational forms: قال (*qAl*), *he said*; يقول (*yqwl*), *he says*; قول (*qwl), a saying*; قيل (*qyl*), *it is said*; قل (*ql*), *say* (imperative), etc. In such cases it is hard to analyse the word back to its root.

1.4.2.3 Normalization

A natural language processing system may apply orthographic normalization to reduce noise and sparsity in the data. Generic tasks such as punctuation separation and encoding clean-up are universal for all types of language scripts. There are certain tasks that are specific to Arabic language processing, of which diacritic removal is particularly important. Infrequent and irregular occurrences of diacritics are considered noise and are therefore removed. Another important aspect which brings inconsistencies is that letter marking on certain types of letters is optional. Thus all letters in a particular class type are conflated to one class; for example, humzated forms of Alif (ا), أ (<), إ (>), آ (|), are replaced with bare Alif, ا (*A*). Similarly, non-Alif forms of Humza, ؤ (&) and ئ (}) are conflated to the bare humza letter ء (`). One issue with applying unsupervised learning to normalized text is that known characteristics about the language are used to manually regularize the dataset.

1.4.3 Motivation for Morphological Analysis

As seen earlier, about 5000 roots can possibly combine with approximately 500 patterns to form base words which may be further appended with multiple layers of affixes and clitics. The proliferation of word types found in a dataset due to multi-layer fusion of morphemes is quite pronounced.

With such a morphologically rich language, it becomes difficult to adequately capture word level dependencies. Due to the different patterns along with concatenation of morphemes, especially clitics, the number of alternative formations of words increases considerably. For instance, in the Quranic Arabic Corpus[2] (comprising around 80,000 word tokens) the root ب-ت-ك has 43 realizations due to different pattern and concatenative affixes and clitics. These are shown in Table 1.7.

From a machine learning perspective, derived and inflected forms reduce the number of instances of many words. This may give rise to data sparsity problems, which in turn

[2] http://corpus.quran.com/download/default.jsp

may necessitate increasing the number of parameters to obtain feasible models. Also, unique word growth is seen to occur at an exponential rate with the growing corpus size making it difficult to apply to unseen data. In Figure 1.3, the vocabulary growth rate for Arabic is shown in contrast to English. The number of unique words in an Arabic corpus increases steadily as the size of the corpus increases. In contrast, in English the growth rate tends to flatten, meaning that relatively fewer new words are seen in a corpus as the size of the corpus increases.

Word	**Translit-eration**	**Gloss**
بكتاب	bktAb	By a book
كتابكم	bktAbkm	By your book
بكتابى	bktAbY	By my book
بالكتاب	b'lktAb	By the book
كاتبوهم	fkAtbwhm	His book
فاكتبنا	f'ktbnA	Our book
فاكتبوه	f'ktbwh	so write it
فليكتب	flyktb	so he writes
كتبهافسا	fs'ktbhA	So I will write it
كاتب	kAtb	Writer
كاتبا	kAtbA	Writer
كاتبون	kAtbwn	Writers
كاتبين	kAtbyn	Writers
كتاب	ktAb	Book
كتابا	ktAbA	Book

Word	**Translit-eration**	**Gloss**
كتابه	ktAbh	His book
كتابها	ktAbhA	Her book
كتابهم	ktAbhm	Their book
كتابك	ktAbk	Your book
كتابنا	ktAbnA	Our book
كتابيه	ktAbyh	Booklet
كتب	Ktb	Books
كتبنا	ktbnA	Our books
كتبناها	ktbnAhA	We write it
كتبت	Ktbt	You write
اكتتبها	'kttbhA	I make him write it
لكتاب	lktAb	For a book
الكتاب	'lktAb	The book
للكتب	Llktb	For the books
مكتوبا	mktwbA	Written

Word	Translit-eration	Gloss
سنكتب	Snktb	We will write
ستكتب	Stktb	Will be written
تكتبوه	Tktbwh	You write it
تكتبوها	tktbwhA	You write it
بالكتاب	wb'lktAb	By the book
وكتاب	wktAb	And Book
واكتب	w'ktb	And I write
وكتبه	Wktbh	And his

Word	Translit-eration	Gloss
		books
وكتبنا	wktbnA	And we wrote
والكتاب	w'lktAb	And the book
وليكتب	Wlyktb	And he writes
ونكتب	Wnktb	And we write
يكتب	Yktb	He writes
يكتبون	Yktbwn	They write

Table 1.8: 43 realizations of the root ب-ت-ك in the QAC

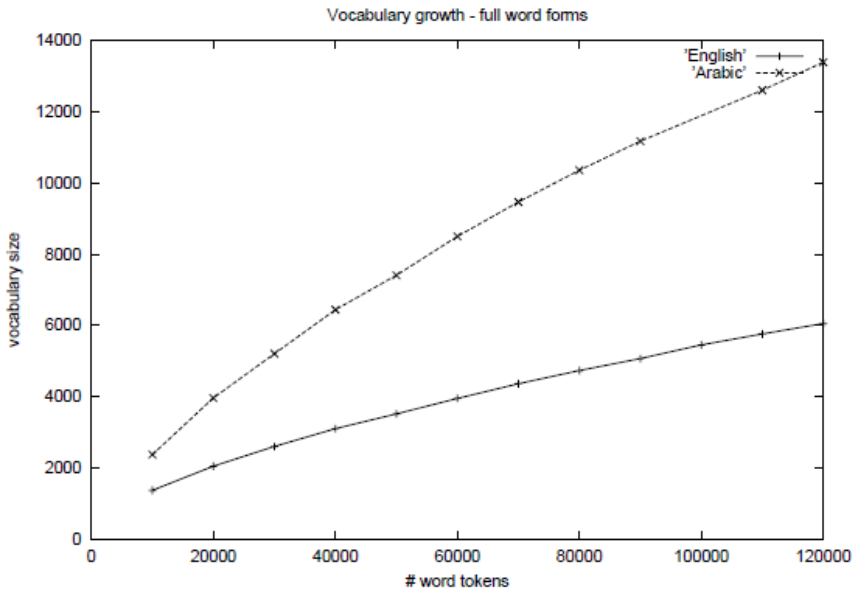


Figure 1.3: Vocabulary growth contrasted for English and Arabic (Kirchhoff *et al*, 2006)

1.5 Unsupervised Learning for Arabic Morphology

Having looked at the definition of unsupervised learning along with its related issues, and at the complexities of Arabic morphology, a model for learning the morphology of Arabic in an unsupervised manner is now presented. This section looks at: the input dataset and the characteristics of the language in it; the output, level of analysis and evaluation criteria; and, the techniques for unsupervised morphology learning using minimal supervision and language specific characteristics assumed.

1.5.1 Input Data

Unlike in supervised learning, where the training set is labelled and a separate unlabelled set is used for testing, in most work on unsupervised learning, the system learns from the unlabelled data and applies it back to the same.

I have chosen the Quranic Arabic Corpus (QAC)[3] as a test-bed for investigating unsupervised learning techniques for non-concatenative morphology. Attributes of the QAC along with relevant pre-processing for input to an unsupervised learning system are discussed below.

1.5.1.1 Undiacritized Text and Normalization

Since most Arabic text is written without vowels, a realistic setting of unvowelled text is adopted for the dataset. Using undiacritized text can be an advantage or a disadvantage depending on the type of analysis which is being attempted. Since the scope of unsupervised morphological processing is limited in terms of analysis to either stem or root, working without diacritics is an advantage which decreases the diversity of forms to be learnt. The orthographic normalization process involves, besides removal of diacritics, the normalization of Alif (ا), Humza (ء) and Ya (ى) as stated in section 1.3.2.3.

[3] See Appendix C for details of the Quranic Arabic Corpus (QAC)

1.5.1.2 Stemmed vs. Unstemmed Data

Intuitively, it would be appropriate to stem off the sequentially appended morphemes before examining non-concatenative morphology. Unstemmed words would greatly increase data sparsity, making root identification extremely challenging. Longer words also imply an exponential increase in the search space of possible solutions making the algorithm computationally more expensive. Therefore, for learning non-concatenative morphology, I use stemmed data, in order to gauge the true performance of the templatic morphological learner. This means that all inflection prefixes, suffixes and clitics are removed. Since techniques for concatenative unsupervised morphology learning are fairly advanced, stemmed words are computable through such approaches. For this research however, stemmed words in the dataset were available through a manually created resource.

1.5.1.3 Size and Composition

Unsupervised, non-concatenative morphology of Arabic is learnt using the undiacritized, stemmed vocabulary of the Quranic Arabic Corpus (QAC). The size of the vocabulary is 7369 words, of which approximately 88% of the words are derived words, composed of a pattern and a root. The high proportion of derivational forms makes it suitable for unsupervised learning. Also, the relatively small dataset size simulates the scenario for most of the world's languages of scarcity of linguistic resources and data.

1.5.2 Analysis Output

The non-concatenative morphology learning algorithm has three outputs: a scored pattern lexicon, a scored root lexicon and a procedure for morphological analysis of a word into a root and pattern. The score of an entry in the lexicons indicates the confidence the learning algorithm assigns to each morpheme in terms of its soundness.

The morphological analysis is the chosen root and pattern morpheme of a word which gives the highest combined score. In this work, the analysis is restricted to triliteral root morphemes as these account for most of the vocabulary of the language. In order to evaluate the accuracy of analyses, the percentage of correctly analysed roots is reported

as the performance measure for the algorithm. I consider only sound rooted words for evaluation. Weak rooted words would be out of the scope of the system to learn completely. (However, partial evaluation would be possible with correct identification of either one or two root radicals.)

Patterns are henceforth represented using the '-' marker to indicate the abstract letters *f, E* or *l*, intertwined with pattern affix letters. Roots are represented as triliteral strings. Due to the absence of short vowels, words are expected to contain single letter infixes. Hence at some points, the learning procedure is restricted to allow only single character occurrences between root radical place-holders in pattern templates. Two example analyses are shown in Table 1.8.

Word	**Root**	**Pattern**
ktAby	Ktb	--A-y
tEArf	Erf	t-A--

Table 1.9: Example analyses of two words

1.5.3 Model

Model formulation for morphology is different for concatenative and non-concatenative morphological structure. For concatenative morphology learning, the search space of possible morphemes (i.e. a root and affixes) is all non-interleaved substrings of a word. For an *n* character word there are 2^{n-1} possibilities. In contrast, for non-concatenative morphology, the possibilities for the root of a word are all contiguous and non-contiguous sequences of characters of length 1 and above. This corresponds to the powerset of the characters in the word minus the empty set; there are $2^n - 1$ such possibilities. Hence, the search space for both kinds of morphology is exponential, but for any value of $n>1$, there are almost twice as many possibilities for non-concatenative morphology. Table 1.9 shows the possible analyses of a four-character word, showing the possible outcomes of concatenative and non-concatenative analyses.

No.	Segmentations	Root, Pattern Combinations
1	a b c d	a,-bcd
2	a b cd	b, a-cd
3	a bcd	c, ab-d
4	ab cd	d, abc-
5	ab c d	ab, --cd
6	abc d	ac, -b-d
7	a bc d	ad, -bc-
8	abcd	bc, a--d
9		bd, a-c-
10		cd, ab--
11		abc, ---d
12		abd, --c-
13		acd, -b--
14		bcd, a---
15		abcd, ----

Table 1.10: Comparing the number of possible analyses of a hypothetical word *abcd* for concatenative and non-concatenative morphology

1.5.4 Unsupervised Learning Techniques

This thesis describes two techniques to analyse the non-concatenative morphology of Arabic to obtain the analyses of words as described above. The first technique (described in Chapter 3) uses a machine learning technique, Maximum Entropy

modelling, adapted for unsupervised learning, inputting powerset like morpheme combinations as features to train a model to cluster words based on either root or pattern similarity. The lexicons are then derived in a subsequent stage. The second technique (Chapter 4 and Chapter 5) builds a graph of all possible connections between patterns and roots, then analyses the links to reveal the potential root and pattern lexicons. This technique falls in the domain of what are known as Link Analysis Ranking algorithms which have been applied to Internet webpage ranking (Borodin *et al*, 2005).

Certain language specific characteristics are assumed in order to make the learning task feasible. As mentioned earlier, only 3-letter roots and corresponding 3-placeholder patterns are permitted. This is a supervised parameter which is basic to the learning of the morphology. Another such language specific property, in the case of undiacritized text, is to disallow root and pattern analyses where more than one consecutive infix letter is present in the pattern template. Arguably, for truly unsupervised learning there should be no such limitations; however these particular ones are minimal.

The process for unsupervised learning of Arabic morphology is illustrated in Figure 1.4.

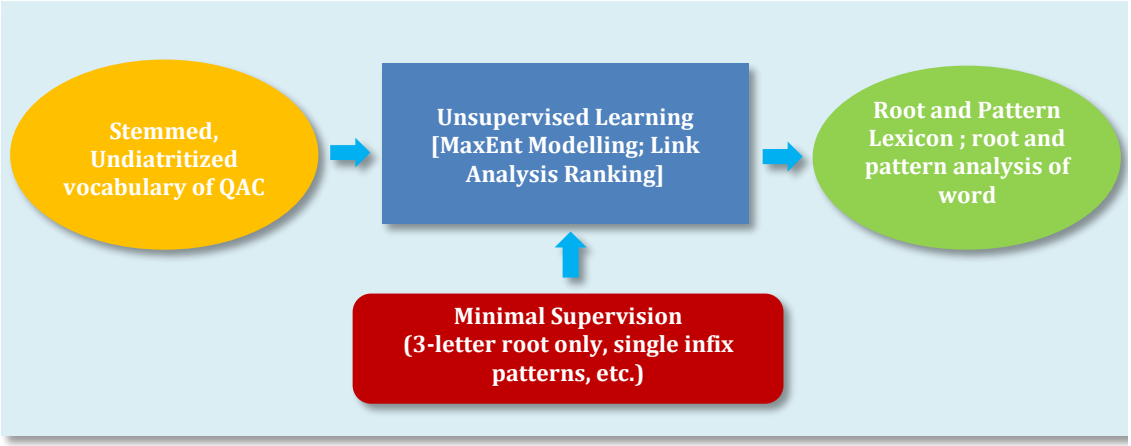


Figure 1.4: Sketch of the unsupervised learning procedure

1.6 Thesis Organization

Chapter 2 presents a survey of previous work applying empirical methods to the problem of morphology learning. Then, the first of the two methods for unsupervised learning, based on Maximum Entropy modelling, is introduced in Chapter 3. The second methodology for morphology induction, contrastive learning, based on comparative counts of roots and pattern is described in Chapter 4. This second method is then extended in Chapter 5 along with a comparison with existing manually built stemming tools. Finally, Chapter 6 summarizes the outcomes of this research and proposes areas for future work.

Chapter 2

Literature Survey

2.1 Introduction

An active area of research in natural language processing is to have computer systems capable of processing human text with little explicit knowledge about a language; this has motivated the study of unsupervised or minimally supervised learning of the underlying structure of the language purely from naturally occurring text. In the last two decades, many techniques to process and learn morphology automatically have been applied. This is partly due to advances in machine learning techniques and their successful application to such tasks, and also the increased availability of vast amounts of electronic text.

Chapter 1 introduced the task of unsupervised or minimally supervised learning of morphology, which takes a large text corpus and outputs the analysis of each word occurring in the corpus with little or no explicit knowledge of the nature of the language under consideration. Section 1.2.1 presented an 'implication hierarchy' showing different levels of analysis chosen by researchers ranging from simple same-stem identification to more complex complete word-form analysis or morphological analysis. For concatenative languages commonly the task of unsupervised learning is the automatic segmentation of word forms into morphemes. For more complex intercalated, non-concatenative languages, additional analysis could include identification of root or base form.

2.1.1 Chapter Organisation

The literature review is divided into two main parts: firstly, the area of unsupervised learning of morphology is covered generally without focusing on particular morphology

type, introducing the main techniques that have been applied to numerous tasks in this area (section 2.2). This is followed by the second part which looks at general empirical methods and some unsupervised learning methods that have been developed for Arabic and other Semitic languages (section 2.3). The review concludes by surveying areas that have received little attention to date, and are the subject of the work in this thesis (section 2.4).

2.2 Unsupervised Approaches to Morphology Learning

After introducing early approaches, the review goes on to cover work that has been influential in the past decade or so. Almost all approaches have two main steps to reach the final solution. The first step is a rough initial estimate of the solution arrived using methods involving frequency statistics of *n*-character grams taking their inspiration from the work of Harris (1955). Then follows an optimization step using various methods ranging from semantic clues to information theoretic approaches, to refine the initial solution. For non-concatenative morphology nearly all approaches involve an alignment step between inflected forms and root forms.

2.2.1 Early Work

The work of Harris (1955) is one of the first convincing attempts at unsupervised analysis of words based on the distribution of sub-strings. Harris applied his approach to English; the approach has been developed further by many researchers. The process considers distributional properties of phonemic representations of a large set of utterances, in order to identify morpheme boundaries, outputting segmented words. The technique is based on the concept of *letter successor variety*, where the frequency of the $n + 1^{st}$ letter, given the first *n* letters, is measured and a potential morpheme boundary is hypothesized at positions where sudden frequency peaks occur.

This work was given a more formal perspective by Hafer & Weiss (1974) in terms of probabilistic notions, with the inclusion of entropy into the formulation. They elucidate and make improvements on the heuristics proposed by Harris proposing many different

measures for identifying potential morpheme boundaries. Sometimes the best results were obtained using the local maximum of prefix conditional entropy, while in other cases obtaining a value above a certain threshold was used as the measure; and sometimes two measures were combined, one from the beginning to end and the other, from end to beginning of a word, using a predetermined threshold to yield the best results. While no one single measure gave the best overall performance, the best obtained result gave a precision score of 0.91 with recall of 0.61 on a corpus of 6200 words.

2.2.2 Information Theoretic Approaches

2.2.2.1 Minimum Description Length (MDL) Approaches

Goldsmith (2000, 2001, 2006) developed an unsupervised morphology induction system called *Linguistica,* which uses the *Minimum Description Length* (MDL) framework. Being publicly available, it has been used extensively as a standard for comparison by other researchers. Goldsmith focuses his attention only on suffixation (though the system is extendable to cover other affixations) applying his work to five languages, English, French, Spanish, Italian and Latin. Goldsmith's application of MDL to the problem of unsupervised morphology seeks to globally optimize the analysis of the words in the corpus. It is based on the insight that the number of letters in a set of words (in written text) is greater than the number of letters if the same words were broken down into sets of stems and morphemes. Thus, the more accurately we are able to identify the correct morphemes, the smaller (more compressed) would be the length of the decomposed data.

The input to the system is a large unannotated corpus and the output is a list of signatures and associated stems. *Signatures* are simply groups of suffixes that have been identified to be affixed by a certain group of stems e.g. *NULL.er.ing.s*. Signatures are different from paradigms as each signature may contain both derivational (-*er*) and inflectional affixes (-*ing* and -*s*) as in the above example. Also, each signature may not contain the complete set of affixes, as in paradigms such as those missing the past tense suffix, -*ed*, as in certain groups of stems having an irregular past tense e.g. *blow, drink,*

feel. Signatures are therefore only derived based on corpus statistics, and obtaining paradigmatic groupings from the signatures is not addressed in this work by Goldsmith. Obtaining signatures in *Linguistica* is a two-step process: firstly, *candidate generation* yields potential signatures and associated affixes as a starting point; secondly, *candidate evaluation* refines the initial set of candidates based on the MDL framework which is the main focus and novelty of the research.

For *candidate generation*, Goldsmith considers two heuristic methods for obtaining the initial candidates, one more rigorous, considering every possible word split and the other more intelligent and computationally concise. In the latter, the author collects all statistics of word-endings of each word up to 5 characters (6-grams including an end word marker) since he is restricting himself to languages which can have suffixes of size at most 5 characters long. Using a metric he ranks the 100 most frequently occurring suffixes. The words in the corpus are parsed using these suffixes with possible multiple parses per word. The best parse for each word is then found using another metric which assigns a probability to each parse, preferring longer suffixes over shorter ones. The results of this process are groups of stems and their associated signature. Some further heuristic processing is applied to remove signatures with one stem and stems with one suffix in the signature, resulting in what he refers to as *regular signatures*.

Once the initial set of signatures (and corresponding stems) have been obtained, in the *candidate evaluation* step these are further evaluated and refined. Based on the principle of MDL the best set will be the one that (on morphological decomposition) gives the most compact description of the corpus and of the morphology. Using different heuristics, many of the erroneous signatures are removed or modified. Each time the morphology is adjusted using a particular heuristic, MDL analysis is applied to verify any improvement in the adjustment. A lower description length indicates that the altered signature is more appropriate to keep; otherwise it is discarded. The author evaluated the system, *Linguistica*, in terms of accuracy (Goldsmith, 2006), which he sees as a ‘practical’ consideration, as opposed to the commonly used recall/precision measure. A gold standard of 15,000 words with the correct morphological analyses was created. A positive value is assigned to the analyses of words that correctly matched the gold

standard analysis. Of the first 300,000 words of the Brown corpus, the system achieved an accuracy of 72%.

2.2.2.2 Constraint Based Incremental Learning

Cavar *et al* (2004, 2005) apply Alignment Based Learning (ABL) algorithms for grammar induction to unsupervised induction of morphological rules and lexicons. The approach differs from other MDL approaches in that it is an incremental generation and induction of the grammar, word by word, rather than one-off generation and revision, which is computationally intensive. Another key feature of their work is that there is no built-in knowledge in the system such as the type of morpheme, as with Goldsmith's *Linguistica*. In the design of the algorithm, consideration is given to computational, cognitive and linguistic aspects for optimisation. Each iteration of the learning algorithm is divided into three steps: (i) In the *ABL Hypothesis Generation* step if a morpheme (restricted to independently occurring morphemes) is a sub-morpheme of an input word, the edges of the morpheme are considered to be the morpheme boundaries of that word, and a hypothesis is generated with the morpheme along with the affix morphemes. (ii) The *Hypothesis Evaluation and Selection* step uses a number of criteria to decide the credibility of a valid hypothesis. (iii) In the grammar extension step, for each valid hypothesis a signature is created, similar in structure to Goldsmith (2001), and is merged with an existing signature of the same base word. This extended grammar is final and not revised except in subsequent input iterations.

The metrics used in hypothesis evaluation are of key consideration. Three information theoretic metrics, Mutual Information (MI), Description Length (DL) and Relative Entropy (RE) are used to optimize the size and efficiency of the generated grammar. They are motivated by cognitive aspects of languages and grammar, adding constraints which limit use of available memory resources; the metrics application process is fast, computationally efficient and results in a grammar which minimizes space usage. MI predicts the number of bits needed to the left and right of a morpheme. This value is maximized to prefer hypotheses with more segmentation. This in turn is countered by the other criteria to prefer fewer morphemes. RE measures the cost of adding a hypothesis to the grammar by minimizing the divergence of a particular hypothesis from the grammar. The notion of DL is similar to the case Goldsmith (2001) i.e. to be

able to determine for each hypothesis if the new grammar would have greater or smaller length. Besides these metrics, further criteria are used to refine the evaluation: boundary morpheme frequencies are used to detect a potential boundary for a word; hypotheses with longer morphemes are preferred in order to prevent a degenerate state where each letter becomes a morpheme candidate. Certain weights for the evaluation criteria have to be set arbitrarily although the authors argue these could be learnt in an unsupervised way with further research.

Two types of evaluations were carried out by the authors: evaluation of the signature/rule and of the morphological parse of words. For the former, each rule was evaluated to determine whether it contained correct morphemes and stems. The best F-score for the rule set was 80% on a large portion of the general fiction section of the Brown corpus. For the latter, the parsed words of the initial input "under certain circumstances" had precision of 100%, but a lower recall of 60%.

2.2.3 Syntax and Semantics

2.2.3.1 Latent Semantic Indexing

Schone & Jurafsky (2000, 2001) argue that it is impractical to rely on orthographic and phonological features alone for morphology induction. Such approaches would incorrectly produce the analysis *all+y* as opposed to *-ally* and not cater for spelling changes by analysing for example, *hated* as *hat+ed*. They propose to incorporate semantics to aid in the induction of morphology. In their initial work (Schone & Jurafsky, 2000), they first identify and extract potential affixes. Although only suffixes are dealt with in this work, their approach is extendable to include prefixes and circumfixes; also unlike some previous work they do not ignore capitalization. They too have a two-step approach which first identifies a potential set of affixes and later apply their semantics approach to pairs of words. For the first step they build for all words what is called a *character trie*, which is a treelike data structure, the nodes of which are characters with edges linking the characters in each word. Morpheme boundaries are identified where branching occurs, i.e. a variety of subsequent character nodes is observed. Once they have identified morphemes, they pair words having the same stem

but different affixes (e.g. *car/cars; care/cares*). Thereafter they use Latent Semantic Analysis (LSA) as a means to induce the level of semantic relatedness between the words in each pair. The technique identifies semantically related word pairs which are morphological variants of each other. They apply their work to English and compare with Goldsmith's *Linguistica*, showing comparable performance using their semantics-only approach.

In follow up work (Schone & Jurafsky, 2001), they extend their model to incorporate syntactic and orthographic features. While applying semantics three further cues, affix frequency, syntactic context, and transitive closure are included. Their model now incorporates prefixes and circumfixes, and they apply their extended algorithm to three languages, German, English and Dutch. First, the semantic probability of word relatedness is augmented with the orthographic probability of circumfixes (including prefixes and suffixes) measured using affix frequencies; thereafter the probability of the syntactic context for each morphologically related pair is incorporated into the formulation. Each of the additions progressively improves the performance with the best F-scores obtained being 88.1%, 92.3% and 85.8% for English, German and Dutch, respectively. The results are better than any other system of that time.

2.2.3.2 Mutual Information

Baroni *et al* (2002) use similar approach to Schone and Jurafsky, although instead of LSA, they use mutual information to infer the semantic relatedness between pairs of words. For orthographic similarity they measure the minimum edit distance between words. Their model does not assume any kind of morpheme concatenation nor do they incorporate distributional data of word sub-strings such as affix frequencies etc.

2.2.4 Feature-based Classification

De Pauw & Wagacha (2007) and De Pauw *et al* (2007) adapt a machine learning methodology to learn the morphological relatedness of words. They consider words to be composed of features of initial, terminal and middle substrings. Using a maximum entropy classifier they build lists of related nearest neighbour words based on orthographic relatedness. The idea of using features is that common orthographic

features amongst words would be given lower weight while features that occur less frequently get higher weight thus potentially identifying a stems morpheme. This approach, the authors argue, has greater ability to capture long range dependencies between words than other approaches such as minimum edit distance, as used by Baroni *et al*.

2.2.5 Irregular and Non-Concatenative Morphology

Yarowsky & Wicentowski (2000) describe a lightly supervised technique for irregular and non-concatenative morphology induction from a large corpus. Their first goal is to learn from data a table of alignments mapping inflected forms to their roots. Thereafter, using this information they train a morphological analyser capable of performing automatic morphological induction. Some language specific resources are needed for the procedure: inflectional part-of-speech categories and corresponding canonical suffixes; a dictionary of noun, verb and adjective roots along with an approximate way of tagging words in the corpus; and finally, a list of consonants and vowels of the language.

The main challenge addressed by Yarowsky & Wicentowski is to correctly align an irregular form with its root, e.g. *sang* with the root *sing* rather than the regular inflected form e.g. *sanged*. An obvious approach would be to just consider their ratios of occurrence in the corpus; for example, *sang/sing* with ratio 1.19/1 as compared to *sanged/sang* with ratio 0.007/1. However at times this can be misleading since some inflectional forms of words occur rarely. In order to deal with this the authors calculate the (smoothed) distributions of ratios over an entire class of inflected/root forms of words. For example, for the class VBD/VB (Penn Treebank tags for Past Tense Verb/Verb Root) the smoothed distribution, *log*(VBD/VB), is calculated and for each pair like *sang/sing*, the log ratio value indicates whether it fits the distribution well or not. This distribution is not obtained at the outset, as initially the alignments of irregular/root forms are unknown. The authors observe that the distribution of alignments of regular/root forms is similar to the irregular/root forms, so they initialize with statistics of simple suffix stripped and inflected forms. This is naturally noisy, but as the discovery of irregular forms progresses, the distribution improves. Other

distributions between ratios of inflected forms are also considered e.g. VBG/VBD (where VBG is a tag for gerund/particle ending in *-ing*).

Two additional cues used to identify related forms are distributional and orthographic similarity. Weighted context vectors representing each word are compared to other word forms using the cosine similarity measure. The authors argue that morphologically inflected words have more similar contexts than synonym words. Further, they use the weighted Levenshtein edit distance to gauge orthographic similarity between words, assigning higher cost to consonant changes and lower cost to vowel changes. The end application is a morphological analyser, estimated using an interpolated back-off model, which predicts a stem change, given a root, suffix and POS tag. Although each individual metric discussed above does not on its own perform well, combining all the metrics together results in an effective morphological analyser giving accuracy of 99.2% over all evaluated words (including irregular).

2.2.6 Complete Language Independence

Hammarström (2007a) presents an exhaustive survey of research on Unsupervised Learning of Morphology (ULM). Based on this survey, he makes some key observations about previous research in this area:

- Seemingly due to lack of awareness, a lot of work by different researchers has gone on in parallel streams. The same or related ideas have been pursued by different researchers with little sharing.
- There have been lots of experimentation and heuristics proposed without sound supporting models or theory.
- Most approaches are built with the aim of applying to a certain language or group of languages. These approaches are governed by language specific parameters and thresholds that need to be determined in a supervised manner.

Based on these observations, Hammarström proposes a model for concatenative morphology to overcome shortcomings in past research, which aims to cater to a topologically diverse set of languages, without the incorporation of language specific

constants and parameters. The aim is to build a theory side-by-side with reasonable experimental results and not just to aim at good results without explanation.

Hammarström (2006a) proposes a formalism which he calls a 'Naive Model of Affix Extraction'. It is naive in the sense that it does not take into consideration the intricate affixational requirements of the different languages of the world. The formalism is based on the intuition that affix strings (he focusses on suffixes) occur in a corpus with much higher frequency than stem or base strings; this asymmetric relationship between base forms and suffixes can be exploited. Two main underlying assumptions are made: (1) Arbitrary Character Assumption (ACA) which states that a character is equally likely to occur in any word-position of the base or suffix string; (2) Frequent Flyer Assumption (FFA): the members of the set of suffixes are very frequent. The algorithm for identifying suffixes makes use of three properties of suffixes: *Frequency*, *Curve Drop* and *Random Adjustment*. All terminal segments and their respective frequencies are recorded. The Curve Drop property is then used to see which of these segments is well-segmented to the left i.e. *-ing* and not *-ng*. Random Adjustment is used to distinguish frequent but random segments such as *-a* from non-random segments (like *-ing* or *-ng*). Finally the three properties are combined to give a score to each segment. A ranked list is produced with suffixes at the top and incoherent segments at the bottom. Exactly where the demarcation occurs between suffixes and such segments is a difficult problem to solve.

Hammarström (2006b) applies the approach to the problem of same-stem word recognition, which is an easier problem than having to accurately extract suffixes. He uses a metric based on co-occurrence statistics to quantify which end-segments are prone to attach to the same stem. The technique achieves very good results when applied to four topologically diverse languages. The author has also successfully applied his affixation approach to the problem of language identification (Hammarström, 2007b). Unlike previous approaches to this task which can analyse text in the range of 100 characters or more, in this work the author builds a more fine-grained model which can accurately classify a one-word input and even classify concatenations of words from different languages. "Competitive" accuracy is reported in experiments on a multilingual parallel Bible corpus.

2.2.7 Conclusion

This section has covered influential lines of work in unsupervised morphology learning. As stated earlier, the desired outputs differ considerably with some systems outputting segmentations, some identifying affixes etc. This lack of agreement makes it difficult to make reliable comparisons across different approaches. Much progress has been made in achieving high accuracy in unsupervised learning which is comparable to supervised systems. The MDL-based approach has gained much popularity; however, as pointed out by Hammarström (2007a), this technique lacks a sound theoretical basis; and on the experimental side, the use of thresholds and constants mars the success of reported results. A sound model for unsupervised morphology learning with a solid theoretical basis is yet to emerge.

2.3 Computational Morphology of Arabic and Semitic Languages

Semitic languages, for example Arabic, are challenging to process automatically. This is due to several reasons including: rich morphology; ambiguity in the writing system due to omitted diacritics; complexity of the way roots and patterns combine to form a word; and lack of standardized encoding schemes. Most work in Arabic computational morphology so far has been built on knowledge-based, linguistic foundations and targeted only for Modern Standard Arabic (MSA). Tools constructed using this approach are expensive to build and cannot be easily adapted to other languages or dialects. The need for data-driven machine learning approaches is pronounced for such languages given the large number of variant dialects.

This section surveys the various empirical techniques that have been used to learn Arabic and other Semitic language morphology. It starts by reviewing work on supervised approaches, followed by unsupervised techniques. For concatenative morphology, most of these techniques inherit from the general approaches of previous research with minor adaptations. The real challenge is to address non-concatenative morphology in order to identify the root and pattern from a given word and to simultaneously deal with concatenative morphology.

2.3.1 Supervised and Semi-Supervised Approaches

2.3.1.1 Language Model (LM) based Arabic Word Segmentation

Lee *et al* (2003) use bootstrapping to incrementally update a language model (LM) for best segmentations of a word into morphemes (prefix*-stem-suffix*), starting with a small manually segmented corpus and a table of prefixes and suffixes of the language. Although Lee *et al* don't treat infixes, they segment into multiple prefix/suffixes as opposed to one prefix and/or suffix per word. This is important for applications such as machine translation, since almost every morpheme is meaning-bearing, having corresponding words in another language. The input to the algorithm is a sentence. Each token of the sentence is analysed in sequence. For each token all possible segmentation scores are computed using an initial trigram language model. The segmentation with the highest score is selected. At token boundaries, morphemes from the previous token are used as histories for the subsequent token morpheme. Unseen stems are classified as 'unknown'. Possible segmentations of a word are restricted to those derivable from a table of prefixes and suffixes, obtained from the initial corpus. Derivation of sub-segmentations of matching prefixes/suffixes enables the system to identify possible segmentations which would have been missed out otherwise. However, there is some level of filtering (called PS-Filter) which detects illegal segmentations. For example, sub-segmentation of the whole prefix *Al-* into *A-* and *l-* is illegal and hence ignored.

The way the algorithm works is that, starting with an initially segmented corpus and vocabulary, a language model based segmenter is built to segment subsequent partitions of the unsegmented corpus. The training corpus is divided into a number of partitions. At each iteration, the current segmenter is used to segment the next partition, thus acquiring new stems and adding new words to the vocabulary. The new segmenter is built using the enlarged vocabulary. The algorithm selects the final segmenter and vocabulary such that the next partition does not yield further improvement. New stems are acquired based on three criteria: (i) frequency threshold, (ii) filtering of stems containing substrings having high probability of being a prefix, suffix or prefix-suffix, and (iii) contextual filtering, which filters out stems with probability of occurrence of prefix/suffixes being greater or lesser than certain thresholds.

The authors evaluate performance using Word Error Rate (WER) on 28,449 words extracted from a test corpus. As a baseline, each word is assigned a segmentation which most frequently occurs in the training corpus. This gives a WER of 26%. Using only a trigram LM for segmentation, the WER reduces to 14.7 (an improvement of about 50%). Augmenting this trigram LM with the PS-Filter and the three criteria for new stem acquisition further improves the accuracy by about 30%. Some segmentations require the token's Part-of-Speech (POS) to be known. Hence the authors adjust the model to accommodate sub-string POS probabilities into the model. They achieve an improvement of 10% (WER of 2.9% to 2.6% for 110K word training corpus). Overall they report 97.3% accuracy which is comparable to state-of-the-art performance of the time.

2.3.1.2 Constraint Based Learning

Daya (2004) applies a machine learning approach to identify roots for Hebrew, and extends the approach to Arabic (Daya *et al*, 2008). They use a multi-class classifier, SNoW, to build three classifiers for each of three root radicals in triliteral roots. They chose features having grammatical and statistical characteristics such as character location, character bi-grams, prefixes and suffixes. To train their classifiers they used a development set of 4800 words extracted from a corpus of 15,000 words and manually annotated with root information. Two baselines were built. Baseline 'A' was a single multi-class classifier attempting to learn the whole root at once. This was inaccurate given the large number of target roots and sparseness of the training data. Baseline 'B' was a combination of three classifiers, one for each consonant of the root. The target space for each classifier is now reduced to 22 (the number of letters in Hebrew) for which there is ample training data. Since the classifiers are combined straightforwardly and independently without considering interdependence of the root radicals, this too is inadequate. In order to account for this, they chose an HMM to model the sequential occurrence of the three consonants. The probability of the three consonants in sequence is now maximized given the word and model. But this too is simplistic and does not capture morpho-phonological alterations (such as assimilation and metathesis) from the root to the surface form; nor does this model irregular pattern formations. Also there are phonological constraints that limit the possibility of certain root formations. All these

linguistic constraints have to be accommodated for. The number of possible targets in the classifier is further reduced given the (linguistic) fact that (almost always) only consonants that occur in the inflected form occur in the root along with a few weak radicals that occur in different consonant places. The model with these classifiers becomes a new baseline for the extensions that follow.

Further refinements are applied not to the classifier but rather to re-rank a ranked list of plausible roots that are output by the SNoW system for each word. Each root is assigned a confidence score based on the soundness of its formation assigned after applying some linguistic checks/constraints. A further measure taken into consideration is the inverse edit distance between the roots and the word. Thus three scores are combined (equally, by taking their product) to obtain a new ranking: (i) product of the three classifier outputs, (ii) confidence scores, and (iii) inverse edit distance. The top ranking roots are returned as output. Also, multiple roots are retrieved for some words whose scores are higher and close to each other. This boosts recall while minimally decreasing precision. Overall, the authors report 80.90% precision, 88.16% recall and F-score of 84.38% for held-out data. This performance is comparable to performance by Hebrew speaking human subjects (F-score of 81.86%) who too have difficulty in extracting correct roots from words.

Daya *et al* extend the approach to Arabic, for which the problem is somewhat more difficult than for Hebrew: the number of letters is greater, hence the number of targets (40 in Buckwalter transliteration) is greater; more patterns and infix letters make the linguistic constraints more complicated; and the average number of ambiguous roots per word is much greater. One advantage over Hebrew is that training data is more abundant. Although the linguistic constraints are more simplistic, the system still achieves only slightly inferior performance to Hebrew with precision of 78.21%, 82.80% recall and F-measure of 80.44%.

2.3.1.3 Automatic Morphological Analysis

Darwish (2002) describes the development of an Arabic morphological analyser called Sebawai, which he later enhanced (Darwish & Oard, 2007). It learns a probabilistic model for combining affixes with stems based on the output of an existing Arabic

morphological analyser, ALPNET (Beesley, 1996). It derives the rules and statistics to estimate the occurrence probabilities of templates, prefixes, and suffixes. It is trained on a list of word-root pairs to first derive the templates that produce stems from roots. Thereafter, a list of prefixes and suffixes is generated. Finally, by estimating the probability of occurrence of templates, stems, and roots, the system is able to output a suitable analysis for a word. The author reports accuracy of 84% in extracting the correct root of a word.

2.3.1.4 Finite State Transducers

Clark (2001, 2002, 2007) experiments with memory-based algorithms for learning the morphology of a language with the aim of understanding human acquisition of language. He first builds a supervised model to address the problem of associating base with inflected forms, and then enhances the model so that it can be used with semi-supervised learning. The choice of Arabic as a test-bed was to study modelling of the complex phenomena of non-concatenative morphology which can be best exemplified by the Arabic broken plural.

Clark approaches the problem through the use of finite state methods which are able to model all morphological processes though with added extensions to accommodate for non-concatenation. The model used is a non-deterministic stochastic transducer, defining a joint probability over input and output distributions. The model attaches the output function to states rather than transitions, bringing it close to a type of Hidden Markov Model (HMM), called a Pair HMM (PHMM). This resemblance to HMMs allows them to be learnt in the same manner as HMMs. An adaptation of the Expectation Maximisation (EM) algorithm is used with extensions to the algorithm to accommodate all possible combinations of input and output strings. The trellis data structure in the Viterbi learning is extended to three dimensions, with two dimensions for the two inputs and one for the position. Unfortunately, sometimes the EM training is not effective, with the model converging to a local maximum, meaning that the most likely state transition sequence is not the most likely output. Although empirically the model works well for simple cases, a better approach is to infer the conditional probability distribution of the output given the input from the joint probability of the input and output strings and maximizing over the random samples. Another

complication is that a single large model models all possible input/output combinations which is inefficient, requiring a large parameter space. A more appropriate strategy is to use mixtures of models for each morphological paradigmatic class, which can then be parameterised easily. Clark therefore subdivides the training data into classes and builds a model for each class before mixing them. In order to extend the work to make it semi-supervised, the author takes input as two lists of unaligned inflected and base forms. This can be viewed as a permutation of the two lists having *n!* alignments, and can be modelled as a hidden layer with n^2 parameters. Using the EM algorithm, the permutation and string transduction can be simultaneously optimized.

For evaluation of the semi-supervised approach Clark used two types of datasets. The first (PN1) simply consisted of all singular forms in one list and all plurals in another. For this set near perfect alignments were obtained with precision and recall of 96.8% and 95.5%. A second more realistic set (PN2) consisted of lists with half the words randomly removed from each list resulting in half the number of words with correct alignments and the rest left unaligned. For this dataset, the system achieved alignments with precision and recall of 84.1% and 65.1%, respectively.

2.3.2 Unsupervised Learning of Arabic Morphology

2.3.2.1 Constraint Based Learning

Rodrigues & Cavar (2005, 2007) apply their earlier work (Cavar *et al*, 2004, 2005) to induction of Arabic morphology. They have a two-tiered approach to dealing with the complex morphology of Arabic. In the first phase they deal with identification of the root and in the second they deal with concatenative morphology in the usual way. Identification of the root in Arabic is a difficult task due to the complex system of variation of the root word into many variant patterns, with the help of short vowel changes and infixes. The authors apply a heuristic unsupervised approach for identification of roots. The heuristics identify root letters by assigning them a confidence score. There are two parts to the score, the positive evidence which is normalized by the negative evidence. These are calculated in terms of frequency of occurrence of potential root letters and affixes. The intuition behind this measure is to

capitalize on the promiscuity of roots as compared to vowel templates. Root templates are a more frequent open class, whereas vowel templates are a less frequent closed class. Some restrictions and constraints are applied (in a supervised way) to the root learner in order to speed up the algorithm. The authors restrict themselves to identifying only three letter roots, which is the most common form of verbs in Arabic. Also, they constrain their search for roots by requiring a maximum distance of five characters between the first and last root radicals, and of three characters between each radical, thus excluding unlikely character combinations. Once the roots have been identified, the characters from first to last in the radical are replaced by a symbol and any occurrences of characters around the symbol are assumed to be concatenations which are then dealt with in the second phase of the algorithm in the same manner as in their previous work (Cavar *et al*, 2004, 2005).

Rodrigues & Cavar tested their approach on 10,000 words having prefixes, suffixes and infixes, containing only triliteral roots, generated by the Buckwalter Morphological Analyser. Quantitative results were only obtained for the root identification part. The system reaches 75% precision after 10,000 words. They observe that incremental learning, with longer words input first gives higher final precision. Also, clustering by length and frequency of the words revealed distinct categories of open and closed class words.

2.3.2.2 Parallel Corpora (Concatenative Morphology)

Snyder & Barzilay (2008) harness the connection between languages through parallel corpora in order to learn morphology of three major Semitic languages, Arabic, Hebrew and Aramaic. They show how cross-lingual parallelism can be utilized to improve morphological segmentation without any supervision. Furthermore they investigate how the outcome is affected by languages of the same or different families. While researchers have in the past exploited parallel corpora for various linguistic tasks including morphology, they have done so in an asymmetric, supervised way using annotations in one resource-rich language to induce information in another. In contrast, Snyder & Barzilay build one multilingual model simultaneously capturing the structural regularities in each language without any supervision. Advantages from a joint analysis are that structural regularities and irregularities which occur between languages such as

prepositional morphemes attached to a word in one language can be identified by corresponding missing or detached occurrences in another language; cognates in two languages would tend to align, splitting off any attached affixes.

The authors apply a hierarchical Bayesian model to capture multi- and monolingual dependencies between two languages, extendable to multiple languages. Distributions need to be identified over two types of morphemes, (i) *stray*, which are ones that occur in one language and not in the other, and (ii) *abstract*, which are morpheme pairs in two languages that may be cognates or share syntactic and semantic properties. The distributions over all finite-length *stray* morphemes in the respective languages are modelled using a Dirichlet Process (DP) having a base prior distribution encoding two properties of the morphemes: the morpheme length and the end-morpheme character. The distributions over all possible pairs of finite strings (from respective alphabets) of *abstract* morphemes are also modelled using a DP having a base prior distribution encoding the lengths of the component morphemes. In the case of related languages with known phonetic correspondences between alphabets, string-edit distance between the correspondences can also be used as a parameter for capturing cognate resemblance in the prior distribution. The advantage of using a DP is that it concentrates most of the probability mass on a small number of morphemes/morpheme pairs while still reserving a small proportion for all other possible strings/string-pairs. Once these two distributions are obtained, the next phase is to generate parallel phrases using a generative model. This is a four step process: (i) draw the counts of *abstract* and *stray* morphemes in each language from a Poisson distribution, (ii) draw the *abstract* and *stray* morphemes according to their counts from their respective DP distributions, (iii) order the morphemes using a uniform distribution over all permutations of morpheme orderings, and finally (iv) fuse the morphemes into words using again a uniform distribution over all permutations of morpheme fusions. The results of this phase are parallel phrases that have morphemes that have been implicitly aligned. The final step is to obtain a segmentation of the morphemes having high joint-probability marginalizing over all possible draws from all three distributions. This is achieved using Gibbs sampling.

For evaluation, the authors use two baselines: (i) a state-of-the-art system, Morfessor (2007), and (ii) monolingual segmentation obtained using monolingual morpheme distributions. They evaluate the bilingual models with and without character-to-character morpheme correspondences. In the former case, they obtain a marked improvement over the monolingual baseline for all models (except one, Hebrew + Aramaic). No difference was observed between adding English (having a different morphological structure) over any other Semitic language (with similar structure). However when character-to-character morpheme correspondences were included, a boost in performance was seen, reducing relative error for Arabic/Hebrew by 24%.

2.3.2.3 Learning Vowel-Consonant Distinction From Phonemes

Goldsmith & Xanthos (2009) learn the vowel-consonant distinction and structure using statistical methods based on phonemes rather than word orthography. Starting with techniques applied by a Russian researcher, Boris Sukhotin, for the task of differentiating vowels from consonants, the authors explore two additional superior techniques for the task. They further apply the techniques to determine vowel harmony and syllable structure. They report excellent results for their applications.

Sukhotin's conceptually and computationally simple approach is based on the assumption that vowels occur more frequently than consonants and that alternation between vowels and consonants is much more frequent than between vowel-vowel and consonant-consonant. To accomplish the task, a square, symmetric matrix is used with rows and columns each representing phonemes in the corpus. The values in the matrix are the counts of the number of times in the corpus that a phoneme in a row occurs adjacent to a phoneme in a column. Next, one vowel is identified by assigning a score to each potential vowel. A candidate vowel phoneme would be one whose difference between its frequency with a consonant and its frequency with a vowel is positive and substantial. This difference is the score assigned to each phoneme; the one with the highest score, being the candidate, is then removed from the matrix. In this way two classes consisting of vowels and consonants is formed. Looking at the poor results obtained on a test set for the English and French datasets, Goldsmith & Xanthos identified two sources of failure: firstly, infrequent phonemes suffer from the problem of data sparsity, lacking diversity of context; and secondly, high frequency consonants

are likely to be classified as vowels since the initial decisions are based on only the overall frequency of the phonemes.

Goldsmith & Xanthos discuss the application of spectral clustering, in which the phonemes are presented as nodes in an undirected weighted graph. They obtain a symmetric square adjacency matrix with values being the 'distributional similarity' between the phonemes. This distributional similarity is calculated by evaluating similarity in the contexts of neighbouring phonemes. Thereafter spectral analysis is applied to the graph obtaining its second Eigenvector (Fiedler vector). This vector assigns a single value to each node of the graph, i.e. each phoneme, such that similar phonemes get similar values. This has the effect of grouping similar phonemes along different points on the linear scale. Considerable performance improvements were observed over Sukhotin's algorithm, yet misclassification of phonemes were still observed.

The final computational technique discussed by the authors is based on an HMM with two states, one for each class. Each state has a probability distribution across every phoneme it generates and a distribution over transitions to itself, or to the other state. The aim is to determine these two distributions such that the probability of the dataset is maximized. The Baum-Welch EM algorithm is used to estimate these distributions, guaranteeing a local maximum. This suffices, as only the local structure of words is being evaluated. The idea is that consonant-vowel variation will result in a different distribution of phonemes in each state. That is, if there is a tendency of a phoneme to alternate (i.e. between vowels and consonants) the two groupings would be expected to be divided such that the emission probabilities for one of the sets will be higher in one state than the other. Also with this tendency we would expect the transition probability between the two states to be higher than the transition probability between the same states. The results of experiments on both English and French confirm this hypothesis. Vowels and consonants of each language indeed converge perfectly to the two sets having different emissions in each state.

Xanthos (2007) applied Sukhotin's algorithm to a symbolically transcribed wordlist for Arabic. The transcriptions thus included short vowel symbols (represented by diacritics) in the phonetically transcribed text given as input to the system. Thus, given a word, as

a sequence of phonemic symbols, Xanthos's system (named *Arabica*) attempts to decompose the transcription into a root and pattern, and also identifies the rules that govern their combination. It first applies Sukhotin's algorithm, as described above, to identify vowels and consonants based on their distributions. Starting with the simple assumption that the root of a word is a sequence of consonants and a sequence of vowels is a pattern, the system looks through the dataset for regularities in the combinations of roots and patterns. That is, it tries to identify roots that consistently combine with certain patterns. Once it finds that a certain set of roots combine with a certain set of patterns, words are assigned a structure known as an RP-Structure, which can be thought of as a rule for combining certain roots with certain patterns. Words that are not assigned an RP-Structure are left unanalysed and their hypothetical roots and patterns are discarded. Next the system tries to extend its set of roots and patterns, by identifying a set of unanalysed words that correspond to a particular RP-Structure. These words are added to the structure, provided that the integration simplifies the morphology, gauged using the Minimum Description Length principle as introduced by Goldsmith (2001). The algorithm terminates when there are no unanalysed words left.

2.4 **Conclusion and Prospective Work**

There is very little reported research on computational approaches to processing non-concatenative morphology. Most work on unsupervised morphology learning has been targeted towards European languages, in which non-concatenative morphology is almost non-existent. For Semitic languages, many of the same techniques can be adapted for obtaining word segmentations. However, complete analysis of Semitic languages including both root-and-pattern morphology and segmentation of words is a difficult problem due to the morphological richness of the languages. This is evident in the work of Daya (2004) and Rodrigues & Cavar (2005) who manually encode many linguistic constraints and restrictions pertaining to a language. Arguably the best attempts to analyse the non-concatenative morphology of Arabic to date has been presented by Xanthos (2007), who presents an approach to statistically learn phonological categorizations of roots and patterns without any linguistic knowledge. Here too there is a restriction on the input to only non-inflected words. Simultaneously

dealing with concatenative and non-concatenative morphology is a problem that has been little explored. The divide-and-conquer approach dealing with concatenative and non-concatenative morphology separately, as adopted by Rodrigues & Cavar (2005), is a plausible way to obtain a complete analysis. One may be tempted to apply concatenative analysis and then deal with root identification; but as shown by Rodrigues & Cavar (2005), it is more efficient to deal with concatenation once the root is identified. No previous research appears to have addressed root identification for unvowelled text, which is the naturally written form of text with short vowels omitted.

In the area of unsupervised learning of the complex morphology of Semitic languages, many problems remain to be addressed. A framework to represent the common morpho-phonological alterations that occur amongst the various languages in this family needs to be formulated. Based on this, a model could be presented defining and learning parameters shared by these languages. Root and pattern categorization using spectral analysis and HMMs also seems a promising prospect that has so far not been explored. A practical tool is needed for analysis to deal with naturally occurring unvowelled text, outputting roots and patterns.

Chapter 3

Maximum Entropy Based Learning

3.1 Introduction

The past two decades have seen machine learning techniques applied to a wide range of tasks in natural language processing (NLP). Computational power has improved greatly making it possible to learn predictive models from vast amounts of information. Maximum Entropy (ME) modelling is one such statistical modelling technique which learns the most uniform model (having largest entropy) over data given the constraints.

Although it is a supervised learning methodology, ME modelling has been adapted to do unsupervised learning to learn morphological relatedness between words in an unannotated corpus. This approach has been pioneered by De Pauw & Wagacha (2007) and De Pauw *et al* (2007) who apply the adapted methodology to learn morphological relatedness for under-resourced languages exhibiting concatenative morphology.

3.1.1 The Approach

I approach the morphology induction problem by first deriving a morphological analyser consisting of two lexicons: a root lexicon and a pattern lexicon. The method for developing the lexicons is itself divided into two procedures. First, use the ME based machine learning approach to induce groupings or clusters[4] of words with orthographic similarity between words in terms of the two kinds of morphemes: roots and patterns. Second, extract the morphemes from the clusters, which are identified on the basis of how the words are related, whether by pattern or by root. This manner of morpheme identification is similar to the work of De Pauw & Wagacha (2007) who apply it to extract prefixes of words exhibiting concatenative affixation.

[4] Cluster here refers to a collection of words related in terms of morpheme types, without referring to application of any clustering algorithm.

The output of the first step to obtain morphologically similar word collections is comparable to obtaining orthographic similarities between words using Minimum Edit distance metrics as used by Baroni *et al* (2002). Two simultaneous models are built: one model abstracts roots based on orthographic properties for each word, and is used to derive the root-based word clusters; the other model represents pattern based features in order to derive pattern-based clusters of words.

Using machine learning, De Pauw & Wagacha (2007) capture dependencies between orthographically distinct words which are not identifiable by the Minimum Edit Distance approach. In the work described below, I present a model based on orthographic features for approximating word similarity; this considers two different types of morpheme features to obtain word similarity in terms of roots and patterns.

From the morpheme based clusters resulting from the previous procedure, the next step extracts the morphemes from the clusters, which are identified on the basis of how the words are related, whether by pattern or by root. Two lists/lexicons are thus obtained for pattern templates and for roots, with each entry ranked according to its plausibility. These lexicons constitute the induced morphological analyser which is applied back to the vocabulary, analysing each word to obtain its root and pattern template.

3.1.2 Chapter Organization

The approach to unsupervised lexicon induction based on Maximum Entropy (ME) modelling is explained in section 3.2. The section contains a brief introduction to ME modelling (3.2.1), followed by the feature selection process (3.2.2); thereafter model training with different possible settings is discussed in detail (3.2.3) with a final discussion about model application (3.2.4). The next major phase is lexicon extraction which is described in section 3.3, covering the method for weighting the morphemes and different scoring methodologies. Section 3.4 describes the morphological analysis and section 3.5 presents an evaluation. Finally, the overall design of the system for unsupervised learning and conclusion are given in section 3.6 and 3.7, respectively.

3.2 Morpheme-Based Clustering

3.2.1 Maximum Entropy Modelling

The main goal of machine learning is to make predictions about previously unseen cases or phenomena by generalizing from (incomplete) available data about the random process; this is known as a model of the data. Thereafter, using this model, predictions about future occurrences of the phenomena are made. There are two main tasks to be accomplished: firstly, the acquisition of useful facts about the data – this is called feature selection; and secondly, choosing a good representation by doing model selection.

For the problem of morphology induction, I approach the goal of data prediction using a modelling approach based on the Maximum Entropy (ME) principle. The ME framework is able to represent unbounded problem-specific knowledge that is interdependent and overlapping which, unlike some other machine learning paradigms, such as Naive Bayes, does not require the features to be independent. For example, in problems where classification decisions are made in a sequence, like parsing and tagging, it is possible that for the task of classification, the models would use the previous classification decisions that have been taken in the sequence. Other than that there is great diversity in the nature of features that is possible to incorporate, where the contribution, or weight of each feature is determined by a scaling process. Thus it is well suited to modelling morphological processes, where the morphological features derived from a word, e.g. *book*, could have the features for example, *@b, @bo, @boo, ook#, ok#, k#, o, oo*. Here the features are overlapping since they have overlapping characters with some features incorporating boundary markers to indicate context. These features are clearly not independent of each other.

The basic idea behind maximum entropy modelling is to choose the most uniform model of the data given a set of constraints (which may be independent or overlapping); or, in other words, to model that which is known while not assuming anything about that which is not known. The ME model built for the morphological features as given in the above example would generalize over features which are infrequent yet occur more frequently than chance, representing the base forms or lemmas of words. The next two

subsections give an overview of ME modelling which has been adapted from the descriptions of Berger *et al* (1996) and Ratnaparkhi (1998).

3.2.1.1 Modelling

Let x be the input of a particular random process from a set of all possible inputs X that produces y as output from a set of all possible outputs, Y. The aim is to produce a model that would learn the conditional probability, $p(y|x)$ i.e. to predict with what probability we expect to see the output y given x as the input. Assume a random process that produces an output y from a set of possible outputs. The building blocks of the model are the examples of x and y in the training data. For each input-output pair of a large number of samples (totalling N) from training data, (x_1, y_1), (x_2, y_2),… (x_N, y_N), the expected probability of occurrences of each pair is calculated as

$$\tilde{p}(x, y) = \frac{1}{N} \times \text{number of times } (x, y) \text{ are seen in the data} \tag{3.1}$$

This value will typically be low, especially in the case where input variables and output classes are large, where it would be close to zero for most cases. Let f be a function, called **feature function** or **feature** for short, that denotes the presence or absence of a pair (x, y),

$$f(x, y) = \begin{cases} 1 & \text{if } x \text{ and } y \text{ are found} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

A feature is thus an individual measurable heuristic property of the phenomenon being observed. The expectation of f with respect to the empirical probability distribution, $\tilde{p}$, would then be

$$\tilde{p}(f) = \sum_{x,y} \tilde{p}(x, y) f(x, y) \tag{3.3}$$

The probability that the model assigns to each feature, f, with respect to the model $p(y|x)$ is given by the expected value,

$$p(f) = \sum_{x,y} \tilde{p}(x)p(y|x)\, f(x,y) \tag{3.4}$$

where $\tilde{p}(x)$, is the distribution of x in the training data. This model expectation is equated to the expected value of f in the training data:

$$p(f) = \tilde{p}(f) \tag{3.5}$$

$$\sum_{x,y} \tilde{p}(x)p(y|x)\, f(x,y) = \sum_{x,y} \tilde{p}(x,y)\, f(x,y) \tag{3.6}$$

Thus the model of the process, $p(y|x)$, has been constrained to considering only those cases which are in agreement to the training data where the output has the feature f. The equation is referred to as the **constraint equation** or simply **constraint**. Hence, any new knowledge can be incorporated in the model by constraining the expected value the model assigns to the corresponding feature as in (3.5) and (3.6).

3.2.1.2 Principle of Maximum Entropy

Assume a set of features f_i for $i = 1,2, \ldots n$, each imposing a constraint C_i and having probability, $p(f_i)$. In order to make the model conform to these features seen in the training sample, we have from (3.5), $p(f_i) = \tilde{p}(f_i)$. As stated earlier, the aim of ME modelling is to find a uniform model, $p \in C$, for $C = C_1 \cap C_2 \cap \ldots C_n$, which is the set of allowable models confined to $\tilde{p}(f_i)$. A measure of uniformity over the conditional distribution $p(y|x)$, is provided by *conditional entropy*, thus

$$H(p) \equiv \sum_{x,y} \tilde{p}(x)p(y|x) \log p(y|x) \tag{3.7}$$

A unique model p_*, which gives the maximum entropy from the set C of possible models is selected as the best representative,

$$p_* = \max_{p \in C} H(p) \tag{3.8}$$

This is a problem in constrained optimization, wherein we try to find p_*, which gives the maximum conditional entropy of all models $p \in C$. The method of Lagrange multipliers from theory of constrained maximization has been used by Pietra et al. (1995) . The main steps are outlined below.

The solution for model $p_* \in C$ can be obtained using a parametric form $p_\lambda(y|x)$ deduced using a Lagrangian function,

$$p_\lambda(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n} \lambda_i f_i(x,y)\right) \tag{3.9}$$

where λ_i , the Lagrangian multiplier, is the weight of each feature, f_i , and $Z(x)$ is the normalization factor, or the partition factor ensuring $\sum_y p_\lambda(y|x) = 1$ for all x. The probability distribution of the form shown in (3.9) is the one that is closest to $\tilde{p}$ in terms of Kullback-Leibler divergence, when subjected to the set of feature constraints.

Given an exponential model with n features and a set of training data (empirical distribution), the next step is to do parameter estimation: find the associated real-value weight, λ , for each of the n features, which maximize the model's log-likelihood $L(p)$,

$$L(p_\lambda) = \sum_{x,y} \tilde{p}(x,y) \log p_\lambda(y|x) \tag{3.10}$$

The optimized weight, λ_*, with respect to the exponential model, p_λ is

$$\lambda_* = \underset{\lambda}{\operatorname{argmax}}\, L(p_\lambda) \tag{3.11}$$

The maximum p_* of $H(p)$ for $p \in C$ is then

$$p_* = p_{\lambda_*} \tag{3.12}$$

There are numerous techniques to find λ_*, for which the solution cannot be found analytically but can only be obtained through numerical methods.

Certain optimization methods are specifically tailored for maximum entropy modelling. One such method is the iterative scaling method designed by Darroch & Ratcliff (1972) which is applicable to the problems where feature functions are nonnegative, which of course is true for a binary-valued feature-function. Another efficient optimization method recently found to be quite effective for ME modelling is the Limited-Memory Variable Metric (LBFGS) of Malouf (2002). More details on these optimization schemes and the methods chosen follow in section 3.2.3.

3.2.2 Morphological Features

Features are encoded as feature functions as described in equation (3.2), referred to as *contextual predicates* in the terminology of Ratnaparkhi (1998). Thus each contextual predicate holds a certain (output) class, of the classification problem, as true if the required set of possible contexts or textual material is observed. With interdependent features, there is a possibility that a contextual predicate may depend on the outcome of another contextual predicate.

3.2.2.1 Feature Sets

There are two feature sets that need to be determined for building two separate models, one for root based clustering and the other for pattern based clustering. Both kinds of models are considered below.

In conventional uses of ME modelling for classification tasks, the problem is usually to classify entities, based on the contexts in which they occur, into a pre-defined collection of output classes. Contexts are encoded as features. In our case, the entities, i.e. vocabulary words, are themselves the output classes and rather than learning a classification of these entities, the aim is to determine the proximity between the classes.

In this approach, there is no consideration of the external context of occurrence of the word entities themselves but rather, the context features are derived from each word's orthography, consisting of characters and subwords with different placements within the word. For example, given an (outcome) word 'WORD', its context may be {B=W, B=WO, B=WOR,I=O, I=R, I=OR, E=ORD, E=RD, E=D} where each feature value is tagged with the position of occurrence of the subwords within the word, such that "B=" is beginning of word, "I=" is inside the word and "E=" is the ending part of the word.

We define a *contextual template* to automatically derive context from each vocabulary word, to be used as its contextual predicate. Further, using adaptations and linguistically motivated heuristics, different levels of feature details can be obtained. These vary from an exhaustive feature set, containing every possible combination of characters derivable from a word, down to a more selective set. The aim is to discover which feature types are contributing towards better efficiency while minimizing the set size to reduce the computational cost of model building and application.

3.2.2.2 Feature Extraction

The template for building the initial feature set for root based clustering is simply to take the powerset combinations of every character occurring in the word; interpreting each element of the set as an ordered sequence of characters – the ordering matching that of the original word. The feature set thus contains the features from a single character to multiple character combinations with contiguous and non-contiguous characters from the word. Non-contiguous character combinations account for the inter-digitation of roots with pattern templates, by bringing the root radicals together from the word where they occur non-contiguously, intertwined with pattern characters. In contrast, for concatenative morphology, the set would be reduced to only considering sequential subsets of character combinations as in the case of the 'WORD' example in

the previous subsection. The explosion of features due to taking the powerset shows how more complicated the task becomes for non-concatenative languages.

The feature set for the pattern-based clustering model follows from the root based feature set: root radicals are replaced by placeholder characters ('-') in the pattern; characters that are dropped from the root features (i.e. from the non-contiguous character combinations), simply appear as affix characters in the pattern template. An example of this template application for root-based and pattern-based features is shown in the third column of Table 3.1. This feature set is referred to as **PS_NBC**.

Word	**Root-Based Feature**	**Pattern-Based Features**
slAm	s, sl, slA, slAm, slm, sA, sAm, sm, l, lA, lAm, lm, A Am m	-lAm, --Am, ---m, ----, --A-, -l-m, -l--, -lA-, s-Am, s--m, s---, s-A-, sl-m, sl--, slA-

Table 3.1: PS_NBC features as powerset combination of word characters without boundary characters

A starting boundary character ('@') and an ending boundary character ('#') are appended to the word before applying the contextual template. These added features contribute information to reveal the context of the characters, while giving required emphasis to the first and last character of the word. Feature sets with boundary characters appended to the word undergo refinement by removal of features.

The experiments (described in section 3.5, below) evaluate five different variants of test features (including PS_NBC above).

From the set of all possible powerset combinations of characters including the boundary characters, all spurious boundary character features such as individual occurrences of either '@' or '#' or those without word characters such as "@#" are removed. This comprehensive feature set is referred to as **PS_XBC**. The next feature set I consider is one that resembles the type of features used by De Pauw & Wagacha (2007) who

append features with strings to mark beginning, ending, and inside character substrings. These distinguish where the word beginning and ending occurs similarly to the case of the "WORD" example where features are tagged with "B=" and "E=" tags to indicate beginning and ending of word; the absence of boundary characters would indicate the inside subword feature similar to the "I=" features. Thus, strings where the first and last character of the word appear without a boundary character are dropped. This feature set is referred to as **PS_BBC**.

Another smaller feature set, **PS_1BC,** is considered, where I remove features where starting and ending boundary characters both occur. This is in order to gauge the advantage of using a smaller feature set, while observing any significant change in performance.

Finally, the last type of feature set, **NC1_BBC** excludes those feature strings with two consecutive characters formed by non-contiguous characters from the word spaced apart by two or more characters. In other words, two consecutive characters as potential infixes are not allowed. This restriction is based on the fact that for undiacritized text we would expect to see only single infix characters.

Table 3.2 illustrates the operation of these feature variants for the word *slAmA* (meaning *peace*) after appending the boundary characters. The corresponding pattern-based features derived from the root features in a similar manner to that described for PS_NBC, appear in Table 3.3. Here, if a boundary character occurs in the root feature, the corresponding pattern feature also retains the same boundary character.

PS_XBC	PS_BBC	PS_1BC	NC1_BBC
@s, @sl, @slA, @slAm, @slAmA, @slAmA#, @slAm#, @slAA, @slAA#, @slA#, @slm, @slmA, @slmA#, @slm#, @slA, @slA#, @sl#, @sA, @sAm, @sAmA, @sAmA#, @sAm#, @sAA, @sAA#, @sA#, @sm, @smA, @smA#, @sm#, @sA, @sA#, @s#, @l, @lA, @lAm, @lAmA, @lAmA#, @lAm#, @lAA, @lAA#, @lA#, @lm, @lmA, @lmA#, @lm#, @lA, @lA#, @l#, @A, @Am, @AmA, @AmA#, @Am#, @AA, @AA#, @A#, @m, @mA, @mA#, @m#, @A, @A#, s, sl, slA, slAm, slAmA, slAmA#, slAm#, slAA, slAA#, slA#, slm, slmA, slmA#, slm#, slA, slA#, sl#, sA, sAm, sAmA, sAmA#, sAm#, sAA, sAA#, sA#, sm, smA, smA#, sm#, sA, sA#, s#, l, lA, lAm, lAmA, lAmA#, lAm#, lAA, lAA#, lA#, lm, lmA, lmA#, lm#, lA, lA#, l#, A, Am, AmA, AmA#, Am#, AA, AA#, A#, m, mA, mA#, m#, A, A#	@s, @sl, @slA, @slAm, @slAmA#, @slAm#, @slAA#, @slA#, @slm, @slmA#, @slm#, @slA#, @sl#, @sA, @sAm, @sAmA#, @sAm#, @sAA#, @sA#, @sm, @smA#, @sm#, @sA#, @s#, @l, @lA, @lAm, @lAmA#, @lAm#, @lAA#, @lA#, @lm, @lmA#, @lm#, @lA#, @l#, @A, @Am, @AmA#, @Am#, @AA#, @A#, @m, @mA#, @m#, @A#, l, lA, lAm, lAmA#, lAm#, lAA#, lA#, lm, lmA#, lm#, lA#, l#, A, Am, AmA#, Am#, AA#, A#, m, mA#, m#, A#	@s, @sl, @slA, @slAm, @slm, @sA, @sAm, @sm, @l, @lA, @lAm, @lm, @A, @Am, @m, l, lA, lAm, lAmA#, lAm#, lAA#, lA#, lm, lmA#, lm#, lA#, l#, A, Am, AmA#, Am#, AA#, A#, m, mA#, m#, A#	@s, @sl, @slA, @slAm, @slAmA#, @slAm#, @slAA#, @slA#, @slm, @slmA#, @slm#, @sl#, @sA, @sAm, @sAmA#, @sAm#, @sAA#, @sA#, @s#, @l, @lA, @lAm, @lAmA#, @lAm#, @lAA#, @lA#, @lm, @lmA#, @lm#, @l#, @A, @Am, @AmA#, @Am#, @AA#, @A#, @m, @mA#, @m#, @A#, l, lA, lAm, lAmA#, lAm#, lAA#, lA#, lm, lmA#, lm#, l#, A, Am, AmA#, Am#, AA#, A#, m, mA#, m#, A#

Table 3.2: Root based feature sets for *@slAmA#*

PS_XBC	PS_BBC	PS_1BC	NC1_BBC
@-lAmA, @--AmA, @---mA, @----A, @-----, @-----#, @----A#, @---m-, @---m-#, @---mA#, @--A-A, @--A--, @--A--#, @--A-A#, @--Am-, @--Am-#, @--AmA#, @-l-mA, @-l--A, @-l---, @-l---#, @-l--A#, @-l-m-, @-l-m-#, @-l-mA#, @-lA-A, @-lA--, @-lA--#, @-lA-A#, @-lAm-, @-lAm-#, @-lAmA#, @s-AmA, @s--mA, @s---A, @s----, @s----#, @s---A#, @s--m-, @s--m-#, @s--mA#, @s-A-A, @s-A--, @s-A--#, @s-A-A#, @s-Am-, @s-Am-#, @s-AmA#, @sl-mA, @sl--A, @sl---, @sl---#, @sl--A#, @sl-m-, @sl-m-#, @sl-mA#, @slA-A, @slA--, @slA--#, @slA-A#, @slAm-, @slAm-#, -lAmA, --AmA, ---mA, ----A, -----, -----#, ----A#, ---m-, ---m-#, ---mA#, --A-A, --A--, --A--#, --A-A#, --Am-, --Am-#, --AmA#, -l-mA, -l--A, -l---, -l---#, -l--A#, -l-m-, -l-m-#, -l-mA#, -lA-A, -lA--, -lA--#, -lA-A#, -lAm-, -lAm-#, -lAmA#, s-AmA, s--mA, s---A, s----, s----#, s---A#, s--m-, s--m-#, s--mA#, s-A-A, s-A--, s-A--#, s-A-A#, s-Am-, s-Am-#, s-AmA#, sl-mA, sl--A, sl---, sl---#, sl--A#, sl-m-, sl-m-#, sl-mA#, slA-A, slA--, slA--#, slA-A#, slAm-, slAm-#	@-lAmA, @--AmA, @---mA, @----A, @----#, @----A#, @---m-#, @---mA#, @--A-A, @--A--#, @--A-A#, @--Am-#, @--AmA#, @-l-mA, @-l--A, @-l--#, @-l--A#, @-l-m-#, @-l-mA#, @-lA-A, @-lA--#, @-lA-A#, @-lAm-#, @-lAmA#, @s-AmA, @s--mA, @s---A, @s----#, @s---A#, @s--m-#, @s--mA#, @s-A-A, @s-A--#, @s-A-A#, @s-Am-#, @s-AmA#, @sl-mA, @sl--A, @sl---#, @sl--A#, @sl-m-#, @sl-mA#, @slA-A, @slA--#, @slA-A#, @slAm-#, s-AmA, s--mA, s---A, s----#, s---A#, s--m-#, s--mA#, s-A-A, s-A--#, s-A-A#, s-Am-#, s-AmA#, sl-mA, sl--A, sl---#, sl--A#, sl-m-#, sl-mA#, slA-A, slA--#, slA-A#, slAm-#	@-lAmA, @--AmA, @---mA, @----A, @--A-A, @-l-mA, @-l--A, @-lA-A, @s-AmA, @s--mA, @s---A, @s-A-A, @sl-mA, @sl--A, @slA-A, s-AmA, s--mA, s---A, s---#, s---A#, s--m-#, s--mA#, s-A-A, s-A--#, s-A-A#, s-Am-#, s-AmA#, sl-mA, sl--A, sl---#, sl--A#, sl-m-#, sl-mA#, slA-A, slA--#, slA-A#, slAm-#	@-lAmA, @--AmA, @---mA, @----A, @----#, @----A#, @---m-#, @---mA#, @--A-A, @--A--#, @--A-A#, @--AmA#, @-l-mA, @-l--A, @-l---#, @-l--A#, @-l-m-#, @-l-mA#, @-lAmA#, @s-AmA, @s--mA, @s---A, @s----#, @s---A#, @s--m-#, @s--mA#, @s-A-A, @s-A--#, @s-A-A#, @s-AmA#, @sl-mA, @sl--A, @sl---#, @sl--A#, @sl-m-#, @sl-mA#, @slA-A, @slA--#, @slA-A#, @slAm-#, s-AmA, s--mA, s---A, s----#, s---A#, s--m-#, s--mA#, s-A-A, s-A--#, s-A-A#, s-AmA#, sl-mA, sl--A, sl---#, sl--A#, sl-m-#, sl-mA#, slA-A, slA--#, slA-A#, slAm-#

Table 3.3: Corresponding pattern based feature sets derived from the root based feature set (Table 3.2) replacing root characters with ‘-‘ while copying missing characters from the word. Boundary characters are copied from the root-based features without change.

3.2.3 Model Training

3.2.3.1 Parameter Estimation

Section 3.2.1 argued that one advantage of ME modelling is the ability to incorporate a wide diversity of features which are overlapping and therefore not independent of each other. But there is a cost to this kind of representation. The model parameters that need to be estimated require large amounts of training data since there are large number of free parameters. Also, the estimation process could be subject to rounding-off errors due to sparsity of the features. Due to these reasons, a highly efficient and accurate method of parameter estimation is required.

The general algorithm for parameter estimation is as follows:

Input: Feature functions, $f_1, f_2, \dots f_n$; empirical distribution $\tilde{p}(x, y)$
Output: Optimal parameter values; optimal model $\tilde{p}$.

1. Initialize $\lambda_i = 0, i \in \{1,2,3, \dots, n\}$
2. Do for each, $i \in \{1,2,3, \dots, n\}$
    a. Apply method to compute $\Delta\lambda_i$
    b. Perform update : $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$
3. Repeat step 2 until λ_i converges

The most significant step (2a) in the algorithm is the method that is used to compute the updates $\Delta\lambda_i$. There are two types of estimation methods used for computing the updates for maximum entropy modelling in the context of natural language processing: iterative scaling and gradient-based learning. In this section we outline each of these two types of estimation techniques and their merits.

Iterative Scaling

Iterative scaling (Huang *et al*, 2010) is based on iteratively updating the parameters ensuring that the objective function is improved at each iteration. Thus, weights are updated such that the change in log-likelihood, $L(p_{\lambda+\Delta\lambda}) - L(p_\lambda)$ is always positive leading to the maximal value for $L(p_\lambda)$. Leaving aside the details of the derivation, after

solving for the change in log-likelihood, the updates, $\Delta\lambda_i$, are optimized by finding the solution to the equation

$$\tilde{p}(f_i) = \sum_{x,y} \tilde{p}(x)p(y|x)\, f(x,y)\exp\Big(\Delta\lambda_i f^{\#}(x,y)\Big) \tag{3.13}$$

where

$$f^{\#}(x,y) = \sum_{i=1}^{n} f_i(x,y) \tag{3.14}$$

There are two methods for iterative scaling: Generalized Iterative Scaling (GIS) (Darroch & Ratcliff, 1972) and Improved Iterative Scaling (IIS) (Berger, 1997; Pietra *et al*, 1995). GIS requires that the value of $f^{\#}(x,y) = C$, a constant, i.e. that the features sum to a constant. In this case the updates can be determined analytically by taking the factor proportional ratio

$$\Delta\lambda_i = \log\left(\frac{\tilde{p}(f_i)}{p_\lambda(f_i)}\right)^{\frac{1}{C}} \tag{3.15}$$

If the rows of the training data do not sum to a constant, then the value of the constant C is determined empirically by introducing a "correction" feature f_{n+1}

$$C = \max_{x,y} f^{\#}(x,y) \tag{3.16}$$

$$f_{n+1} = C - \max_{x,y} f^{\#}(x,y) \tag{3.17}$$

The rate of convergence depends on the step-size, which in turn is determined by the value of C: the higher value of C the smaller will be the step size.

The disadvantage of GIS is that the step size may be very small due to the factor $1/C$ leading to a slow convergence. IIS tries to avoid the use of a correction feature and hence the slow convergence by obtaining the solution to the equation (3.13), where $f^{\#}$ is the sum of feature values for event y, and $\exp(\Delta\lambda_i)$ is determined numerically, as opposed to analytical solution for GIS, by using Newton's method.

Gradient-Based Method

Gradient Based methods (Malouf, 2002) aim to optimize the weight updates according to the gradient function

$$G(\lambda) = \tilde{p}(f_i) - p_\lambda(f_i) \tag{3.18}$$

Again, a solution cannot simply be obtained analytically by equating $G(\lambda) = 0$ and solving for λ. Numerical methods must be applied, adjusting the value of λ at each step.

The primary strength of iterative scaling methods lies in the ability to compute the expected value $\tilde{p}(f_i)$ without explicitly depending on the expensive calculation of the gradient of the log-likelihood function. In actual fact, the expected values vector required by the iterative scaling methods is essentially the gradient itself.

Since the objective is to maximize the log-likelihood, the parameter needs to be updated at each step k, in the direction in which the objective function's value increases rapidly, maximizing the log-likelihood, $L(\lambda^k + \Delta\lambda^k)$. The update for the weights is thus,

$$\Delta\lambda^k = a^k \boldsymbol{d}^k \tag{3.19}$$

where a^k is the step size usually set such that $L(\lambda^k + \Delta\lambda^k) > L(\lambda^k)$, and $\boldsymbol{d}^k$ is the direction step calculated such that $\boldsymbol{d}^k G(\lambda) > 0$. As the log-likelihood function is concave, the method of steepest ascent ($\boldsymbol{d}^k = G(\lambda)$) is guaranteed to find the optimal solution. The *Newton* method further takes into consideration the curvature of the gradient, determining the direction $\boldsymbol{d}^k = H^{-1}G(\lambda)$ where H^{-1} is the inverse Hessian matrix. The Newton Method converges quickly but involves the expensive calculation of the Hessian matrix. An approximation of the H^{-1} matrix is a matrix B^k, obtained

with current and previous updates and gradients by the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) method. The matrix B^k is expensive to store, hence a Limited Memory Variable Method (LMVM) is used which approximates the matrix using only the current parameters, i.e. the previous *m* values of the updates and the gradient.

A special advantage of the LMVM-BFGS algorithm is that it is proven to converge very fast (Malouf, 2002; Andrew & Gao, 2007). It requires only the gradient to be computed at each step, unlike iterative scaling which needs several derivational steps to obtain update rules.

3.2.3.2 Smoothing

As we are dealing with many features we are bound to encounter issues of data sparsity. Several strategies have been proposed to deal with this issue. One possibility is to perform a limited number of iterations of model weight training in order to avoid over-fitting. Another approach is the cut-off method, i.e. to exclude from training features which have low frequency counts which are deemed to be uninformative and insignificant.

A technique that has recently been found to be more successful with dealing with data sparsity is to replace Maximum Likelihood (ML) estimation with maximum *a posteriori* estimation (MAP) with Gaussian Priors (Chen & Rosenfeld, 2000). The weights, λ, are determined to maximize the posterior probability of the model trained on data, D. According to Bayes' Rule

$$P_{pos}(\lambda|D) = P(D|\lambda) \times P_{prior}(\lambda) \tag{3.20}$$

Taking the log,

$$\hat{L}(p_\lambda) = L(p_\lambda) - \sum_i \left(\frac{1}{2\sigma_i^2}\right) \lambda_i^2 \tag{3.21}$$

Setting the prior has the effect that it gives a penalty to the model if the model weights are too high or too small. Penalizing this objective function has the effect of avoiding

over-fitting. Modifications to the iterative scaling weight calculation algorithm are apparent in the update equation,

$$\tilde{p}(f_i) = \sum_{x,y} \tilde{p}(x)p(y|x)\, f(x,y)\exp\Big(\Delta\lambda_i f^{\#}(x,y)\Big) - \frac{\lambda_i + \Delta\lambda_i}{\sigma_i^2} \qquad (3.22)$$

This equation again has no analytical solution but can be solved with little overhead using, for example, the Newton-Raphson method. For the gradient function we have the following modification,

$$G(\lambda) = \tilde{p}(f_i) - p_\lambda(f_i) - \frac{\lambda_i}{\sigma_i^2} \qquad (3.23)$$

This modification also does not have a substantial overhead.

Gaussian MAP estimation has been successfully applied to various NLP tasks and shown to reduce overfitting. Examples include part-of-speech tagging (Curran & Clark, 2003) and language modelling (Berger, 1998).

3.2.3.3 Model Training for Morphology Induction

I used a ME modelling toolkit, implemented by Zhang (2004), which implements GIS for iterative scaling and the LMVM-BFGS gradient-based method. In the experiments I tested with both types of estimation method to see which would work best for the unsupervised training task.

As for the smoothing of ME models, in supervised learning tasks the Gaussian prior is usually determined using held-out data. I experimented with various values for the Gaussian prior ranging from 0 to 2 or order to gauge any difference in performance due to using the different priors.

The number of iterations also plays a vital role in determining model performance. Undertraining and over-training would both result in suboptimal performance. After each step of the training iteration, the log likelihood of the model is increasing (i.e. the

probability decreases). I choose a cut-off point for the number of iterations to be where the training accuracy reaches 100%.

3.2.4 Model Application

Having obtained the various models, I apply the models back to the same training data features. In supervised machine learning tasks, the trained model would be applied to unseen data to obtain the best classification output. For the unsupervised learning, for every target word's feature set, rather than retrieve only the best class, which would be the target word itself, all classes are retrieved with proximity values to the target word's features. This proximity is the probability for each class estimated by the maximum entropy model given the morpho-orthographic constraints. The novel application of a machine learning technique in this manner, where reclassification of the training data set takes place, does not bring any kind of unfair advantage in the unsupervised learning process (DePauw & Wagacha, 2007).

Given V vocabulary words, in the application phase, for each word of the corpus vocabulary, its features are applied to the model to get V word classes with proximity values. Sorted in descending order, this results in a ranking of word proximities with the most similar words at the top. The top entry in the ranking with score ≈1 would be the target word itself whose features are input into the model. Thereafter words with the most probable features to the target word's features are ranked in order. Probability values along the ranked list drop drastically, so I decided to cut off the list at k=500 words, as a sufficient number to gauge proximity for root based and pattern based word similarity. In summary, the output file of the model application phase consists of V clusters of k nearest neighbour words. An example of one such pair of root and pattern clusters for the word, *slAm* is shown in Table 3.4.

	slAm	**0.999664**
1	slAmA	0.000283
2	ElAm	2.24E-05
3	ZlAm	7.80E-06
4	slm	7.39E-06
5	'slAm	4.87E-06
6	slmA	4.43E-06
7	glAm	2.68E-06
8	klAm	1.07E-06
9	slAlp	5.73E-07
10	slTAn	5.46E-07
11	sAlt	1.54E-07
12	Alm	1.31E-07
13	lA	1.15E-07
14	sAlmwn	9.73E-08
15	mslmA	9.10E-08
16	rslA	8.70E-08
17	lm	5.82E-08
18	slfA	4.71E-08
19	'zlAm	3.20E-08
20	'HlAm	2.52E-08

	slAm	**0.998416**
1	slym	0.000853
2	ElAm	1.89E-05
3	klAm	1.84E-05
4	glAm	1.84E-05
5	ZlAm	1.84E-05
6	sqym	6.32E-06
7	smwm	6.28E-06
8	sAhm	6.28E-06
9	slmA	1.28E-06
10	slfA	1.27E-06
11	Hlym	1.27E-06
12	tlwm	1.27E-06
13	Elym	1.26E-06
14	mlym	1.26E-06
15	smAn	1.26E-06
16	'lym	1.26E-06
17	tlkm	1.25E-06
18	'qAm	1.24E-06
19	sHAr	1.24E-06
20	s'Al	1.24E-06

Table 3.4: Top entries for the nearest neighbours to the target word *slAm* (peace) in terms of root (left side) and pattern (right side)

The advantage of this approach to obtaining the morphological relatedness of words over other approaches such as minimum edit distance is the ability to identify and better capture morpheme dependencies between words which may be orthographically quite different. This is especially so for morphologically complex languages where the base form is quite small, as in the case of Arabic, with the root consisting of mostly three letters or sometimes 4 (and very rarely 5). Indeed, the number of affix characters may typically equal or even exceed the number of base characters.

An ME based technique is well suited to such morphologically complex cases since it is able to find morpheme relatedness of morphological features, (automatically) generated from a word. Considering the examples in Table 3.2 and Table 3.3, common features such as "@s", "l", "m#", would lack selective power, providing weak constraints to group words. Other features such as *@slm#,* occurring less frequently, should carry more weight and form useful constraints to group words.

3.3 Lexicon Extraction

The approach described above uses machine learning to obtain the morphological relatedness of words. However, it does not separate words into morphemes. The next step applies a procedure which utilizes the clusters' word proximities to give a weight to each prospective morpheme. Each morpheme is given a weight. The morphemes are sorted so those with the highest weights are at the top. This results in two sorted lists of roots and patterns each with the most plausible morphemes at the top. These two lists form lexicons that support the process of morphological analysis.

The next subsection outlines the procedure to weight each morpheme, given cluster proximities. The technique is again inspired by the work of De Pauw & Wagacha (2007), but correcting a significant flaw in their method.

3.3.1 Morpheme Weighting

The clusters are each used to weight the two types of morphemes: affixes are weighted using the root-related word proximity clusters; likewise roots are weighed utilizing the pattern-related word proximities.

The proximity score for each morpheme type is accumulated at two levels: the local cluster level and globally over the entire set of clusters. Looking at the clusters, the top element is the word whose features exactly match its own features giving the probability score ≈1. This top element is referred to as the headword, with each subsequent word having proximity to the headword based on either root oriented feature constraints or pattern oriented constraints. The set of all headwords of each cluster constitutes the vocabulary of the dataset.

The headword of each cluster is decomposed into all possible combinations of triliteral roots and corresponding patterns. For example the word *yErf* (meaning *he knows*) is decomposed into the four possible pairs of root and pattern morphemes:

$$yErf \rightarrow \begin{Bmatrix} \langle y\ E\ r, & - - - f\ \rangle, \\ \langle y\ E\ f, & - - r - \rangle, \\ \langle y\ r\ f, & - E - - \rangle, \\ \langle E\ r\ f, & y - - - \rangle \end{Bmatrix} \tag{3.24}$$

These are the candidate morphemes which are each assigned weights, locally at each cluster level and globally over all the clusters. Firstly, weights are assigned to the patterns using the root-related clusters. For each pattern in the headword we match all words in our cluster which contain the corresponding root and accumulate the score of each of the matched words. Conversely, for each root in the headword we match all words in the cluster which contain the corresponding root and accumulate the score of each of the matched words. The scoring method is described below in 3.3.2. For example, using the 20 top entry clusters shown in Table 3.4, the local score for root and pattern candidates of word *slAm,* is shown in Table 3.5

Root	Pattern	Word with Pattern	Cumulative PatternScore	Word with Root	Cumulative Pattern Score
slA	- - - m	slym, ElAm, klAm, glAm, ZlAm, sqym, smwm, sAhm, Hlym, tlwm, Elym, mlym, 'lym, tlkm, 'qAm	0.000853 + 1.89E-05 + 1.84E-05 + 1.84E-05 + 1.84E-05 + 6.32E-06 + 6.28E-06 + 6.28E-06 + 1.27E-06 + 1.27E-06 + 1.26E-06 + 1.26E-06 + 1.26E-06 + 1.25E-06 + 1.24E-06 = 0.000955	slAmA, 'slAm, slmA, mslmA, slfA	0.000282663 + 4.87324E-06 + 4.42598E-06 + 9.09942E-08 + 4.71455E-08 = 0.0002921
slm	- - A -	ElAm, klAm, glAm, ZlAm, smAn, 'qAm, sHAr, s'Al	1.89E-05 + 1.84E-05 + 1.84E-05 + 1.84E-05 + 1.26E-06 + 1.24E-06 + 1.24E-06 + 1.24E-06 = 7.91E-05	slAmA, slm, 'slAm, slmA, sAlmwn, mslmA	0.000282663 + 7.38568E-06 + 4.87324E-06 + 4.42598E-06 + 9.73338E-08 + 9.09942E-08 = 0.000299536
sAm	- l - -	slym,	0.000853 +	slAmA,	0.000282663 +

		ElAm, klAm, glAm, ZlAm, slmA, slfA, Hlym, tlwm, Elym, mlym, 'lym, tlkm	1.89E-05 + 1.84E-05 + 1.84E-05 + 1.84E-05 + 1.28E-06 + 1.27E-06 + 1.27E-06 + 1.27E-06 + 1.26E-06 + 1.26E-06 + 1.26E-06 + 1.25E-06 = 0.000938	'slAm, sAlmwn	4.87324E-06 + 9.73338E-08 = 0.000287633
lAm	s - - -	slym, sqym, smwm, sAhm, slmA, slfA, smAn	0.000853 + 6.32E-06 + 6.28E-06 + 6.28E-06 + 1.28E-06 + 1.27E-06 + 1.26E-06 = 0.000876	slAmA, ElAm, ZlAm, 'slAm, glAm, klAm, 'zlAm, 'HlAm,	0.000282663 + 2.24062E-05 + 7.80066E-06 + 4.87324E-06 + 2.68159E-06 + 1.07234E-06 + 3.20347E-08 + 2.51734E-08 = 0.000321554

Table 3.5: Example of Pattern and Root candidates scoring for word *'slAm'*

Each morpheme's score further is summed globally over all the local cluster scores where the morpheme occurs in the headword of the clusters. Even though locally some implausible morpheme might have a higher score, as we accumulate the score globally, the more sound morphemes such as *slm* and - - *A*- would tend to gain weight and progress up the list of plausible roots and patterns. Some examples of top scoring morphemes are shown in Table 3.6.

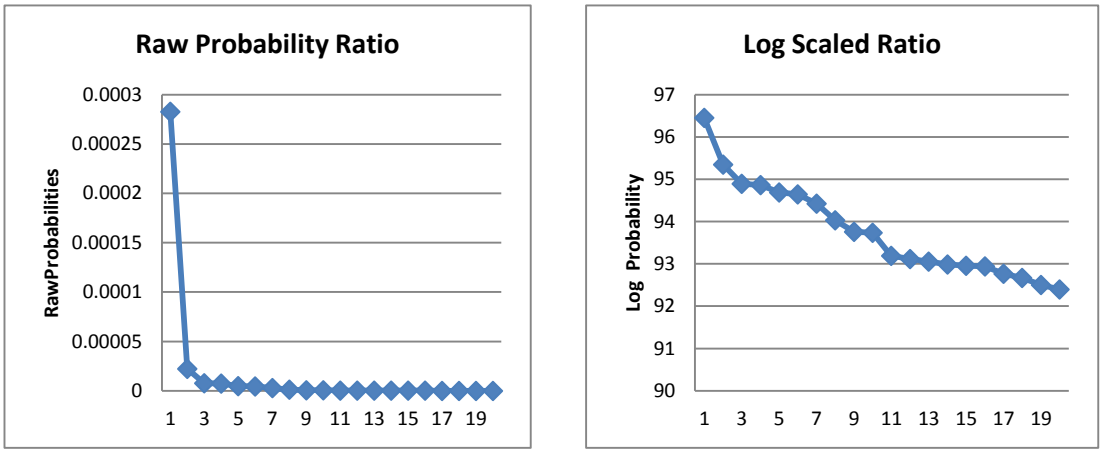
Pattern	Roots (Sense)
- - -A	Sdq (truth)
- - - -A	'mn (faith)
- -A -	Hsn (beauty)
-A - -	xrj (depart)
- - -t	qym (establish)
' - - -	Elm (know)
m - - -	nzl (descend)
- - - -Y	kfr (disbelieve)
- -y -	jmE (gather)
t - - -	smE (listen)
t - - - -	xlf (differ)
y - - -	slm (submit)
…	…

Table 3.6: Top scoring patterns and
roots after global scoring

3.3.2 Scoring Measure

Empirically, the raw probability values obtained for each word's proximities to the headword decrease exponentially going down the cluster list. The ratios are hard to compare and aggregate over since similar words with lower values tend to give insignificant contributions to the score. This is shown in Figure 3.1a. Thus a method is required to scale these raw probability values such that the relative difference is reduced, bringing them closer to a linear scale.

Logarithmic scaling is a technique that is often applied to data of this nature, transforming exponential functions into linear ones. In this case the function to compute an exact linear form would be too intricate to obtain since it would be a combination of feature weights and other parameters. In order to visualise this effect, Figure 3.1 plots the raw probabilities to the log scaled values for the first 20 cluster values for the word *slAm*.



(a) Raw probability values for top 20 words in the cluster.

(b) Log scaled probability values for the same words

Figure 3.1: Comparison of raw probabilities with log scaled ratios for the first 20 entries

As we would be taking sum of the log probabilities, negative values for the resulting logarithmic score are undesirable. In order to obtain a positive range of values, all probabilities are divided by a reference probability before taking the log. The reference probability is the lowest probability globally in the entire set of clusters, $Prob_{min}$:

$$\log\left(\frac{Prob}{Prob_{min}}\right) \equiv \log(Prob) - \log(Prob_{min}) \tag{3.25}$$

This approach differs from that of DePauw & Wagacha (2007) who incorrectly take the negative log of the probability in order to make the log score positive before summing

the probabilities[5]. Contrary to intuition, when going down the ranked cluster list the proximity of the words to the headword increases instead of decreasing. Words having lower probability would contribute a higher score which contradicts the principle of morphological relatedness as implied here. Even though intuitively incorrect, De Pauw & Wagacha (2007) report good results. The reason is that this scoring measure tends to give more consideration to longer matched words that appear lower in the cluster list. The way to amend this aspect is to give appropriate consideration to longer morphemes. I therefore introduce an adjustment to accommodate for varying lengths of each morpheme, as described below.

3.3.3 Length Adjustment

As discussed in the previous section, the character length of morphemes affects their ranking in the lexicon. Since we are only considering triliteral root morphemes, the length adjustment procedure need not be applied to root morphemes. But pattern morphemes vary in length depending on the number of infix characters in the pattern template. Some advantage needs to be given to longer patterns since words containing longer patterns tend to get lower probability scores in the morpheme based clusters.

The length adjustment procedure is based on a technique of Chung & Gildea (2009) and Liang & Klein (2009), who use an exponential *length penalty* measure to adjust their Chinese word segmentation model according to the number of segments. They penalise longer segmentations of a sentence using the penalty $e^{-|z_k|^{\beta}}$, where $|z_k|$ is the number of word segments, and β is the strength of the penalty.

I adapt this measure, such that, to give advantage to higher length morphemes, I multiply each pattern morpheme score by $e^{|p|}$, where $|p|$ is the number of characters in the pattern morphemes. I overlook the β penalty strength parameter in order to keep the procedure as parameter free as possible, assuming unit strength ($\beta = 1$). This measure is intuitively appealing since it can be expected that morpheme frequencies are exponentially related to the character length of the morpheme.

---

[5] This flaw has been communicated to the authors of the paper, who have acknowledged it.

3.4 Morphological Analysis

Given the root and pattern lexicons, I use a simple procedure for morphological analysis. A word is analysed into its root and pattern template by considering every possible combination of triliteral roots and corresponding patterns, $\langle r^x, p^x \rangle$, as exemplified in (3.24). A combined score for each root and pattern combination is computed, i.e. each analysis is scored with the sum of the scores for the root, r^x, and pattern, p^x, in the root lexicon and pattern lexicon, respectively. Due to the different ranges of scores for root and pattern, the score for the root morphemes is scaled with respect to the pattern morphemes, as shown in the equation below, in order to guarantee equal contributions:

$$SS(r) = S(r) \times \frac{\max(S(p))}{\max(S(r))} \tag{3.26}$$

The analysis, x, with the highest score is selected as the output:

$$\max_{x=1..n} \left(S(r_w^x) + SS(p_w^x) \right) \tag{3.27}$$

3.5 Evaluation

In common with the evaluations elsewhere in this thesis, this evaluation assumes a realistic setting of unvowelled text, since most Arabic text is written without vowels. The data is an undiacritized version of the Quranic Arabic Corpus (see Appendix C); I chose this corpus since it identifies the root of each word, facilitating robust evaluation. The fact that the corpus contains a relatively small vocabulary of around 7000 words also simulates the scenario for most of the world's languages of scarcity of linguistic resources and data.

The next section (3.5.1) describes the evaluation data. This is followed by an explanation of the baseline and the measure used for evaluation (3.5.2).

3.5.1 The Dataset

The Quranic Arabic Corpus (QAC) consists of approximately 77,900 word tokens, with a total of around 19,000 unique tokens. Since I am interested in investigating learning from undiacritized text, I removed all short vowels and diacritical markers. The size of the resulting vocabulary, after removal of vowels, is approximately 14,850. Further details of the corpus and steps taken to prepare the input, such as diacritic removal, are given in Appendix A.

I took as input lightly stemmed words, i.e. words with clitics removed, but with most inflectional markers attached. The justification for this is that stemmed words are obtainable using existing tools for unsupervised concatenative morphology learning. For example, the technique of Poon et al (2009) could be used to accurately extract the stem for each word. The stemmed unvowelled vocabulary size is around 7370.

The original corpus is annotated with roots for all derived and inflected words. More than 95% of words are tagged with their root forms since the Quran consists mostly of inflected forms, with very few proper nouns. There are 7192 stemmed words with available roots.

In Arabic, some morphological alterations take place; for example, when moving from a root containing a long vowel to the surface word, the long vowel might change its form. Such words, whose characters do not match every radical of the root, were removed from the evaluation as they are beyond the scope of the learning algorithm to identify. Removing these word and root pairs leaves 5532 stemmed types.

Triliteral roots account for the vast majority of root types in Arabic. In the QAC, 64 out of the total of 5532 stemmed types have quadraliteral roots and none have any other root types. These 4-letter root words form 1.15 % of the total stemmed types. Removing these leaves 5468 triliteral root words.

3.5.2 The Baseline and Evaluation Measure

As a baseline for evaluation, I derived root and pattern lexicons in a similar manner to the procedure for lexicon extraction described in section 3.3. Patterns are scored by counting the number of co-occurring roots in the vocabulary. Likewise, the root score is obtained by counting the number of words in the vocabulary with co-occurring patterns. In contrast to lexicon extraction from ME based clusters, the baseline can be viewed as accumulation of unit weight of 1 from a single 'cluster', the vocabulary set.
The evaluation measures the accuracy of morphological analysis of the 5468 words in the evaluation dataset, described in the previous section. The morphological analysis procedure (section 3.4) is applied to each word and the percentage of correct analyses is recorded.

3.5.3 System Configuration

The experiments compare a number of system configurations, as outlined below.
The five variant sets of morphological features described in section 3.2.2 are:

- PS_NBC: powerset feature combinations without boundary characters
- PS_XBC: powerset features with boundary characters appended, after removal of uninteresting features common in all words such as '@', '#', etc.
- PS_BBC: PS_XBC after removing features where the beginning and end word characters occur without the boundary character.
- PS_1BC: a slightly more refined feature set than PS_BBC after considering features with either of the two boundary characters.
- NC1_BBC: further refinement by removal of features from PS_BBC with two or more non-contiguous characters.

As stated earlier, the various feature-sets have been selected for comparison purposes. PS_BBC seems likely to give the best results given its meaningful boundary character usage and its close resemblance to the feature set used by DePauw & Wagacha (2007).

Two weight optimization schemes (section 2.3.2) are investigated:

- ME_IIS: Maximum Entropy model based on IIS
- ME_LBFGS: Maximum Entropy model based Limited memory BFGS

LBFGS is the preferred optimization scheme as this has been shown to perform better, in terms of faster convergence and prediction accuracy, for Maximum Entropy modelling (Malouf, 2003).

For model smoothing, three approaches are compared (section 3.2.3.2)

- ME_GS_Val: where Val is the value of the Gaussian from the set {0.0, 0.5, 1.0, 1.5, 2.0}
- ME_PB_Itr*N*: where N is the number of iterations for pattern based feature model training
- ME_RB_Itr*N*: where N is the number of iterations for root based feature model training

Three types of length adjustment methods are compared, the third following the procedure of section 3.3.3.

- ME_RW: raw probability values used as word scores
- ME_LS _LA: log-scaled probability values as word scores
- ME_LS_LA: log-scaled scores adjusted for morpheme length

3.5.4 System Evaluation and Discussion

This section presents a comparative evaluation between the preferred and other system configurations defined in section 3.5.3, based on the evaluation criteria described in section 3.5.2.

3.5.4.1 Feature Set Evaluation

The first evaluation is over the different feature sets to compare the performance of the chosen set with the others. We compare the different powerset combinations with and without the boundary character as defined in section 3.2.2. Table 3.7 outlines the results of the comparison of the different feature sets.

Configuration	Total Correct	Percentage Correct
PS_NBC	4551	83.23
PS_XBC	4447	81.33
PS_BBC	4715	86.23
PS_1BC	4695	85.86
NC1_BBC	4680	85.60

Table 3.7: Comparison of different feature sets

The feature set computed without the boundary character (PS_NBC) gives lower performance than the feature sets computed with the boundary characters attached (PS_BBC, PS_1BC, NC1_BBC). Simply appending the boundary characters before computing the complete powerset (PS_XBC) does not give any advantage in distinguishing word beginning and ending, but instead adds to the ambiguity due to the introduction of the two additional characters. The set needs to be cut down to allow only those features where the first and last characters of the word are attached with the boundary characters (@,#). PS_BBC disallows features where first and last characters occur without the boundary character. This model is able to better predict the relatedness of words.

The results also show that dropping features containing simultaneous occurrences of both boundary characters (PS_1BC) gives a slight loss in performance. Nor is there any gain by only considering features with single non-contiguous characters (NC1_BBC). However, these sets may have advantages in cases where lower computation cost is a requirement for model training and application, since there are fewer features in these sets.

3.5.4.2 Optimization Scheme Evaluation

Using the chosen PS_BBC feature set, schemes for parameter estimation are compared. The two types of weight optimization schemes are the chosen LBFGS method and IIS.

Configuration	Total Correct	Percentage Correct
ME_LBFGS	4715	86.23
PS_IIS	4197	76.76

Table 3.8: Comparison of two parameter estimation techniques

The results in Table 3.8 show a marked performance benefit for LBFGS over iterative scaling. This agrees with previous results (Malouf, 2002), where the LBFGS method for parameter estimation is shown to be a more effective and efficient method especially suited to Maximum Entropy modeling.

3.5.4.3 Gaussian Smoothing Evaluation

For unsupervised learning, keeping the technique parameter free, I assume no Gaussian prior (=0.0) but for experimental purposes, I compare different values of the Gaussian prior in the range 0.0-2.0.

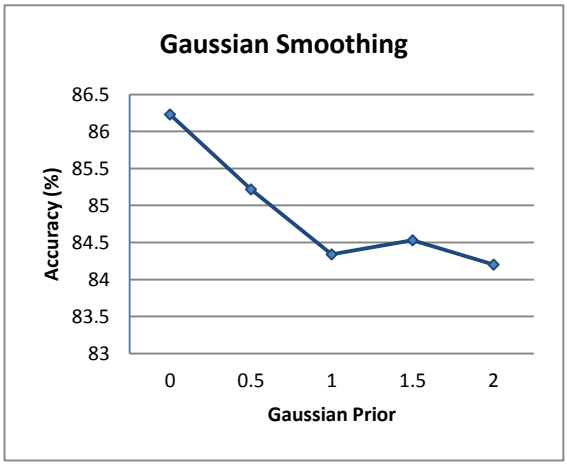


Figure 3.2: Illustration of Table 3.9

Configuration	Total Correct	Percentage Correct
ME_GS_0.0	4715	86.23
ME_GS_0.5	4660	85.22
ME_GS_1.0	4612	84.34
ME_GS_1.5	4622	84.53
ME_GS_2.0	4604	84.20

Table 3.9: Comparison of different Gaussian Priors

As seen in Figure 3.2 and Table 3.9 contrary to expectation, the performance drops when adding a prior for Gaussian smoothing. Thus, Maximum *A Posteriori* (MAP) learning does not give any advantage over Maximum Likelihood Estimation (MLE). Perhaps over-fitting is not a serious concern here as the training and test sets are the same.

3.5.4.4 Number of Iterations Evaluation

Model training using LBFGS parameter estimation was performed with cut-offs at various numbers of iterations. I compared the two models, the Root Based (RB) feature model and the Pattern Based (PB) feature model separately. For each I evaluated as before with respect to correct analysis accuracy while simultaneously reading training log-likelihood and training accuracy. The results for RB model are shown in Figure 3.3 and Table 3.10, and for PB model are shown in Figure 3.4 and Table 3.11.

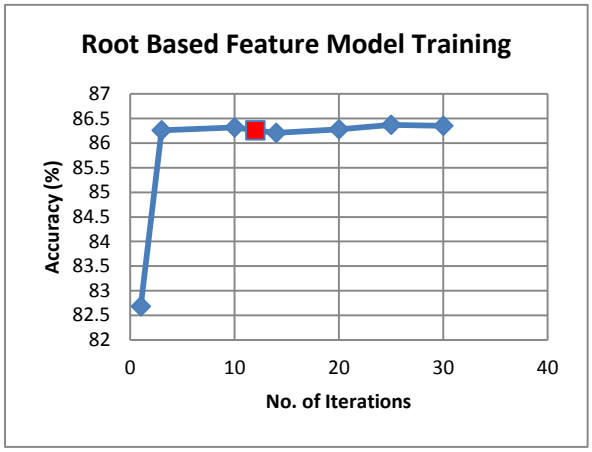


Figure 3.3: Illustration of Table 3.10

Itera-tion	Training Log-Like-lihood	Training Accu-racy(%)	Total Corr-ect	% Corr-ect
1	-7.20	59.54	4521	82.68
3	-1.07	91.45	4717	86.26
10	-9.54e-3	99.99	4720	86.32
12	-6.90e-3	100.00	4717	86.26
14	-3.72e-3	100.00	4714	86.21
20	-1.78e-3	100.00	4718	86.28
25	-9.52e-4	100.00	4723	86.37
30	-3.66e-4	100.00	4722	86.35

Table 3.10: Comparison of RB models trained at different iteration levels.

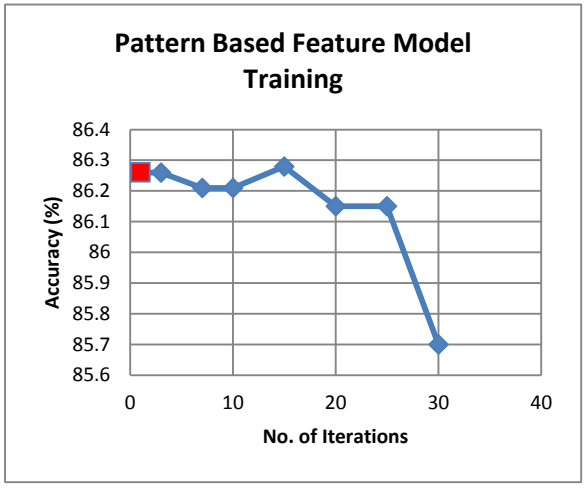


Figure 3.4: Illustration of Table 3.11

Iteration	Training Log-Likelihood	Training Accuracy(%)	Total Correct	% Correct
1	-3.61	100.000	4717	86.26
3	-1.25e-1	100.000	4717	86.26
7	-3.63e-3	100.000	4714	86.21
10	-7.33e-4	100.000	4714	86.21
15	-4.47e-5	100.000	4718	86.28
20	-1.22e-6	100.000	4711	86.15
25	-8.03e-8	100.000	4711	86.15
30	-5.66e-9	100.000	4686	85.70

Table 3.11: Comparison of PB models trained at different iteration levels.

The log-likelihood decreases consistently with each iteration in both models and is not a good indicator of a stopping point. Interestingly, the analysis accuracy for the two models shows opposite behavior. With each iteration of the RB model, accuracy tends to show very gradual improvement without showing any sign of over-fitting at higher iteration levels. For the PB model, accuracy tends to gradually fall, with a sudden drop seen at higher iterations. Here the PB model seems to be suffering from over-training.

A good indicator for the stopping point in both cases is where the training accuracy reaches 100%. For the RB model this is achieved at the 12th iteration (square marker in Figure 3.3) beyond which there is a very faint improvement in analysis accuracy. For the PB model, the training accuracy reaches 100% after the 1st iteration and hence it is stopped here or else the analysis accuracy may deteriorate further in subsequent iterations.

3.5.4.5 Length Adjustment Evaluation

Finally the different length adjustment methods are compared: cumulative raw probability, cumulative log scaled probability and morpheme length adjusted log scores. The three results are shown in Table 3.12.

Configuration	Total Correct	Percentage Correct
ME_RW	3491	63.84
ME_LS	4364	79.81
ME_LS_LA	4717	86.26

Table 3.12: Comparison of the methods using scaled score and length adjustment against the raw score

The poor performance of using raw probability values in calculating the cumulative morpheme score is visible in this comparison. With logarithmically scaled scores, accuracy is increased by approximately 6% over raw scoring. With length adjustment a further improvement of 7% is seen in comparison to the simple log-scaled value.

3.5.4.6 Comparison with the Baseline

The overall best configuration is based on PS_BBC, using LBFGS, without any Gaussian smoothing, and trained to give 100% accuracy on training data. This is used to score morphemes after logarithmically scaling probability values and length adjusting the scores. Table 3.13 shows the accuracy for the resulting lexicons of morphemes compared to the baseline lexicon. This configuration is more accurate than the baseline by 12.1 percentage points.

Configuration	Total Correct	Percentage Correct
ME	4717	86.26
Baseline	4055	74.16

Table 3.13: Comparison of the final ME model with the baseline

As stated earlier, two aspects of the ME based approach give it an advantage over the baseline: the clustering and the word proximity scores. In the baseline there is only one cluster which is the vocabulary set and a unit proximity value to mark the presence of morpheme in a word of the vocabulary.

3.6 System Design for Unsupervised Learning

For unsupervised learning, we would want to the system be fully automated, free from any kind of external parameter settings. I have aimed to achieve this, and let the system choose its parameters from the data without external intervention. The final system configuration is given in Table 3.14.

System Module	System Parameter	Automatic Parameter Setting
Word Cluster Formation	**Feature Generation**	RB features: Powerset combination of word characters where first and last character occur with boundary character PB features: Replace root characters in RB features with placeholder character and copy missing ones from word
	Parameter Estimation	LBFGS
	Gaussian Smoothing	None
	Number of training Iterations	When training data accuracy reaches 100%
Morpheme Extraction	**Size of Word Cluster**	Arbitrarily large
	Scoring	Logarithmically scaled probability value
	Length Adjustment	Exponential in terms of each morpheme length and independent of penalty strength parameter

Table 3.14: Final unsupervised ME based morphology induction system without any dependence on external parameters

3.7 Conclusions

This chapter addresses the task of analyzing the non-concatenative morphology of Arabic in an unsupervised manner. I adapted a technique for unsupervised concatenative morphological analysis described by DePauw & Wagacha (2007). They apply a machine learning approach using Maximum Entropy modeling to obtain groupings of words which are morphologically similar.

One novelty in my work is the selection of features which are suited to generalize over morphemes from the intercalated morphology of Arabic. Moreover, I was able to mirror the modeling procedure for two morpheme pairs in a word, the root and the pattern, by choosing features that are the converses of each other. I studied the different aspects of the machine learning process, experimenting with different types of feature sets, weight optimization methods, smoothing and number of iterations for training models.

In the subsequent step for morpheme extraction from morphologically related words I thoroughly investigated the morpheme scoring procedure. I introduced a procedure for logarithmic scaling which brings the ratios of related words into a comparable range and able to be combined arithmetically. Further, these scores were adjusted for the length of each morpheme.

Using a morphological analysis procedure based on the best root and pattern combination for each word I evaluated the inflected words of the Quranic Arabic Corpus for the correct identification of the root. The final ME system was devised such that the parameters of the system are automatically set by the system.

There are a number of areas that warrant further investigation. These include different morphological feature sets. The failure of Gaussian smoothing perhaps needs attention too. For morpheme extraction from word clusters, the range of word relatedness values is quite variable. Besides logarithmic scaling, further scaling procedures could be investigated.

Although reasonably effective and accurate, the procedure for morphological analysis using machine learning is computationally expensive. This could be a serious

impediment to the successful application of unsupervised natural language processing techniques that besides accuracy, need swift processing.

Indeed, the baseline obtained using a method for quickly obtaining lexicons was fairly competitive. This suggests that it would be fruitful to investigate this method further, to replace the machine learning based techniques with a conceptually simpler, rescoring technique; such a technique is described in the next chapter.

Chapter 4

Contrastive Learning

4.1 Introduction

The work described in this chapter addresses the same task as in the previous chapter: to analyse the non-concatenative morphology of the Arabic Language in an unsupervised manner. Thus the input and outcome remain the same but the approach differs. The machine learning approach to this task, described in Chapter 4, was a lengthy process having multiple stages of processing before obtaining the lexicons. This chapter builds on the method used to obtain the baseline in Chapter 4, which is a faster method to obtain morpheme rankings. The algorithm is comparable in accuracy to the one described in the previous chapter but it is much more computationally efficient, giving the output in a matter of seconds as compared to possibly hours in the machine learning case.

4.1.1 The Approach

The motivation for the approach is based on the desire to use simple counts of root and pattern morphemes co-occurring together in words of the vocabulary to develop a faster and efficient algorithm. The chapter describes a technique that develops into a link analysis algorithm for ranking morphemes. The insight behind the technique is that roots are linked to (co-occur with) a variety of patterns and likewise patterns are linked to a many roots. Mutual learning takes place in a recursive manner to identify potential morphemes. This computation of the ranking corresponds to using the power method from linear algebra to compute the eigenvector of the adjacency matrix representing the link structure of the root and pattern morphemes. I compare my algorithm to a very similar, well recognized algorithm for ranking webpages, the Hyperlink-Induced Topic Search (HITS) algorithm (Kleinberg, 1999).

4.1.2 Chapter Organization

Before describing the algorithms, some mathematical notations are first introduced in section 4.2. The contrastive morpheme learning approach is explained in section 4.3 and the contrastive learning approach is developed into mutually recursive algorithm, described in section 4.4. The comparison and application of the HITS algorithm is given in section 4.5. The morphological analysis process is outlined in section 4.6. The evaluation of the different scoring and rescoring methods is presented in section 4.7. Finally, section 4.8 concludes.

4.2 Preliminaries

In order to present the technique, it is necessary to define some mathematical notations. This section introduces the basic notations for word and morpheme formation, explains the process of morpheme derivation using a decomposition function and introduces the sets and other notation based on the morpheme derivation, which will be used in the remainder of the chapter.

4.2.1 Base Notations and Sets

The basic notations and sets are as follows:

- Lowercase-letter variables, such as, $w, r, p, s, \ldots \in \sum^*$ are character strings ranging over the alphabet $\sum$ and could represent words, morphemes, strings, etc.
- $V, R, P, \ldots \subseteq \sum^*$ are capital-letter variables ranging over sets of words, morphemes or strings
- $V = \{w_1, w_2, w_3, \ldots, w_n\}$ is the set of all word types in the vocabulary.
- $|\cdot|$ denotes either length of a string or cardinality of a set.
- The character sequence, $C_w = (c_1, c_2, \ldots, c_l, \ldots, c_{|w|})$ is the sequence of characters, c_l, constituting a word, w with l as the index position of the character in w with length $|w|$.
- $\div$ is a special character which is used to denote a *placeholder* or *slot* for character, c_l, in a word character sequence, C_w.

- $f_p(\div)$ denotes the frequency of occurrence of character $\div$ in the pattern p.
- $r \lhd w$: r is the potential root of a word, w, which is formed of a subset of characters, c_l of C_w along with the condition, $|r| < |w|$. In this chapter, $|r| \geq 3$. Thus, $C_r \subseteq C_w$ and $|w| > |C_w| \geq 3$; in other words, a root is formed of any possible combination of characters, c_l in $C_w$with a minimum length of three characters and a maximum length $|w| - 1$.
- $p \rhd w|r$: p is the potential pattern string derived from w given r, consisting of c_l in C_w and $\div$, such that we copy each character in w to p except those in r, which are replaced by $\div$ in p. Hence, $C_p = (C_w - C_r) \cap \{\div\}$ and also $f_p(\div) = |r| \geq 3$. Note that $|w| = |p|$ or $|C_w| = |C_p|$.
- $p \oplus r = w$: a pattern p and root r may combine to form a word w if $f_p(\div) = |r|$, such that each $\div$ in p is replaced by each character in r in sequence.
- $\langle r, p \rangle$: the pair of *adjoining* or *co-occurring* morphemes, r and p, such that $p \oplus r = w$.

4.2.2 Decomposition Function

A word may be decomposed into a set of constituent root and pattern pairs, each pair having a minimum of three characters or three slots for root radicals, respectively. Here we formalize the procedure for decomposing words described informally in chapter 4, by using a decomposition function, $D(w)$,

$$D{:}\, w \rightarrow D(w) = \{\, \langle r, p \rangle \mid r = r \lhd w \;\wedge\; p = p \rhd w : \\ w \in \mathrm{V}, C_r \in \mathbb{P}(C_w), f_p(\div) = |C_r| \geq 3 \;\} \tag{4.1}$$

In equation (4.1), r is derived from $w$such that r consists of all character sequences that can combine as a powerset combination of characters in w, $\mathbb{P}(C_w)$. Let D_w be the set of root and pattern pairs obtained using the decomposition function $D(w)$,

$$D_w = \{\langle r, p \rangle_1, \langle r, p \rangle_2, \ldots, \langle r, p \rangle_{L(|w|)}\} \tag{4.2}$$

The number of elements in D_w is given by the powerset cardinality (2^n) minus the sequences with fewer than 3 characters,

$$L(|w|) = 2^{|w|} - \binom{|w|}{2} - \binom{|w|}{1} - 1 \tag{4.3}$$

4.2.3 Further Notations and Sets

Based on the decomposition function, below are some set notations upon which the algorithms and scoring functions will be based.

- $D_V = D_{w_1} \cup D_{w_2} \cup D_{w_3} \ldots \cup D_{w_n}$: the set of all possible root and pattern pairs derived from every word of the vocabulary V.
- $P_w = \{p_1, p_2, \ldots p_{L(|w|)}\}$: the set of all possible patterns of the word w obtained from D_w
- $R_w = \{r_1, r_2, \ldots r_{L(|w|)}\}$: the set of all possible roots of the word w obtained from D_w
- $P_V = P_{w_1} \cap P_{w_2} \cap \ldots P_{w_n} = \{p_1, p_2, \ldots p_m\}$: the set of all possible patterns of all words obtained using the decomposition function, over the entire vocabulary
- $R_V = R_{w_1} \cap R_{w_2} \cap \ldots R_{w_n} = \{r_1, r_2, \ldots r_m\}$: the set of all possible roots of all words obtained using the decomposition function, over the entire vocabulary
- $P_r = \{p_1, p_2, \ldots p_{|\langle r, p_i \rangle|}\}$: the set of patterns $p_i \in P_V$ that occur with root r i.e. $|\langle r, p_i \rangle| \in D_V$
- $R_p = \{r_1, r_2, \ldots r_{|\langle r_i, p \rangle|}\}$: the set of roots $r_i \in R_V$ that occur with pattern p i.e. $|\langle r_i, p \rangle| \in D_V$

4.3 Contrastive Learning

Most accounts of morphology learning have focused attention on attempting to identify morphemes by looking at the frequency of occurrence of segments to distinguish the most frequent substrings as possible candidates. This is based on the notion that the more frequent a substring is, after adjusting for randomness, the more likely it is a candidate for being a morpheme. These methods usually look at the frequency of a substring directly to establish its significance.

Here I propose to learn the morphology gauging the importance of a morpheme by examining the frequency of occurrence of adjoining morphemes. Thus, for example the importance of a stem could be judged by the frequency of occurrence of adjoining affixes. Likewise, the significance of root could be gauged by the frequency of occurrence of the intercalated pattern. Based on the notion that entities may be revealed by their converses, I apply a contrastive scoring method to learn roots based on pattern counts, and patterns based on root counts. Thus, if a potential root occurs with a particular pattern, that pattern (if valid) should be fairly common in the dataset, hence should be assigned a high score to the root and vice versa. I refer to the score functions as the base scoring functions, and the outputs of these functions will be used in subsequent scoring processes.

4.3.1 Base Scoring Functions

Decomposing each word into all of its constituent root and pattern morpheme pairs, each candidate root is scored by counting the number of words with the co-occurring pattern. Each time the same root is encountered in other words in the vocabulary, the counts of co-occurring patterns are accumulated. In this way, each root's score is the frequency of all the patterns that the root occurs within the dataset. An analogous procedure is performed for each pattern.

Let S generically denote a scoring function for a morpheme type, then scoring functions $S^{\#}$ for root and pattern morphemes are,

$$S^{\#}(r) = \sum_{w_{i=1}}^{|V|} \sum_{w_{j=1}}^{|V|} \left(1 \left| \langle r,\ p \rangle \in D_{w_i} \ \wedge \ p \in P_{w_j} \right. \right) \tag{4.4}$$

$$S^{\#}(p) = \sum_{w_{i=1}}^{|V|} \sum_{w_{j=1}}^{|V|} \left(1 \left| \langle r,\ p \rangle \in D_{w_i} \ \wedge \ r \in R_{w_j} \right. \right) \tag{4.5}$$

The inner summation represents the co-occurring morphemes' frequencies, while the outer summation denotes the cumulative frequencies of all co-occurring morphemes for a scored morpheme over the entire vocabulary. The thing to note in these functions is that the morpheme score is not only dependent on the co-occurring morpheme's frequency count but also depends on its own frequency of occurrence. Thus the inner summation determines the co-occurring frequency counts while the outer summation is the accumulation of the score over the number of occurrences of the target scored morpheme. This is referred to below as *contrast-plus* scoring.

The scoring function S^{*} averages over the frequency counts of co-occurring morphemes and is thus independent of the frequency of occurrence of the target morpheme being scored. This is referred to below as *contrast-pure* scoring,

$$S^{*}(r) = \frac{1}{|P_r|} \sum_{w_{i=1}}^{|V|} \sum_{w_{j=1}}^{|V|} \left(1 \left| \langle r,\ p \rangle \in D_{w_i} \ \wedge \ p \in P_{w_j} \right. \right) \tag{4.6}$$

$$S^{*}(p) = \frac{1}{|R_p|} \sum_{w_{i=1}}^{|V|} \sum_{w_{j=1}}^{|V|} \left(1 \left| \langle r,\ p \rangle \in D_{w_i} \ \wedge \ r \in R_{w_j} \right. \right) \tag{4.7}$$

To illustrate the working of the scoring functions, shows an example for the transliterated Arabic word *yErf*. The word is decomposed into its constituents $\langle r, p \rangle_x \in D_{w=yErf}$, as shown in first two columns of the table. Example words containing root $r_x \in R_{w_i}$, are shown in column 3 with their counts in columns 5; likewise, words containing patterns, $p_x \in P_{w_i}$, are shown in column 4 with counts in column 6. This corresponds to the inner summation of the cumulative scoring formula for a particular morpheme, where each morpheme has been assigned a *local* score based on its occurrence in the word *yErf*.

Pattern, $p_x \in P_{w_i}$	**Root $r_x \in R_{w_i}$**	**Words, W_r**	**Words, W_p**	**$\lvert W_r \rvert$**	**$\lvert W_p \rvert$**
y---	Erf	'ErAf, Erf, ErfA, ErfAt, 'Etrf, mErwf, mErwfA, mErwfp, tEArf, tErf, yErf, ytEArf	yAbs, y$Aq, …yzyd, yzyg	12	490
-E--	Yrf	yErf, yHrf, yqtrf, yrfE, ysrf, ySrf, ytEArf	bEdA, bEDA, … yEZm, zEym	7	161
--r-	yEf	yDAEf, yEf, yEfw, yEfwA, yEkf, yErf, ystDEf, ystEff, ytEArf	$Ark, bArd,… zwrA, zxrf	9	280
---f	yEr	yEmr, yEr, yEr, yErD, yErf, yErj, yESr, yEt*r, ytEArf	'Asf, 'DEf,… 'zlf, zxrf	9	77

Table 4.1: The counts of morphemes in each word of the vocabulary (local score)

Next, consider the *global* score of a particular morpheme over the entire dataset. An illustration with the pattern and root morphemes, *y- - -* , and *Erf*, respectively, is shown in Table 4.2. Summing over all the local scores that the morpheme occurs in gives the global score of each morpheme. For example, the local score for *y- - -* in word *yErf*, as computed previously is 12, but its local score in word *yktb* is 18 which is the frequency of occurrence of root *ktb*. Thus summing over all the local scores gives the global sum

for the pattern morpheme *y- - -* as 2143. Using the first pair of scoring functions, $S^{\#}$, we can leave the score as the global sum. But the in the latter case, S^{*}, I take the average of all local scores. In this way the score for *y- - -* is independent of its count, distributed evenly amongst its 490 occurrences in the vocabulary.

Pattern $p_x \in P_V$	w_i	$r_y \in R_{p_x}$	$\lvert W_{r_y} \rvert$	$S^{\#}(p_x)$	$S^{*}(p_x)$
y---	yErf	Erf	12	12+	$\frac{2143}{490}$ =4.37
	Yktb	ktb	18	18+	
	Yslm	slm	19	19+	
	ybd'	bd'	3	3+ … = 2143	
	…	…	…	…	
…	…	…	…	…	…

Root $r_y \in R_V$	w_i	$p_x \in P_{r_y}$	$\lvert W_{r_y} \rvert$	$S^{\#}(p_x)$	$S^{*}(p_x)$
Erf	'ErAf	‘- - A -	132	132+ 507+ 45+ 88+ … = 1746	$\frac{1746}{12}$ =145.5
	ErfA	- - - A	507		
	ErfAt	- - - At	45		
	'Etrf	‘ - t - -	88		
	…	…	…	…	
…	…	…	…	…	…

Table 4.2: Aggregating and averaging the counts over all the whole vocabulary (global score)

Note the range of the scores $S^{\#}(r)$ and $S^{\#}(p)$ are quite similar because the combination of local and global scores balance each other out, bringing them into comparable range. This is not so for $S^{*}(r)$ and $S^{*}(p)$, where $S^{*}(r) \gg S^{*}(p)$, since the average frequency of occurrence of pattern morphemes is much greater than the average frequency of occurrence of root morphemes. Hence, some kind of normalization of scores is needed to bring the scores into a comparable range (see section 4.4.1).

4.3.2 Alternative Representation

The procedure described above for computing scores is computationally expensive and can be simplified by viewing the connections between root and pattern morphemes as links between two sets of vertices of a bipartite undirected graph (or bigraph). The graph, G, is defined as,

$$G = (R_V, P_V, E)$$

$$E = \{(r, p): r \in R_V\ , p \in P_V\}$$

The links (or edges), $e_i \in E$ in the graph correspond to the pair $\langle r, p\rangle_i \in D_V$. The morphemes r and p are thus *co-occurring* morphemes. The degree of a particular morpheme vertex corresponds to the number of patterns linked to that morpheme. For example the degree of any root r is equal to $|P_r|$. Also note the bigraph is balanced with $|R_V| = |P_V|$. Based on this representation, four more sets are defined:

- $RP = \{\langle r_1, P_{r_1}\rangle, \langle r_2, P_{r_2}\rangle, \ldots, \langle r_m, P_{r_m}\rangle\}$
- $PR = \{\langle p_1, R_{p_1}\rangle, \langle p_2, R_{p_2}\rangle, \ldots, \langle p_m, R_{p_m}\rangle\}$

Below is an example graph to clarify the concepts. This example is referenced in later parts of the chapter.

4.3.2.1 Example

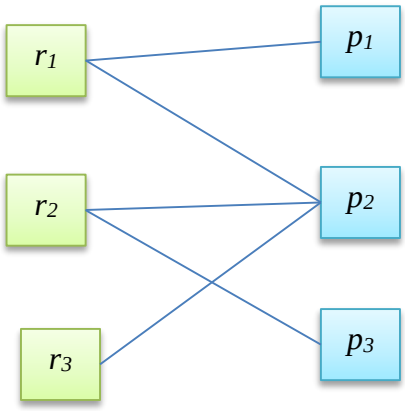


Figure 4.1: Example graph linking roots and patterns

Consider an example graph G_X, shown in Figure 4.1,

For G_X, the sets defined above are:

$$R_V = \{r_1, r_2\ , r_3\}$$

$$P_V = \{p_1, p_2\ , p_3\}$$

$$RP = \{\langle r_1, \{p_1, p_2\}\rangle, \langle r_2, \{p_2, p_3\}\rangle, \langle r_3, \{p_2\}\rangle\}$$

$$PR = \{\langle p_1, \{r_1\}\rangle, \langle p_2, \{r_1, r_2, r_3\}\rangle, \langle p_3, \{r_2\}\rangle\}$$

Table 4.3a and Table 4.3b tabulate the co-occurring morphemes in column 2, for each morpheme in column 1, along with the degree of the co-occurring morpheme in column 3. These morphemes and counts are referenced in the scoring functions defined in the next sections.

Pattern, $p_i \in P_V$	$r \in R_{p_i}$	$\lvert P_{r_i} \rvert$
$\boldsymbol{p_1}$	r_1	2
$\boldsymbol{p_2}$	r_1	2
	r_2	2
	r_3	1
$\boldsymbol{p_3}$	r_2	2

(a)

Root, $r_i \in R_V$	$p \in P_{r_i}$	$\lvert R_{p_i} \rvert$
$\boldsymbol{r_1}$	p_1	1
	p_2	3
$\boldsymbol{r_2}$	p_2	3
	p_3	1
$\boldsymbol{r_3}$	p_2	3

(b)

Table 4.3: Table showing co-occurring morphemes
and degree of co-occurring morphemes

4.3.3 Simplified Base Scoring Functions

It is possible to simplify the formulae, (4.4) and (4.5) in order to make a computationally less expensive algorithm, taking advantage of this representation to first defining initial, base scoring functions. For each r in $\langle r, P_r \rangle \in RP$,

$$S^{\#}(r) = \sum_{p_{i=1}}^{|P_r|} |R_{p_i}| \tag{4.8}$$

Conversely, for each p in $\langle p, R_p \rangle \in PR$,

$$S^{\#}(p) = \sum_{r_{i=1}}^{|R_p|} |P_{r_i}| \tag{4.9}$$

Likewise for the contrast-pure scoring,

$$S^{*}(r) = \frac{1}{|P_r|} \sum_{p_{i=1}}^{|P_r|} |R_{p_i}| \tag{4.10}$$

$$S^{*}(p) = \frac{1}{|R_p|} \sum_{r_{i=1}}^{|R_p|} |P_{r_i}| \tag{4.11}$$

These correspond to equations (4.6) and (4.7) defined earlier.

4.4 **Mutual Recursion**

As root and pattern scores are defined in terms of each other, it is possible to define a mutually recursive procedure to update the score of each morpheme using the other kind. Starting with the scores obtained from the base scoring functions as seeds, we can rescore each of the morphemes; and thereafter, iteratively rescore the morphemes using the previous scores until convergence in ranking is achieved. Since previous scores are being reapplied in subsequent iterations, there is a need to normalize the scores since they would otherwise increase without bound.

Let $\hat{S}$ denote the normalized scores and k denote the iteration. The iterative scoring functions for contrast-plus can be defined as,

$$S_k^{\#}(r) = \sum_{p_{i=1}}^{|P_r|} \hat{S}_{k-1}^{\#}(p_i) * |R_{p_i}| \tag{4.12}$$

$$S_k^{\#}(p) = \sum_{r_{i=1}}^{|R_p|} \hat{S}_{k-1}^{\#}(r_i) * \left|P_{r_i}\right| \tag{4.13}$$

and for contrast-pure,

$$S_k^{*}(r) = \frac{1}{|P_r|} \sum_{p_{i=1}}^{|P_r|} \hat{S}_{k-1}^{*}(p_i) * |R_{p_i}| \tag{4.14}$$

$$S_k^{*}(p) = \frac{1}{|R_p|} \sum_{r_{i=1}}^{|R_p|} \hat{S}_{k-1}^{*}(r_i) * |P_{r_i}| \tag{4.15}$$

It is important to distinguish the two types of convergence taking place: convergence in ranking and convergence in score values. In the absence of any normalization, convergence in ranking would be reached at some k, though with extremely high values for the scoring vectors (if at all computable). Section 4.5.1.4 presents a proof of convergence in ranking for the mutually recursive algorithms. With normalization, scores converge to certain fixed values as $k \to \infty$, while convergence in ranking would be reached at a certain value of k, remaining stable thereafter.

4.4.1 Score Normalization

Score normalization is achieved by simply dividing the score of each morpheme by the *norm* of its respective score vector. In other words, the vectors are converted into unit vectors, without changing their direction, thus keeping the relative ranking. Let $\hat{S}$ denote the normalized score, then

$$\hat{S} = \frac{S}{\|S\|} \tag{4.16}$$

where $\|S\|$ is the norm of the score vector S. Two types of norms are considered, the Manhattan norm, $\|S\|_1$ and the Maximum or max norm, $\|S\|_\infty$ defined respectively as

$$\|S\|_1 = \sum_{i=1}^{|R_V|} S(r_i) \tag{4.17}$$

$$\|S\|_\infty = \max(S(r_i)) \tag{4.18}$$

Empirically, these two norms have different behaviours. The max norm has the effect of scaling the two different quantities with respect to each other, bringing them into comparable range. The Manhattan norm preserves the relative differences in magnitudes

of quantities which are already in comparable range.

4.4.2 Initialization

An important consideration is how to initialize the scoring functions in the first iteration. There are three possible ways: (i) initialize both scoring functions $S(r)_1$ and $S(p)_1$ with normalized seed scores $\hat{S}_0$ as in (4.20) obtained by the **1**s vector from S_0 (4.19); (ii) initialize $S(r)_1$with the **1s** vector, and $S(p)_1$ with $S(r)_1$; or (iii) initialize $S(p)_1$ with the **1s** vector, and $S(r)_1$ with $S(p)_1$.

$$S(r)_0 = S(p)_0 = (1, 1, \ldots 1) \tag{4.19}$$

According to their respective norm definitions, each score vector is converted to a unit vector. For the Manhattan norm, $\|S_0\|_1 = \sum_{i=1}^{|R_V|} S(r_i) = \sum_{i=1}^{|P_V|} S(p_i) = |R_V| = |P_V| = m$, the total number of decomposed morphemes of either type. The normalized form is then:

$$\hat{S}(r)_0 = \hat{S}(p)_0 = (1/m, 1/m, \ldots 1/m) \tag{4.20}$$

For the Maximum norm the normalized score is the same as the normalized seed score.

$$\hat{S}(r)_0 = \hat{S}(p)_0 = (1/1, 1/1, \ldots 1/1) = (1, 1, \ldots 1) \tag{4.21}$$

This is because $\|S_0^*\|_\infty = \max(1, 1, \ldots 1) = 1$.

The choice between the two types of initialization may have a significant bearing on the outcome of the scores. Scores based on pattern counts are more accurate than those based on root counts since patterns are few and abundant hence easily identifiable. Root or pattern scores based on these counts as seeds are likely to give better results. Therefore the initialization choice depends on the scoring function used. For contrastive scoring, type (iii) is more appropriate where pattern seed counts are used to initialize

$S(r)_1$.

4.4.3 An Example

This section exemplifies the working of the mutually recursive ranking algorithm using the graph G_X of section 4.3.2.1. Table 4.4 and Table 4.5 show the first two of the N iterations the algorithm performs for the contrast-plus and contrast-pure scoring functions, respectively. I use type (i) initialization for the former and type (iii) for the latter. The Manhattan norm is used in both cases. These tables illustrate how the calculations of the scoring functions are performed at each iterative step. For this very simple graph the rankings converge in the first iteration and do not change with each iteration. For contrast-plus, the score has not converged in the first two iterations but the contrast-pure score converges at $k = 1$.

p/r	$S_1^{\#}(p)$	$S_1^{\#}(r)$	$\widehat{S}_1^{\#}(p)$	$\widehat{S}_1^{\#}(r)$	$S_2^{\#}(p)$	$S_2^{\#}(r)$	$\widehat{S}_2^{\#}(p)$	$\widehat{S}_2^{\#}(r)$	…
1	$\left(2 \times \frac{1}{3}\right) = 2/3$	$\left(1 \times \frac{1}{3}\right) + \left(3 \times \frac{1}{3}\right) = \frac{4}{3}$	$\frac{2/3}{3} = \frac{2}{9}$	$\frac{4/3}{11/3} = \frac{4}{11}$	$\left(2 \times \frac{4}{11}\right) = \frac{8}{11}$	$\left(1 \times \frac{2}{9}\right) + \left(3 \times \frac{5}{9}\right) = \frac{17}{9}$	$\frac{8/11}{35/11} = \frac{8}{35}$	$\frac{17/9}{43/9} = \frac{17}{43}$	…
2	$\left(2 \times \frac{1}{3}\right) + \left(2 \times \frac{1}{3}\right) + \left(1 \times \frac{1}{3}\right) = \frac{5}{3}$	$\left(3 \times \frac{1}{3}\right) + \left(1 \times \frac{1}{3}\right) = \frac{4}{3}$	$\frac{5/3}{3} = \frac{5}{9}$	$\frac{4/3}{11/3} = \frac{4}{11}$	$\left(2 \times \frac{4}{11}\right) + \left(2 \times \frac{4}{11}\right) + \left(1 \times \frac{3}{11}\right) = \frac{19}{11}$	$\left(3 \times \frac{5}{9}\right) + \left(1 \times \frac{2}{9}\right) = \frac{17}{9}$	$\frac{19/11}{35/11} = \frac{19}{35}$	$\frac{17/9}{43/9} = \frac{17}{43}$	…
3	$\left(2 \times \frac{1}{3}\right) = \frac{2}{3}$	$\left(3 \times \frac{1}{3}\right) = 1$	$\frac{2/3}{3} = \frac{2}{9}$	$\frac{1}{11/3} = \frac{3}{11}$	$\left(2 \times \frac{4}{11}\right) = \frac{8}{11}$	$\left(3 \times \frac{5}{9}\right) = \frac{15}{9}$	$\frac{8/11}{35/11} = \frac{8}{35}$	$\frac{15/9}{43/9} = \frac{15}{43}$	…

Table 4.4: Mutual recursion for contrast-plus scoring using type (i) initialization and the Manhattan norm

p/r	$S_1^*(p)$	$\widehat{S}_1^*(p)$	$S_1^*(r)$	$\widehat{S}_1^*(r)$	$S_2^*(p)$	$\widehat{S}_2^*(p)$	$S_2^*(r)$	$\widehat{S}_2^*(r)$	...
1	$\left(2\times\frac{1}{3}\right)/1$ $=2/3$	$\frac{2/3}{17/9}$ $=\frac{6}{17}$	$\left(1\times\frac{6}{17}\right)+$ $\left(3\times\frac{5}{17}\right)/2$ $=\frac{21}{34}$	$\frac{21/34}{36/17}$ $=\frac{7}{24}$	$\left(2\times\frac{7}{24}\right)/1$ $=\frac{7}{12}$	$\frac{7/12}{119/72}$ $=\frac{6}{17}$	$\left(1\times\frac{6}{17}\right)+$ $\left(3\times\frac{5}{17}\right)/2$ $=\frac{21}{34}$	$\frac{21/34}{36/17}$ $=\frac{7}{24}$	...
2	$\left(2\times\frac{1}{3}\right)+$ $\left(2\times\frac{1}{3}\right)+$ $\left(1\times\frac{1}{3}\right)/3$ $=\frac{5}{9}$	$\frac{5/9}{17/9}$ $=\frac{5}{17}$	$\left(3\times\frac{5}{17}\right)+$ $\left(1\times\frac{6}{17}\right)/2$ $=\frac{21}{34}$	$\frac{21/34}{36/17}$ $=\frac{7}{24}$	$\left(2\times\frac{7}{24}\right)+$ $\left(2\times\frac{7}{24}\right)+$ $\left(1\times\frac{7}{24}\right)/3$ $=\frac{35}{72}$	$\frac{35/72}{119/72}$ $=\frac{5}{17}$	$\left(3\times\frac{5}{17}\right)+$ $\left(1\times\frac{6}{17}\right)/2$ $=\frac{21}{34}$	$\frac{21/34}{36/17}$ $=\frac{7}{24}$	...
3	$\left(2\times\frac{1}{3}\right)/1$ $=\frac{2}{3}$	$\frac{2/3}{17/9}$ $=\frac{6}{17}$	$\left(3\times\frac{5}{17}\right)=\frac{15}{17}$	$\frac{15/17}{36/17}$ $=\frac{5}{12}$	$\left(2\times\frac{7}{24}\right)/1$ $=\frac{7}{12}$	$\frac{7/12}{119/72}$ $=\frac{6}{17}$	$\left(3\times\frac{5}{17}\right)=\frac{15}{17}$	$\frac{15/17}{36/17}$ $=\frac{5}{12}$	...

Table 4.5: Mutual recursion for contrast-pure scoring using type (iii) initialization and the Manhattan norm

4.5 **Hyperlink-Induced Topic Search**

I now compare my algorithm to a very similar algorithm used for ranking webpages, which is well recognized in the field of Information Retrieval, known as Hyperlink-Induced Topic Search (HITS), and also sometimes referred to as the Hubs and Authorities algorithm. This algorithm was developed by Jon Kleinberg (1999), and was a seminal contribution to the family of Link Analysis Ranking (LAR) algorithms used to rank webpages. HITS was a precursor to the PageRank algorithm (Brin *et al*, 1998) currently in use by Google.

I will first discuss the background of this algorithm outlining how the problem of page ranking as described by Kleinberg relates to my work, and then apply the technique to rank morphemes. Finally, I describe the proof of ranking convergence for both HITS and my approach.

Background

There are two types of pages relevant to webpage ranking: authority pages and hub pages. Hubs appear as sizable catalogues acting as gateways to authority pages which actually hold the information useful for a particular information request. Thus, hub pages direct users to useful webpages which are an authority on a particular subject of user interest. The aim is then to distinguish good hub pages from good authority pages. One way proposed by Kleinberg, is to consider those pages which link to many other pages as good hubs, and those pages that are linked to by many other pages as potentially good authority pages. Each page is assigned two scores, a hub score and an authority score. Links to/from important pages, having a high score, in turn contribute to a higher ranking for the page with respect to either the hub or authority being scored.

Procedure

The algorithm works at query time, unlike its successor the PageRank algorithm which computes scores at indexing time. In HITS, the search query is first used to retrieve relevant pages known as the root set. This set is then augmented with pages that link to pages in this set and those pages that are linked from the root set. The augmented set is called the base set. The idea behind making such a set, according to Kleinberg, is to gather the most important authorities. This set with interlinks between webpages, forms a 'focused sub-graph' which is a directed graph with edges indicating the linkages between the pages. This is similar to the bipartite graph for morphemes in section 4.3.2.

Likewise in a similar way to the calculation of morpheme scores, the authority and hub scores are defined in terms of each other in a mutually recursive relationship. The authority score for a page is computed as the sum of the hub scores that point to the page. Conversely, the hub score for a page is computed by summing the scores of the authority scores of the pages that are pointed to by the hub page. Kleinberg uses the Euclidean norm to normalize the scores after each iteration.

Application of HITS to Morphology

Applying the HITS algorithm to the morphology learning task is quite straightforward, and the root and pattern scores can be computed in a similar way as with contrastive learning:

$$S_k^H(r) = \frac{1}{|P_r|}\sum_{p_{i=1}}^{|P_r|} \hat{S}_{k-1}^H(p_i) \tag{4.22}$$

For $k = 0,1,2$ ...

$$S_k^H(p) = \frac{1}{|R_p|}\sum_{r_{i=1}}^{|R_p|} \hat{S}_{k-1}^H(r_i) \tag{4.23}$$

For $k = 1,2,3$ ...

As can be seen from the formulation of the HITS scoring functions, the initial values of the scored morpheme directly depend on counts of the morpheme, unlike in contrastive learning where they are determined by the counts of the co-occurring morphemes. Subsequently, the counts are scaled according to the score of each co-occurring morpheme which in turn have initially been determined by their own occurrence counts. So if root morpheme scores are taken as the seed scores, then the initial scores for the root morphemes are

$$S_0^H(r) = \sum_{p_{i=1}}^{|P_r|} 1 = |P_r| \tag{4.24}$$

We can easily see that $|P_r|$ is the count of the number of words in which r occurs as opposed to the contrastive case where the initial score for r would be $|R_p|$, i.e. the number of words with p such that $\langle r, p\rangle \in D_V$. Thus in essence both algorithms score morphemes through mutual reinforcement but the key difference lies in the contribution

through either self or affiliate morpheme. One can perceive HITS is a simpler version of the contrastive learning algorithm which although computed differently would eventually give a similar or the same ranking. This will be seen in more detail in the evaluation (section 4.7.3).

Applying HITS scoring to the example graph G_X from section 4.3.2.1, Table 4.6 shows the scores for the root and pattern morphemes for the first two iterations but using the Manhattan norm for comparison purposes, rather than the Euclidean norm as in the original HITS implementation. The tables show again that ranking convergence is reached in the first iteration but scoring convergence is not reached in these iterations as for the contrast-plus case.

Table 4.7 shows the ranking for the roots and patterns according to contrastive learning and according to HITS. While relative rankings for p_1, p_3 and r_1,r_2 stay the same due to their similar link structure, it is noteworthy to see that the ranking for r_3 and p_2 is reversed in the two types of scoring, since one gives more emphasis to the morphemes with more links from itself while the other gives more importance to the number of links of associated morphemes. So, the graph shows that r_3 itself has degree one, therefore getting a lower score with HITS but its only co-occurring morpheme has degree three, hence it is given more importance by contrastive learning.

p/r	$S_1^H(p)$	$\widehat{S}_1^H(p)$	$S_1^H(r)$	$\widehat{S}_1^H(r)$	$S_2^H(p)$	$\widehat{S}_2^H(p)$	$S_2^H(r)$	$\widehat{S}_2^H(r)$	...
1	$\frac{1}{3}$	$\frac{1}{3}/\frac{5}{3}$ $=\frac{1}{5}$	$\frac{1}{5}+\frac{3}{5}$ $=\frac{4}{5}$	$\frac{4}{5}/\frac{11}{5}$ $=\frac{4}{11}$	$\frac{4}{11}$	$\frac{4}{11}/\frac{19}{11}$ $=\frac{4}{19}$	$\frac{4}{19}+\frac{11}{19}$ $=\frac{15}{19}$	$\frac{15}{19}/\frac{41}{19}$ $=\frac{15}{41}$	...
2	$\frac{1}{3}+\frac{1}{3}+$ $\frac{1}{3}=1$	$3/\frac{5}{3}$ $=\frac{3}{5}$	$\frac{3}{5}+\frac{1}{5}$ $=\frac{4}{5}$	$\frac{4}{5}/\frac{11}{5}$ $=\frac{4}{11}$	$\frac{4}{11}+\frac{4}{11}$ $+$ $\frac{3}{11}=1$	$1/\frac{19}{11}$ $=\frac{11}{19}$	$\frac{11}{19}+\frac{4}{19}$ $=\frac{15}{19}$	$\frac{15}{19}/\frac{41}{19}$ $=\frac{15}{41}$	...
3	$\frac{1}{3}$	$\frac{1}{3}/\frac{5}{3}$ $=\frac{1}{5}$	$\frac{3}{5}$	$\frac{3}{5}/\frac{11}{5}$ $=\frac{3}{11}$	$\frac{4}{11}$	$\frac{4}{11}/\frac{19}{11}$ $=\frac{4}{19}$	$\frac{11}{19}$	$\frac{11}{19}/\frac{41}{19}$ $=\frac{11}{41}$	...

Table 4.6: Mutual recursion for HITS
using type (ii) initialization and the Manhattan norm

Contrast-Pure (S^*)		**HITS (S^H)**	
Root	**Pattern**	**Root**	**Pattern**
r_3	p_1/p_3	r_1/r_2	p_2
r_1/r_2	p_1/p_3	r_1/r_2	p_1/p_3
r_1/r_2	p_2	r_3	p_1/p_3

Table 4.7: Root and pattern ranking comparison
between HITS and contrast-pure

4.5.1 Proof of Convergence

The proof of convergence of the mutually recursive algorithms given below uses concepts from linear algebra. The approach to proving ranking convergence is similar to one presented in the literature (Borodin *et al*, 2005; Tsaparas, 2004) on linear link analysis algorithms used for ranking pages of a network based on links between pages, such as PageRank, HITS, etc.– also referred to as eigenvector-based ranking algorithms. Here I first adapt the proof for the convergence of the HITS algorithm applied to morpheme ranking, which is congruent to the page ranking problem. Thereafter, I prove convergence for the contrastive algorithms. But firstly, I translate the problem and formulae into an algebraic representation.

4.5.1.1 Algebraic Representation

I assume the bipartite graph as defined in section 4.3.2 to represent the link structure of the morphemes. Thus, in contrast to LAR, where a graph is defined as directed with edges from page to page, the graph for morpheme ranking

$$G = (R_V, P_V, E)$$
$$E = \{(r, p): r \in R_V\ , p \in P_V\} \tag{4.25}$$

maps to a $m \times m$ adjacency matrix, A, where the rows are represented by ith root entries and columns with jth pattern entries. If there is a link from r_i to p_j in graph G, then, $a_{ij} = 1$; all other entries of the matrix are 0. An example adjacency graph for the example G_X is:

$$A_{G_X} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{4.26}$$

Taking the sum of the columns results in a vector where each element i, corresponds to the count, $|P_{r_i}|$. Let this resulting sum of columns vector be ϕ_A. Similarly, summing vertically over all the rows i gives a vector with each element j corresponding to the

count $|R_{p_j}|$. This latter vector is equivalent to taking the sum of rows i, horizontally over the transpose of the adjacency matrix, A^T, which is represented as the vector, ϕ_{A^T}. Further, I refer to inverse of the horizontal and vertical summation vectors, ${\phi^{-1}}_A$ and ${\phi^{-1}}_{A^T}$, as corresponding to values $1/|P_{r_i}|$ and $1/|R_{p_j}|$, respectively. Thus for A_{G_X},

$$\phi_{A_{GX}} = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}, \qquad \phi_{A_{GX}{}^T} = \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix},$$
$$\phi^{-1}{}_{A_{GX}} = \begin{pmatrix} 1/2 \\ 1/2 \\ 1/1 \end{pmatrix}, \qquad \phi^{-1}{}_{A_{GX}{}^T} = \begin{pmatrix} 1/1 \\ 1/3 \\ 1/1 \end{pmatrix} \tag{4.27}$$

Let $\vec{r}_k$ be the root weight vector used to represent the scores $S_k(r_1), S_k(r_2), S_k(r_3), \ldots S_k(r_m)$ at a particular iteration k. Similarly let $\vec{p}_k$ be the pattern weight vector used to represent the scores $S_k(p_1), S_k(p_2), S_k(p_3), \ldots S_k(p_m)$ at iteration k. $\vec{r}_k$ and $\vec{p}_k$ are both column vectors.

The scoring and rescoring functions can be expressed in terms of the vector based representation with A as the adjacency matrix of graph G containing links between roots and patterns, and weight vectors, $\vec{r}$ and $\vec{p}$, respectively. Denoting M^T as the transpose of matrix M, the updated weights for the root and pattern, as described in the equations (4.22) and (4.23) for HITS, are written using vector representation as

$$\vec{r}_k = \rho_k A\, \vec{p}_{k-1} \tag{4.28}$$

$$\vec{p}_k = \gamma_k A^T\, \vec{r}_k \tag{4.29}$$

where $\gamma_k$and ρ_k are normalization constants to ensure that the root and pattern weight vectors are normalized:

$$\rho_k = \frac{1}{\|\vec{r}_{k-1}\|}, \qquad \gamma_k = \frac{1}{\|\vec{p}_{k-1}\|} \tag{4.30}$$

For contrast-plus, let Φ represent the diagonal matrix,

$$\Phi_A = \text{diag}(\phi_A) \text{ and } \Phi_{A^T} = \text{diag}(\phi_{A^T}) \tag{4.31}$$

then,

$$\vec{r}_k = \rho_k A\, \Phi_A\, \vec{p}_{k-1} \tag{4.32}$$

$$\vec{p}_k = \gamma_k A^T \Phi_{A^T}\, \vec{r}_{k-1} \tag{4.33}$$

Where

$$\rho_k = \frac{1}{\|\Phi_A \vec{r}_{k-1}\|}, \quad \gamma_k = \frac{1}{\|\Phi_{A^T} \vec{p}_{k-1}\|} \tag{4.34}$$

For contrast-pure, let Ψ represent the diagonal matrix,

$$\Psi_A = \text{diag}(\phi_A)\text{diag}(\phi^{-1}{}_{A^T}) \text{ and } \Psi_{A^T} = \text{diag}(\phi_{A^T})\text{diag}(\phi^{-1}{}_{A}) \tag{4.35}$$

then,

$$\vec{r}_k = \rho_k A\, \Psi_A\, \vec{p}_{k-1} \tag{4.36}$$

$$\vec{p}_k = \gamma_k A^T \Psi_{A^T}\, \vec{r}_{k-1} \tag{4.37}$$

where

$$\rho_k = \frac{1}{\|\Psi_A \vec{r}_{k-1}\|}, \quad \gamma_k = \frac{1}{\|\Psi_{A^T} \vec{p}_{k-1}\|} \tag{4.38}$$

4.5.1.2 Background Concepts

For a symmetric matrix M, $n \times n$ there exists a vector v, which when multiplied by M, yields a constant multiple of v:

$$Mv = \lambda v \tag{4.39}$$

Vector v is referred to as the eigenvector and the multiplier, λ, is referred to as the eigenvalue corresponding to v.

The **Perron-Frobenius Theorem** states that if M is a *non-negative*[6] (i.e. all values are ≥ 0) *irreducible* or a *symmetric* square matrix, then there exists an eigenvalue a such that the modulus of all other eigenvalues does not exceed a. Corresponding to this eigenvalue an eigenvector can be chosen which is also non-negative.

The set of all eigenvectors associated with a particular eigenvalue λ is known as the *eigenspace* of matrix M from the space $\mathbb{R}^n$. The dimension of this space is the *multiplicity* of λ. Since M is symmetric, the set of all eigenvalues is real and is known to have at most n distinct eigenvalues summing over all multiplicities. The eigenvalues of Mcan be written with multiplicities indexed in order of decreasing magnitude:

$$|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \geq \cdots |\lambda_n| \tag{4.40}$$

For each eigenvalue λ_i, there exists a corresponding eigenvector vector v_i, such that the eigenvectors are an orthogonal *basis* of their respective eigenspaces. A *dominant* or *principal* eigenvector exists such that the eigenvalue associated with this eigenvector has the largest magnitude. In turn this eigenvalue is also referred to as the dominant or largest eigenvalue of the matrix. Thus, if the assumption $|\lambda_1| > |\lambda_2|$ holds, λ_1 is the dominant or principal eigenvalue.

---

[6] The original theorem of Perron requires M to have positive entries, but this was extended to non-negative by Frobenius.

4.5.1.3 The Proof (HITS)

Combining the two equations (4.28) and gives a method for calculating the score vectors in term of the score vector of the same morpheme from the previous iteration,

$$\vec{r}_k = \rho_k \gamma_{k-1} (AA^T) \vec{r}_{k-1} \tag{4.41}$$

$$\vec{p}_k = \gamma_k \rho_k (A^T A) \vec{p}_{k-1} \tag{4.42}$$

Note that the product of the adjacency matrix with its transpose, AA^T and $A^T A$, gives a symmetric matrix which will be useful in the derivation of the *principal* eigenvector in the proof of convergence, below.

As k grows large, the two weight vectors converge to $\vec{r}_k{}^*$ and $\vec{p}_k{}^*$, respectively,

$$\vec{r}_k{}^* = \lambda^* AA^T \vec{r}_k{}^* \tag{4.43}$$

$$\vec{p}_k{}^* = \lambda^* A^T A \vec{p}_k{}^* \tag{4.44}$$

The convergent vectors $\vec{r}_k{}^*$ and $\vec{p}_k{}^*$ each correspond to the dominant eigenvectors of matrices AA^T and $A^T A$, respectively, which are both symmetric, and λ^* as the *dominant* eigenvalue for each of the matrices. Thus we must prove that the sequence $\{\vec{r}_k\}$ converges to a vector, $\vec{r}_k{}^*$, which is a non-negative eigenvector of the largest eigenvalue of AA^T, λ^*. Likewise, the pattern vector sequence, $\{\vec{p}_k\}$, converges to a vector $\vec{p}_k{}^*$, which again is a non-negative eigenvector of the largest eigenvalue of $A^T A$, λ^*.

Since AA^T and $A^T A$ are symmetric their eigenvalues are real and non-negative. As seen above, while the largest modulus eigenvalue can have multiplicity greater than 1, all the other eigenvalues would have a smaller magnitude. The eigenspaces corresponding to each distinct eigenvalue would be orthogonal. In the dominant eigenspace for the largest eigenvalue, we can choose any non-negative orthogonal vector. Since the initial vectors $\vec{r}_0$ and $\vec{p}_0$ are positive, they are not orthogonal to the chosen dominant eigenvectors

which are also non-negative, i.e. the dot product of $\vec{r}_0$ or $\vec{p}_0$ with the principal eigenvector is positive. According to the *Von Mises iteration* algorithm (von Mises & Pollaczek-Geiringer, 1929), since $\vec{r}_0$ or $\vec{p}_0$ would have a nontrivial component in the eigenspace of the principal eigenvector, the vectors $\{\vec{r}_k\}$ and $\{\vec{p}_k\}$ in the same direction would converge to their respective dominant or principal eigenvectors, $\vec{r}_k{}^*$ and $\vec{p}_k{}^*$ with largest eigenvalue modulus λ^* (Golub & Van Loan, 1989).

An alternative way to look at this convergence of the algorithm, as described by Farahat *et al* (2006), is to consider the pattern vector $\vec{p}$ as a linear combination of the eigenvectors, $\vec{v}_1, \vec{v}_2, \vec{v}_3, \ldots \vec{v}_n$.

$$\vec{p}_1 = c_1\vec{v}_1 + \cdots c_n\vec{v}_n \tag{4.45}$$

where $c_i = \vec{p}_1 \cdot \vec{v}_i / \|\vec{v}_i\|$

$$\vec{p}_2 = \lambda_1\alpha_1\vec{v}_1 + \cdots \lambda_n\alpha_n\vec{v}_n \tag{4.46}$$

or

$$\vec{p}_2 = \lambda_1(\alpha_1\vec{v}_1 + \cdots \alpha_r\vec{v}_r) + \lambda_{r+1}\alpha_{r+1}\vec{v}_{r+1} + \cdots \lambda_n\alpha_n\vec{v}_n \tag{4.47}$$

if, $\lambda_1 = \lambda_2 = \cdots = \lambda_r \neq \lambda_{r+1}$

Subsequently,

$$\vec{p}_{k+1} = \xi_k\left(\lambda_1^k[\alpha_1\vec{v}_1 + \cdots \alpha_r\vec{v}_r] + \sum_{i=r+1}^{n} \lambda_i^k\alpha_i\vec{v}_i\right) \tag{4.48}$$

where $\xi_k = \rho_k\gamma_k\xi_{k-1}$

As k grows large, λ_1^k dominates, thus,

$$\vec{p}_k \to c(\alpha_1 \vec{v}_1 + \cdots \alpha_r \vec{v}_r) \tag{4.49}$$

and

$$\rho_k \gamma_k \to \lambda_1 \tag{4.50}$$

Thus the pattern weight vector $\vec{p}_k$ converges to the eigenvector of the dominant eigenvalue λ_1.

4.5.1.4 Convergence for Contrastive Learning

The two equations (4.32) and (4.33) for contrast-plus scoring combine to calculate the score vectors in terms of themselves,

$$\vec{r}_k = \rho_k \gamma_{k-1} \, (A \Phi_A A^T) \, \Phi_{A^T} \, \vec{r}_{k-1} \tag{4.51}$$

$$\vec{p}_k = \gamma_k \rho_k \, (A^T \Phi_{A^T} A) \, \Phi_A \, \vec{p}_{k-1} \tag{4.52}$$

Similarly to AA^T and $A^T A$, $A\Phi_A A^T$ and $A^T \Phi_{A^T} A$ are also symmetric with eigenvalues that are real and non-negative. Since Φ_{A^T} and Φ_A are positive their products with the initial vectors, $\vec{r}_0$ and $\vec{p}_0$, i.e. $\Phi_{A^T} \vec{r}_0$ and $\Phi_A \vec{p}_0$ are positive vectors. They are not orthogonal to the chosen dominant non-negative eigenvectors since the dot product of $\Phi(A^T) \, \vec{r}_0$ or $\Phi_A \vec{p}_0$ with the principal eigenvector is positive. Thus, the vectors $\{\Phi_{A^T} \vec{r}_k\}$ and $\{\Phi_A \vec{p}_k\}$ are in the same direction as $\Phi_{A^T} \, \vec{r}_0$ and $\Phi_A \vec{p}_0$ and converge to their dominant eigenvectors, $\vec{r}_k{}^*$ and $\vec{p}_k{}^*$ with largest eigenvalue modulus λ^*

For contrast-pure scoring the equations (4.36) and (4.37) are combined:

$$\vec{r}_k = \rho_k \gamma_{k-1} \, (A \Psi_A A^T) \, \Psi_{A^T} \, \vec{r}_{k-1} \tag{4.53}$$

$$\vec{p}_k = \gamma_k \rho_k \, (A^T \Psi_{A^T} A) \, \Psi_A \, \vec{p}_{k-1} \tag{4.54}$$

4.6 **Morphological Analysis**

As in Chapter 4, I run a set of experiments to perform morphological analysis for every word in the dataset using the root and pattern lexicons obtained using each scoring technique. Each word to be analysed is decomposed using the decomposition function, D_w. For each $\langle r, p \rangle$ in D_w the analysis score is computed using the respective lexicons containing normalized scores for the morphemes. There are several ways to combine the morpheme scores to obtain the analysis score. In each of the cases, the combined analysis scores are ranked from highest to lowest score, revealing the best analyses on top. As the scores are normalized there is no need to scale the scores as was done in the previous chapter.

One type of combination is considered in the previous chapter, where the scores of the morphemes from their respective lexicons are added linearly. This type of analysis score computation is suited for contrast-plus scoring where the morpheme scores are an amalgamation of root and pattern occurrences. For a word w, given the analysis $\langle r, p \rangle_i$ in D_w, the maximum analysis score is selected as the output:

$$\max_{i=1..L(|w|)} (\hat{S}^{\#}(r_i) + \hat{S}^{\#}(p_i)) \tag{4.55}$$

For contrast-pure and HITS, the scores for each morpheme are wholly representative of one morpheme type. If one score was computed on the basis of root occurrences then the other would be computed in terms of pattern occurrences. Hence a more appropriate method of analysis score calculation is to take the product of scores of roots and patterns. The best analysis would be selected as the maximum of the product of the pair of scored morphemes:

$$\max_{i=1..L(|w|)} (\hat{S}^{*}(r_i) \times \hat{S}^{*}(p_i)) \tag{4.56}$$

$$\max_{i=1..L(|w|)} (\hat{S}^{H}(r_i) \times \hat{S}^{H}(p_i)) \tag{4.57}$$

4.7 Evaluation

The evaluation is performed using the Quranic Arabic Corpus (QAC) with the same setup as in section 3.5. The evaluation measure is also the same as before: the percentage of roots that are correctly analysed against the correctly identified roots available for the QAC. The total number of evaluated words is 5481[7]. The evaluation is divided into three parts. The first part compares the different base scoring strategies for the three types of scoring methods, contrast-plus, contrast-pure and HITS, taking into consideration the various features and configurations. The second part evaluates the iterative mutual recursive scoring technique applied to each of the scoring methods. The final part evaluates the refinement procedure for contrast-pure and HITS.

4.7.1 Base Scoring Evaluation

I start by comparing the performance of the base scoring functions i.e. contrast-plus, contrast-pure and HITS without recursion. I also consider these as the baselines to which mutually recursive scoring is compared. These can be considered as the first iteration of the mutually recursive algorithm. At this stage, I also look at the effect of different norms used, initialization using root or pattern or both simultaneously, and finally the two ways of aggregating morpheme scores using summation or product.

[7] This number is slightly different to that in section 3.5 as there were some words whose root characters needed to undergo normalization; hence 13 additional words now formed part of the evaluation set.

Starting with a seed score of all ones, I perform sequential score updates starting with either roots or patterns. This sequence of dual updates – firstly for the chosen morpheme score with the seed score, and secondly based on counterpart morpheme scores – constitutes one cycle and corresponds to the first iteration of the recursive algorithm. The choice of which morpheme to start with, either root first then pattern (Root-Pattern) or pattern first then root (Pattern-Root), may be of importance for the base scoring functions. I compare this with parallel or simultaneous update of both pattern and root both initialized using an all ones seed score. The results for the three scoring functions are shown in Table 4.8**,** showing the number of correctly identified roots from a total of 5481 evaluated words. The scoring functions for these outputs have been normalized using the Manhattan norm and the analysis score computed by taking the product of morpheme scores.

Initialization	**Contrast-plus**	**Contrast-pure**	**HITS**
Root-Pattern	4632	4321	2805
Pattern-Root	4503	3343	3506
Simultaneous	4293	3900	3026

Table 4.8: Numbers of correct analyses using different initializations

The best performance is exhibited by Root-Pattern for both contrastive scoring methods, but for HITS, Pattern-Root gives the best performance. Here for all three scoring methods, the seed score counts are computed based on pattern occurrence counts. This shows that the counts for patterns are more reliable and accurate than those for roots. This is understandable because there are relatively few patterns but their frequency of occurrence is very high. This makes them easily distinguishable from other morphemes. Triliteral roots occur in large variety having fewer individual counts, and also sometimes overlapping with three letter affixes. Hence these are less easily discernible than their counterparts. For all three methods, simultaneous scoring of morphemes shows inferior performance to the other initializations. All the experimental results below use Root-Pattern initialization for contrastive learning and Pattern-Root for HITS.

Next I look at two types of norms, Manhattan and maximum. Each is a different way to measure the size of a vector. Here the purpose of using the norm is to scale the scores in order to make them comparable and to prevent overflow or underflow for the recursive algorithm. Alongside using the different norms, I look at the way that the analysis score is combined – either using summation or product. The results are shown in Table 4.9.

	Manhattan		**Maximum**	
	Summation	**Product**	**Summation**	**Product**
Contrast-plus	4799	4632	4346	4632
Contrast-pure	4190	4321	3772	4321
HITS	2977	3506	3524	3506

Table 4.9: Comparison using different norms
and analysis scoring combinations

One thing that can readily be seen is that using product to obtain analysis scores makes the scoring independent of the type of norm: the scores for Manhattan and maximum are the same when using product. This is to be expected, as the ranking is independent of the normalization. Also, the results are generally superior when using product. The scores for roots and patterns are computed in terms of each other, hence using product brings out the best morpheme composition in the word. The main exception is contrast-plus where much better results are obtained using summation, normalized using the Manhattan norm. As stated earlier, the morpheme scores in this are a balanced combination of root and pattern occurrences. The Manhattan norm scales yet preserves the relative differences in magnitudes of scored quantities for contrast-plus morpheme scores which are in a comparable range. Thus a linear combination of the quantities yields a better solution.

In all the following experiments, contrast-pure and HITS use product to obtain analysis scores; contrast-plus uses summation and the Manhattan norm for score scaling. The three base scoring methodologies are compared in the which shows their percentage accuracies.

	Correct	Accuracy (%)
Contrast-plus	4799	87.56
Contrast-pure	4321	78.88
HITS	3506	63.40

Table 4.10: Comparison of the best performance
of the three base scoring methods

Table 4.10 clearly shows the advantage of using contrast-plus scoring. Contrastive learning in which morphemes are scored based on all co-occurring morpheme counts, perhaps performs better because it is computed based on statistics of both morpheme types while the other two scoring functions are computed using one morpheme type; the latter two methods could be expected to show better performance when the scores are combined in the mutually recursive calculation process.

4.7.2 Mutually Recursive Rescoring Evaluation

The next set of experiments explores the approach where scoring functions are subject to iterative improvement based on scores computed in the previous cycle in a mutually recursive relationship, until convergence is achieved. There are several aspects to investigate besides identifying the best scoring function, for example rates of convergence, the levels of improvement etc.

Starting with their respective base scores, the recursive functions are repeatedly applied until convergence. I chose $N = 10$ as a sufficiently large number of iterations in order to achieve convergence but as seen in the results below convergence is reached before the 6^{th} iteration.

No. of Iterations	Correct	Accuracy (%)
0	4799	87.56
1	4894	89.29
2	4909	89.56
3	4908	89.55
4	4908	89.55
5	4908	89.55
6	4909	89.56

Table 4.11: Contrast-plus accuracy at different iterations

For contrast-plus, we see from Table 4.11 that the second iteration shows a sudden increase in accuracy by 1.73 percentage points, and then a slight improvement in the third iteration. Thereafter the performance stays more or less constant. After the 6^{th} iteration, the accuracy figures remain unchanged.

No. of Iterations	Correct	Accuracy(%)
0	4321	78.84
1	4964	90.57
2	5024	91.66
3	5041	91.97
4	5046	92.06
5	5046	92.06
6	5046	92.06

Table 4.12: Contrast-pure accuracy at different iterations

For contrast-pure, there is a large initial increase in performance of 11.7 percentage points which takes the accuracy from a long way below contrast-plus to just above. Thereafter, the increase is more gradual with the accuracy reaching 92.06 in the fifth iteration.

No. of Iterations	Correct	Accuracy(%)
1	3506	63.97
2	4590	83.74
3	4983	90.91
4	5058	92.28
5	5072	92.54
6	5076	92.61
7	5076	92.61

Table 4.13: HITS accuracy at different iterations

Finally, for HITS the increase in performance is even more marked (almost 20 percentage points) than for contrast-pure in the second iteration. Keeping up the improvement in subsequent iterations HITS finishes at 92.61 in the sixth iteration, thereafter remaining unchanged.

Finally, Figure 4.2 and the associated Table 4.14 show the learning rates of the three scoring functions. It is interesting to see that contrast-plus starts out as the best base scoring method but shows the least improvement in the subsequent recursive learning process; whereas HITS starts off with the lowest accuracy, but improves to best performing procedure amongst the three with the highest increase in accuracy with an increase of 28.6 percentage points from base to convergence. The contrast-pure learning rates are is between the extremes ending up with a performance only slightly below HITS by about 0.55 percentage points.

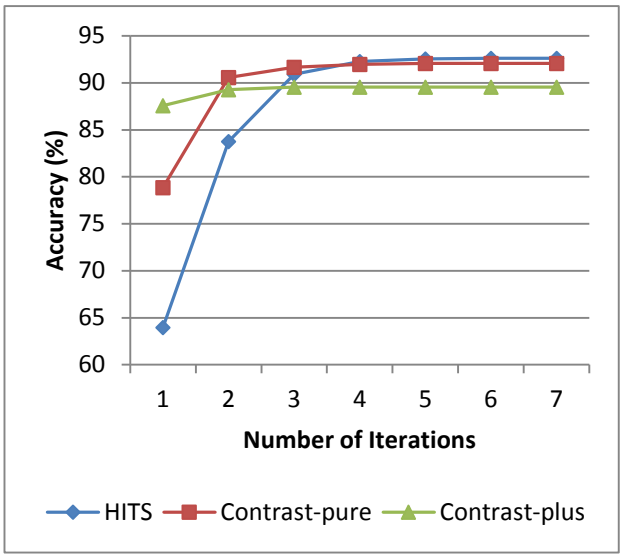


Figure 4.2: Comparison of the three methods showing accuracies at each iteration

No. of Iter-ation	Accuracy Increase (% points)		
	Contrast-Plus	**Contrast-Pure**	**HITS**
1	1.73	11.73	19.78
2	0.27	1.09	7.17
3	-0.02	0.31	1.37
4	0.0	0.09	0.26
5	0.0	0.0	0.07
6	0.018	0.0	0.0
0-7	**2.01**	**13.23**	**28.64**

Table 4.14: Comparison of the three methods with accuracy differences relative to the previous iteration

4.7.3 Summary of Evaluation Results

Table 5.1 summarizes the results, for the three scoring methods after the base scoring and recursive scoring. Overall, the morpheme learning techniques described in this chapter reach approximately 93% correct root morpheme identification.

	Base	**Recursive Scoring**
Contrast Plus	87.5	89.56
Contrast Pure	78.8	92.06
HITS	63.9	92.61

Table 4.15: Comparison of the three methods in terms of accuracy

4.8 Conclusion

This chapter investigated morphology learning using counts of root and pattern occurrences. A contrastive learning approach was presented in which a good root candidate is one that co-occurs with a large number or variety of pattern morphemes, and vice-versa. Within this approach two different strategies were considered: one considering own morpheme counts along with opposite morpheme occurrences were considered, known as contrast-plus; and the other where only opposite morpheme counts were considered, known as contrast-pure.

This contrastive learning approach was then developed further, with previous root scores being used to enhance pattern scores in each subsequent iteration, and pattern scores used to enhance root scores in a mutually recursive relationship until convergence is reached. At this point I introduced for comparison, the well-recognized HITS algorithm, used for ranking web-pages, and applied it to ranking morphemes. There is much similarity between the contrastive learning algorithm and the HITS algorithm. Intuitively, the recursive contrast-pure algorithm is essentially the same as HITS formulated slightly differently. This is verified by the evaluation results, as both methods give only slightly different results.

The three scoring methods were evaluated at different stages of development: at the base level, in the absence of recursive application, it was shown that the best performing method was contrast-plus with an accuracy of 87.5%; it takes into consideration the counts of both morphemes, its own counts and the co-occurring morphemes' counts. The other two methods, i.e. contrast-pure and HITS, gave poorer performance of 78.8% and 63.9%, respectively, at this point relying purely on co-occurring morphemes' counts. However, when the latter two methods were applied in a mutually recursive learning algorithm, they out-performed the former method by about 3 percentage points. Both the contrast-pure and HITS are almost at par due to recursive rescoring, with HITS giving slightly better performance (by 0.55 percentage points).

The contrastive learning approach is a simple yet powerful approach which is superior to the machine learning technique described in Chapter 3 which takes considerably more time in model training and application besides the added complications of

parameter estimation. In Chapter 3, the best performing method gave an accuracy of 86.26% compared to 92.61% for contrastive learning, an improvement of 6.4 percentage points.

Chapter 5

Contrastive Learning Extensions and Stemmer Comparison

5.1 Introduction

Chapter 4 demonstrated the superiority of contrastive learning over the machine learning based approach to non-concatenative morphology induction described in chapter 3 in terms of accuracy and computational efficiency.

This chapter investigates two extensions intended to enhance the contrastive learning technique devised in the previous chapter. The first extension is a refinement procedure which rescores the scores obtained from the base procedure. The intuition behind the rescoring procedure is that a potentially sound morpheme should be recognized if it keeps 'good company': it should receive a higher score if all the morphemes co-occurring with it have high scores. The procedure thus averages the scores of a morpheme over all of its co-occurring morphemes instead of taking the counts of the co-occurring morphemes as was the case in the base procedure.

The second extension is a root size normalization procedure. Shorter potential root morphemes are by their very nature very frequent. Since the contrastive learning technique is wholly dependent on morpheme counts it is important to normalize these counts across the different morpheme sizes. Up to this point the procedure has worked because the analysis has been restricted to consider only triliteral morphemes. After application of the normalization it would be possible to remove this restriction.

Finally, in order to gauge the merit of the unsupervised learning technique, I carry out a comparative evaluation against existing, widely used rule-based Arabic stemmers.

5.1.1 Chapter Organization

The chapter reports three related strands of work. Firstly, section 5.2 formulates a refinement procedure that is applied to the contrastive learning technique. The section discusses methods for rescoring hypothesised morphemes and associated initialization and stopping criteria, and goes on to present a set of experiments. Secondly, section 5.3 introduces a root normalization procedure, outlining two ways to normalize morpheme counts in order to extend morphological induction beyond triliteral roots; the section concludes with a further set of experiments. Thirdly, section 5.4 compares existing stemmers with the contrastive learning procedure incorporating the extensions described in the two previous sections. Section 5.5 concludes with a summary of the three strands of work, and proposes further avenues for investigation.

5.2 Contrastive Learning Refinement: Mean Rescoring

This section explores a refinement procedure that is applied to the recursively derived scores for each morpheme from equations (4.14) and (4.15) of Chapter 4. In this procedure, each morpheme is rescored by taking the average of the scores of all co-occurring morphemes. The idea is that a sound root and pattern should always co-occur with high scoring patterns and roots. If a morpheme co-occurs with a mixture of high and low scoring morphemes then its overall score would decrease, reflecting the fact that it is less reliable. In contrast, if a morpheme always has high scoring co-occurring morphemes it would get a higher overall score.

In this section, the previously described mutually recursive scoring procedure is referred to as the *base* procedure/algorithm/scoring, to which is applied the refinement step or rescoring. Also, consideration is restricted to the two cases, contrast-pure and HITS. Hence, contrast-pure is referred to simply as contrastive learning.

The rescoring procedure is recursive, using the seed score initialized from any of the previous scoring methods of the base procedure. The refinement rescoring functions, denoted by RS, are initialized using the converged scores $RS_0(\cdot) = S_N(\cdot)$ from the base procedure:

$$RS_k(r) = \sum_{p_{i=1}}^{|P_r|} RS_{k-1}(p) \tag{5.1}$$

$$RS_k(p) = \sum_{r_{i=1}}^{|R_p|} RS_{k-1}(r) \tag{5.2}$$

If $RS_k(r)$ is computed first then it uses the values, $k = 1, 3, 5$ ..., and the values $k = 2, 4, 6,$ ... for $RS_k(p)$. The values for k get switched if $RS_k(p)$ is computed first. The recursive iteration differs from the base procedure in that in each iteration only one of the two functions' scores gets computed rather than both. Unlike for contrastive learning and HITS, in this refinement step the rescored vectors do not converge as $k \to \infty$, as will be seen below; hence the stopping criterion at iteration, $k = K$ needs to be determined. Also, note that in the rescoring formula the score vectors from the previous iterations are not normalized as has been the case previously, since the rescoring here is based on computing the mean of scores, resulting in there being no chance of overflow or underflow at each iteration.

5.2.1 Initialization

Similarly to the initialization of the mutual recursion of the base scoring method (section 4.4), there are two choices for initialization: (i) initialize $RS_1(r)$ with the *pattern count oriented* score and $RS_1(p)$ with $RS_1(r)$; or (ii) initialize $RS_1(p)$ with the *pattern count oriented* score and $RS_1(r)$ with $RS_1(p)$. From step one, the *pattern count oriented* score is that which was chosen by the mutual recursive step based on pattern counts. Thus for HITS the pattern count oriented score is $S(p)$, hence my choice is type (i) initialization for refinement rescoring. For the contrastive case I use type (ii) where $S(r)$ is the pattern oriented score in step one.

5.2.2 Convergence

Transforming the rescoring functions into a linear algebraic representation, using $\vec{r}$ to denote the root score and $\vec{p}$ to denote the pattern score,

$$\vec{r}_k = A_r \, \vec{p}_{k-1} \tag{5.3}$$

$$\vec{p}_k = A_c \, \vec{r}_{k-1} \tag{5.4}$$

where $A_r = A \; diag({\phi^{-1}}_A)$ and $A_c = A \; diag({\phi^{-1}}_{A^T})$, which are the adjacency matrix divided by the sum of rows of A (i.e. $1/|P_{r_i}|$) and the sum of columns of A (i.e. $1/|R_{p_j}|$), respectively. Combining the two equations (5.3) and (5.4) results in:

$$\vec{r}_k = \left(A_r {A_c}^T\right)\vec{r}_{k-2} \tag{5.5}$$

$$\vec{p}_k = ({A_c}^T A_r)\vec{p}_{k-2} \tag{5.6}$$

As shown in Chapter 4, it is possible to compute the dominant eigenvalue for a matrix using the power law to prove convergence. For the algorithm to converge, the product of the adjacency matrix product with its transpose has to be either symmetric or diagonalizable. Unfortunately, neither $A_r {A_c}^T$ nor ${A_c}^T A_r$ are symmetric. For a matrix $A_r {A_c}^T$ or ${A_c}^T A_r$ to be diagonalizable, the graph represented by A must be fully connected. This is also not true since it is not likely that all morphemes would be interconnected. Therefore, an alternative solution must be sought for stopping the iterations.

5.2.3 Stopping Criterion

An important thing to note about the recursive rescoring functions is that a root is assigned a score averaged over some patterns; likewise a pattern is assigned a mean score over a certain set of roots. The dimensions of the two morpheme score vectors are different. Since there is a single initial seed score based on the reliable pattern

occurrence counts (as described in section 5.2.1) the initial scores for patterns and roots computed using the rescoring functions have different dimensions indicated by the size or norm of the score vectors. As the algorithm iterates, there comes a point, $k = K$, when the size or norm of the root and pattern vectors are nearly equal, hence the difference, δ_k between them is minimized,

$$\delta_k = 1 - \frac{\|RS_k(p)\|}{\|RS_k(r)\|}$$
$$K = \min_k \ (\delta_k) \tag{5.7}$$

Thereafter, the difference between the vector sizes starts to increase again. This is illustrated in Figure 5.1, for an example using rescoring on contrastive learning. So, in this case, at $K = 4$ the iterations are stopped and the refined root and pattern vectors are output.

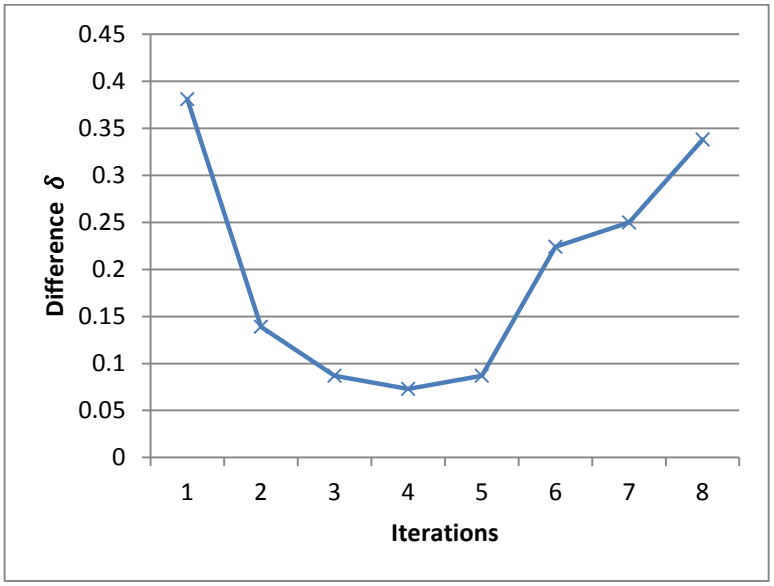


Figure 5.1: The size difference between the root and the pattern vectors

5.2.4 Refinement Scoring Experiments

The refinement procedure is also a recursive procedure where it is crucial to identify the iteration to stop at. Hence while evaluating the contrastive and HITS methods using the same accuracy measure we will also look at how to identify a stopping criterion.

As stated in section 5.2.3, while iterating through the refinement procedure, there comes a point when the difference between the size of the root vector and pattern vector is minimized. This is the point when the two vectors are comparable and accuracy is expected to be maximized.

For contrastive learning, in Figure 5.2, the plot for the vector difference between root and pattern score vectors along with accuracy is plotted on the same graph at each iteration. The two plots indicates clearly that there exists an inverse relationship between the vector difference, δ and accuracy. Hence, at iteration 4, when the value of δ is minimized the accuracy of the lexicons is maximized.

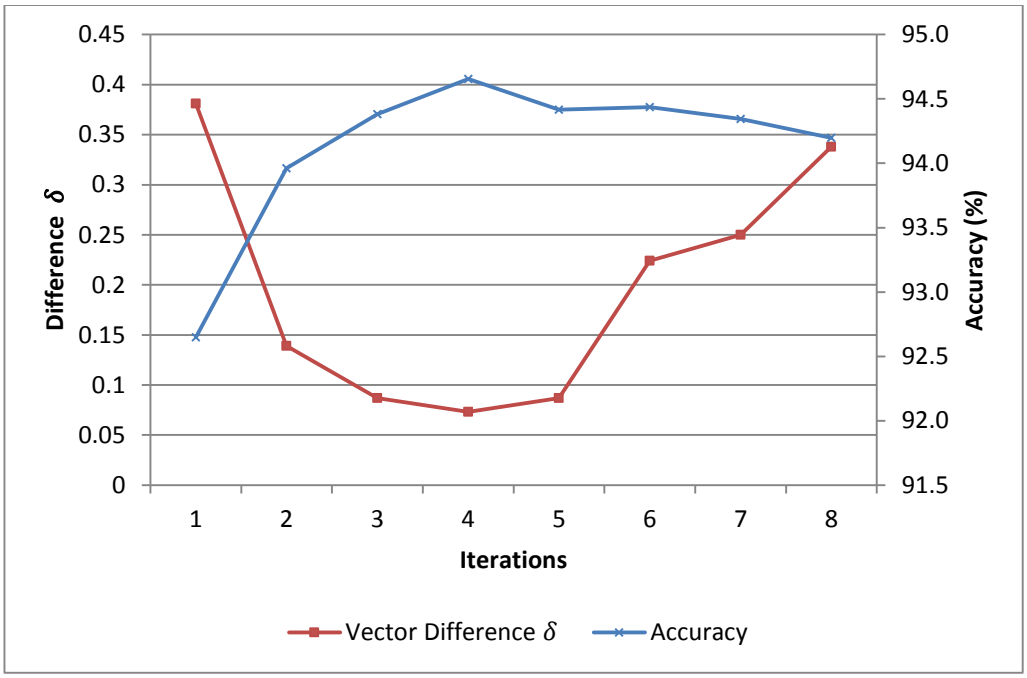


Figure 5.2: The size differences between root and pattern vector alongside the accuracy, for contrastive learning

Almost exactly the same trend is seen for the HITS refinement procedure. Figure 5.3 shows the two plots for vector difference and accuracy, following the same behaviour with only vector difference values scaled differently. This behaviour further corroborates the hypothesis that the contrastive learning and HITS procedure are the same in terms of ranking morphemes.

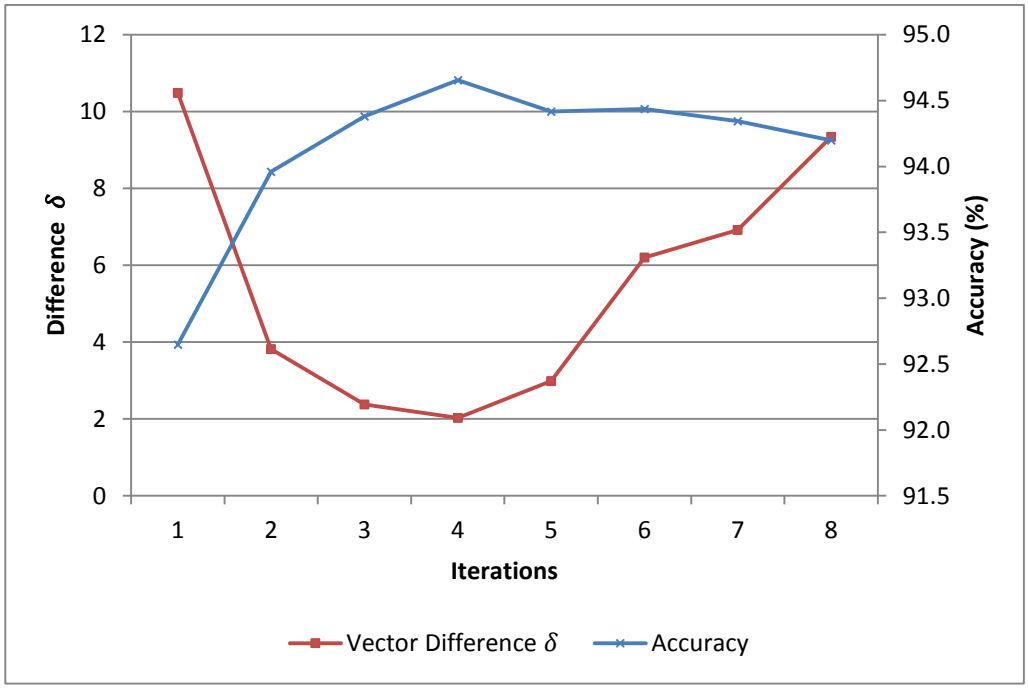


Figure 5.3: The size differences between root and pattern vector alongside the accuracy, for HITS

5.2.5 Summary of Evaluation Results

Table 5.1 summarizes the results for the three scoring methods after the base scoring, recursive scoring and refinement steps. Overall, the morpheme learning technique extended with the refinement step identifies almost 95% root morphemes correctly.

	Base	Recursive Scoring	Refinement
Contrast-Plus	87.5	89.56	---
Contrast-Pure	78.8	92.06	94.7
HITS	63.9	92.61	94.7

Table 5.1: Comparison of the three methods in terms of accuracy

5.3 Root Normalization: Unrestricted Morpheme Size

Up to this point the techniques and experiments have been restricted to triliteral root and pattern morphology induction. In the Arabic language, most words are derived from triliteral roots but there are a few four letter roots and a very few five letter roots. In the QAC, 98.84 % of the derivable words are from three letter roots; the remaining ones (1.16%) are four letter roots and there are no 5-letter roots. Therefore it is not inappropriate to assume only triliteral roots for analysis. However, a goal of this work is to avoid pre-defined parameter settings, since such artificial constraints subvert the objective of unsupervised learning. Thus, the algorithm should be able to freely discover the correct root from any length of substring morphemes, from a single character root upwards.

Root normalization is the procedure which is applied to remove the restriction of only three letter roots. The procedure balances the higher weights of shorter morphemes with the lower weights of longer morphemes. There are two types of normalizations that are carried out in order to balance the raw counts, namely, *root weighting* and *variety counts*; in addition, a combination of both types is referred to as *weighted variety counts*. This work also proposes a method to decide whether or not to apply a candidate pattern to a split a word into a root and pattern or to leave it unanalysed.

5.3.1 Root Weighting

So far, the techniques have used raw morpheme counts. However, by their very nature, shorter potential morphemes occur more frequently than longer ones. For example, if a language has three characters in its alphabet, $\sum = \{a, b, c\}$, then there are four possible strings containing each of the single character substrings, e.g. *a* is contained in *a, ab, ac,* and *abc*; similarly, each two character substring is found in two strings of maximum length three, e.g. *ab* is found in *ab* and *abc*. In general, a substring of size x would occur in 2^{x-y} strings of maximum length y. Using this count, it is possible to normalize the weight, δ_r^w, of each root, r, according to its size in each word, w, applying the formula:

$$\delta_r^w = \frac{1}{(2^{\,|w|-|r|})} \tag{5.8}$$

Table 5.2 shows an example of the three possible sizes of roots in a four character word, along with the corresponding weights, δ_r, in the word.

Root, r	**Word, w**	**Weight, δ_r^w**
a	abcd	$1/2^{4-1} = 0.125$
ab	abcd	$1/2^{4-2} = 0.25$
abc	abcd	$1/2^{4-3} = 0.5$

Table 5.2: Weighted counts of a root relative to its size in a word

Thus, the raw counts of $|P_{r_i}|$ in the summation formula of the pattern scoring procedure, are replaced with the aggregated weights, WT, for each $\delta_{r_i}^w$ of each r_i in w such that $r_i \lhd w$:

$$WT(r_i) = \sum_{r_{i=1}}^{|P_r|} \delta_{r_i}^w \tag{5.9}$$

5.3.2 Root Variety Counts

Another type of normalization of the root counts uses *variety counts* instead of raw counts. Variety counts are the counts of a root morpheme which do not include the counts of any morphemes for which the former is a substring of the latter. For example, given the three strings *nktb*, *tktb* and *ktbA*, the variety count of *ktb* is 3, as this root substring occurs with three different characters. Although the raw count of *kt* is also 3, since it is a substring of *ktb* whose count is 3, the variety count of *kt* is 1.

The procedure for adjusting the counts is as follows. Roots r_i in R_p are traversed in descending order of root size. Let VC_{r_i} denote the variety count of r_i. Starting with longest strings which are not substrings of any other root string, the value of $VC_{r_i} = |P_{r_i}|$, the raw count is assigned. For each root, r_i that is traversed, the counts for all of its substrings, r_j having size $|r_i|$-1, are adjusted by subtracting the counts, $VC_{r_j} = \left|P_{r_j}\right| - VC_{r_i} + 1$ if r_j has not been visited, or $VC_{r_j} = VC_{r_j} - VC_{r_i} + 1$ if it has been visited. Also, a wordlist is maintained for each r_j containing words, w_k such that if another root r_i belonging to R_{w_k} is traversed, VC_{r_j} is not updated; this is in order to prevent double counting in the case of encountering substrings of substrings. Hence, in this way the counts of shorter morphemes which tend to occur as substrings of other morphemes are reduced.

5.3.3 Weighted Variety Counts

A third type of normalization is based on the variety counts, but instead of using the raw counts, the weighted counts from section 5.3.1 are used. In the example given above, for the three strings, *nktb*, *tktb* and *ktbA*, the weighted variety count of *ktb* is 1.5 (0.5+0.5+0.5) instead of the raw count 3; and the weighted variety count of *kt* is 0.25 (0.25+0.25+0.25/3), the average weight of *kt* in the three strings, instead of the raw count 1.

The formulae above are adjusted, such that for the root r_i in descending order, $VC_{r_i} = WT(r_i)$ and for the substrings r_j,

$$VC_{r_j} = WT(r_j) - VC_{r_i} + AWT(r_i) \tag{5.10}$$

and

$$VC_{r_j} = VC_{r_j} - VC_{r_i} + AWT(r_i) \tag{5.11}$$

where

$$AWT(r_i) = \frac{1}{|P_r|}\sum_{r_{i=1}}^{|P_r|} \delta_{r_i}^{w} \tag{5.12}$$

5.3.4 Extended Analysis

Since morphology learning has previously been restricted to triliteral roots, the adjacency graph includes roots of size $|r| = 3$ and patterns with $f_p(\div) = 3$ from w where $|w| > 3$. For example, the decomposition of word *yErf* is

$$D(yErf) = \begin{Bmatrix} \langle y\ E\ r, & - - - f\ \rangle, \\ \langle y\ E\ f, & - - r - \rangle, \\ \langle y\ r\ f, & - E - - \rangle, \\ \langle E\ r\ f, & y - - - \rangle \end{Bmatrix}$$

Further, when analysing a word with $|w| > 3$, it has to undergo a compulsory analysis into root and pattern outputting as $|r| = 3$, the only root size available in set R_w. Words where $|w| \leq 3$ are thus output as whole, unanalysed by the procedure. The accuracy figures quoted so far thus include the correctly identified three letter roots as well as the unanalysed three letter words.

Relaxing the restriction would allow any root size, extending the $\langle r, p \rangle$ pairs in the above example to also include pairs such as $\{\langle y,\ -Erf \rangle, \langle yE,\ --rf \rangle, \ldots\}$. The unrestricted set R_w of all possible substrings r_i of w would analyse any word of size

$|w| > 1$. However due to the restriction $|r| < |w|$, i.e. excluding $r = w$ in R_w, every word would undergo compulsory analysis to a root of maximum size $|w| - 1$. Thus, for example a word *ktb* would be forced down to perhaps *kt* or *kb* without allowing the output *ktb* which is the correct root form.

In order to remove this limitation, allowing words to be output unanalysed, I now extend R_w to include r such that $r = w$. R_w thus includes all possible substrings of w including w. When $r = w$, the corresponding pattern p has $f_p(\div) = |w|$, i.e. all blanks with no affix characters. These patterns will be referred to as *null* patterns, $\bar{p}$. For the example word *yErf*, the pair $\langle yErf,\ ----\rangle$ is now included as a potential analysis. Thus the analysis output has been extended to include whole words instead of compulsory root and pattern constituents.

The raw count of null patterns, $|R_{\bar{p}}|$ is the count of the number of words having $f_{\bar{p}}(\div) = |w|$, which is a very large number in comparison to other pattern counts. This would force the analysis to output whole words only, without possible analysis into root and pattern constituents. An adjustment of the count $|R_{\bar{p}}|$ to a suitable lower value is required in order to induce analysis. I chose to heuristically set the value of $|R_{\bar{p}}|$ to the largest count of a non-null pattern, p_i with $f_{\bar{p}}(\div) = f_{p_i}(\div)$. Hence, for example if the largest pattern count occurs for a pattern '*y*---' is $\left|R_{p=y---}\right| = 150$, then $\left|R_{p=---}\right| = 150$ instead of the count of the number of three letter words, which could be in the thousands.

5.3.5 Experimental Results for Root Normalization

This section presents an evaluation of the various root normalizations, removing the restriction of only considering 3-letter root and pattern template, along with extended analysis to allow whole (unanalysed) words to be output. Previously, the number of evaluated words with triliteral roots was 5481. This increases to 5545 forming the entire set of derivable words for evaluation, with the inclusion of the 64 four-letter root words in the QAC.

Starting with the contrastive learning *base* algorithm (from Chapter 4), I first compare weighted root normalization (CL_WR), followed by variety count (CL_VC) and weight variety count (CL_WVC), finally concluding with the extended analysis (CL_EA_WVC). The comparison of the different configurations is shown in Table 5.3.

Stemmer	**Correct (out of 5545)**	**Percentage Accuracy**
Base	0	0
CL_WR	4268	76.97
CL_VC	4273	77.06
CL_WVC	4297	77.49
CL_EA_WVC	5056(20)	91.18

Table 5.3: Comparison of different root count normalization and extended analysis

As expected, the base algorithm is unable to perform in the absence of the 3-letter root restriction, producing only single character roots as the output for all the words. This is because short root substrings have very high counts, in the order of thousands, whereas triliteral root counts would be in the range zero to 10. By applying root weighting, (CL_WV) an appropriate analysis is output with a considerable number of roots correctly identified. This output, with an accuracy of 76.97%, shows the importance of assigning the correct count weight to each root substring relative to the word size in which the substring occurs.

Next, the variety count normalization (CL_VC) is applied. This normalization scheme shows slightly better performance than root weighting (CL_WR). The variety count not only reduces the count of shorter morpheme substrings relative to the size of a word but also assigns a standalone count independent of the counts of its superstrings. CL_WVC shows further improvement, indicating the advantage of the combined formulation.

Extending the analysis to allow whole words to be output permits words such as *ktb* which have been previously analysed as *kt* to be correctly analysed as the root *ktb*. The

output shows more than 760 three character words have been correctly analysed which were previously constrained to be analysed into the root substring of the word. Moreover, 20 four-letter root words have been correctly analysed which demonstrates that the extensions work beyond triliteral roots. There is only 31.25% accuracy for four letter roots but usually quadraliteral roots are hard to recognize. Since the dataset consists of predominantly three-lettered roots (98.4%), the number of pattern counts associated with 3-letter roots, r_i^3 is far greater than the number of pattern counts for four letter roots, r_i^4: $\left|P_{r_i^3}\right| \gg \left|P_{r_i^4}\right|$. This, therefore, undermines the importance (weight) of the associated four letter substrings.

Thus the performance after applying these extensions is comparable to the performance of the triliteral root restricted system of Chapter 4.

5.4 **Stemmer Comparison**

With the developments to the contrastive learning procedure for unsupervised learning of morphology, I compare the technique to manually built tools for root extraction known as 'stemmers'[8].

There are several stemmers that have been implemented for Arabic which are able to analyse an Arabic word and output its root. Most of these stemmers are rule-based, defining manually written procedures for removing affixes from words relying on hand encoded lists of patterns and affixes and even lists of roots. Some prominent stemmers are described by Khoja & Garside (1999), Al-Shalabi (2005), Taghva *et al*(2005), Ghwanmeh *et al* (2005), and Sonbol *et al* (2008). All of these were produced by labour intensive methods and are limited to the encoded list of patterns and affixes.

I compare my unsupervised learning technique evaluated on accuracy of root extraction to two existing rule based stemmers: (i) the Khoja Stemmer (Khoja & Garside, 1999)

[8] Although generally referred to the process of obtaining stems by removal of prefixes and suffixes, 'stemming' is sometimes referred to mean the process of removing any affixes from words, and reducing these words to their roots.

and the (ii) ISRI Stemmer (Taghva *et al*, 2005). These systems are representative of this class of Arabic stemmers.

5.4.1 Khoja Stemmer

This stemmer makes use of a dictionary of roots and patterns to produce the desired root of a word. The stemmer first removes the longest suffix and the longest prefix using a list of affixes. What remains is then matched with the verbal and noun patterns, thus extracting the potential root. This root is looked up in a root dictionary. If found in the dictionary, the correct analysis is output. In more detail, the procedure for stemming is outlined below:

1. Text is normalized, removing punctuation, numbers and any traces of diacritics representing short vowels.
2. Stop words are removed.
3. Clitics such as the definite article ال (*Al*) and conjunctions و (*w*) are removed
4. Longest suffixes are removed.
5. Longest prefixes are removed.
6. The resulting stem is matched against a list of pattern templates to extract the root.
7. The root is validated against a dictionary of correct roots. If it is not found, the stem is output.
8. All weak radicals or long vowels ا(*A*), و(*w*) and ي(*y*) are conflated to the single vowel و(*w*).
9. All occurrences of humza letters (ؤ, ئ, ء,أ) are conflated to a single letter أ(<)
10. If the root resulting from step 7 has only two characters then the root dictionary is checked to see if there are reduplicated letters, which are then added to the root.

The conflation of weak radical to a vowel letter (step 8) is a debatable attempt to implement the *weak root radical rule* (section 1.3.1.3). This is an oversimplification, making the analysis of weak radical roots correct for only two possible radicals of the

root. It also gives rise to errors in the analysis of non-weak roots, as pointed out by Taghva *et al* (2005), where for example the word *mnZmAt* receives the incorrect analysis *Zm<* instead of the root *nZm*.

5.4.2 Information Science Research Institute (ISRI) Stemmer

The stemmer developed by Taghva *et al* (2005) at the Information Science Research Institute (ISRI) is an attempt to overcome the Khoja stemmer's dependence on a root list. It is currently part of the Natural Language Tool Kit (NLTK)[9] as the stemmer for Arabic. It builds on and improves the rules implemented by the Khoja stemmer without relying on the root dictionary.

Affixes are assigned to classes. Prefixes and suffixes are classified according to their character lengths. Pattern templates are classified in terms of length of the pattern and the length of the root. Also the order of application of each affix within each class is fixed. Thereafter, the procedure, similarly to the Khoja stemmer is applied with a specific order of application of the different classes of prefixes, suffixes and pattern templates. The procedure is outlined below:

1. Normalization: diacritic removal; Humzated letter conflation.
2. Remove longest prefix (length three then two)
3. Remove the conjunction *w*(و) in the case of two consecutive *w* (و و).
4. Conflate all marked alif (أ, إ) to the unmarked alif (ا) or *A.*
5. If the resulting stem is three characters long, output the stem as the root.
6. Length 4 stem: match pattern template to extract three letter root; if there is no match them check suffix and prefix in order. Output triliteral root if there is a match; else output the noralized word.
7. Length 5 stem: first apply step 6 except; if still a 5 character stem remains then apply pattern template with 4 letter roots. Output root if a match is found.
8. Length 6 stem: match appropriate 3-root pattern. If no match remove 1-character suffix or prefix if matched. If removed repeat step 7.

[9] http://www.nltk.org/

9. Match 1-character suffix or prefix. If found repeat step 8.

In summary the algorithm applies certain rules to remove prefixes and suffixes in a particular order, matching the stripped word against a set of patterns for either three or four letter roots; if a pattern is found, the root is then output. The tool minimizes the set of patterns used thus putting less dependence on pre-encoded lists. Taghva considers this to be the main difference between this algorithm and other Arabic root finding algorithms.

5.4.3 Shortcomings of Existing Stemmers

Most Arabic stemmers suffer from the 'affix ambiguity problem' (Al-Shawakfa *et al*, 2010). This ambiguity arises due to a failure to distinguish the suffix and prefix from the pattern affix in a word, leading to application of the incorrect rule. For example, a stemmer might incorrectly stem the word *mskwn* to *msk*, stripping away the suffix *wn*, instead of applying the *mfEwl* pattern to retrieve the correct root, *skn*. The ambiguity also extends to choices for prefix and suffix morphemes which may have the same form as peripheral pattern characters.

The main reason for this failure is the hard-coded order of removal of affixes. Stemmers differ in their sequence of rule applications but these are mostly based on linguistic judgement which caters for the majority of cases in a development dataset but may fail in other kinds of text. Besides, there is the additional overhead of maintenance of the rules to adjust to different language variants.

Dictionary maintenance and updating is a resource intensive requirement for stemmers having modifiable lists of roots and patterns. Though most authors do away with the root dictionary, as in the ISRI stemmer, defining the patterns to be incorporated and the order of application of these patterns is non-trivial. Taghva *et al* (2005) use 44 patterns, while Ghwanmeh *et al* (2005) use up to 80 patterns. This can be compared with a total of 185 (undiacritized) distinct pattern types (see Appendix A) derived from Attia (2011). According to Al-Kabi *et al* (2011), the Ghwanmeh *et al* (2005) stemmer overall performs better than ISRI but on certain occasions ISRI performs better at finding the

correct roots. This is due to the arbitrary choice of pattern order in the stemmers.

5.4.4 Experiments

This section empirically compares the Khoja and ISRI stemmers to the contrastive learning technique in terms of root extraction accuracy. The same normalization procedure is used in both stemmers but this differs slightly from the transliteration and normalization defined in Appendix C which is used in the previous evaluations in this thesis. Hence the vocabulary of conjugated words reduces to 5366 from 5429 due to a smaller alphabet resulting in more conflated character classes. For example, previously the alif madda letter (آ) was transliterated to two consecutive characters ` and *A*; whereas now it must be classed with other Humzated alif letters (أ and إ) which are conflated to the single ا (*A*).

As stated above, conflation of the weak radical to a single letter is an inadequate attempt to implement the weak root radical rule. Nonetheless for the sake of fair comparison with the Khoja stemmer, I consider two types of evaluations: one where the weak root letters are not conflated and one where they are conflated. The comparison showing the former evaluation type is shown in Table 5.4.

Stemmer	**Correct (out of 5429)** (total)(quadraliteral)	**Percentage Accuracy**
Khoja	4431(33)	81.62
ISRI	4504(26)	82.96
Contrastive Learning	4899(0)	90.24

Table 5.4: Accuracy comparison of the Khoja and ISRI stemmers with contrastive learning without weak radical conflation. The number of correct quadraliteral root words shown in brackets.

As expected, the accuracy of the Khoja stemmer is lower than the other two methods. ISRI outperforms Khoja despite not using a root dictionary, but this superiority is only

due to the oversimplification of root radical conflation which renders many of the weak radical roots invalid for Khoja. The comparison shows the significant superiority of the unsupervised learning technique, with an improvement of 7.28 percentage points over the manually built stemmers.

One surprising outcome is that with a slightly different normalization procedure applied to the same dataset, the contrastive learning technique fails to recognize any four letter roots. This may be because simplifying the words conflates important pattern templates thus giving unclear indication of roots. This aspect needs further investigation to know what conflation classes cause the algorithm to perform less well.

The final evaluation conflates the weak root radicals to accommodate for the Khoja stemmer. The results are shown in Table 5.5. In this evaluation, the Khoja stemmer outperforms the ISRI stemmer by 3.33 percentage points. Yet again, as seen in the table, the accuracy of unsupervised contrastive learning without the refinement step is better still, by 3.1 percentage points over the manual approach. With refinement, the accuracy increases giving an overall improvement of 5.07 percentage points over the Khoja stemmer.

Stemmer	**Correct (out of 5429)** (total)(quadraliteral)	**Percentage Accuracy**
Khoja	4775(33)	87.95
ISRI	4594(26)	84.62
Contrastive Learning (basic)	4943(0)	91.05
Refined	5050(0)	93.02

Table 5.5: Accuracy comparison of the Khoja and ISRI stemmers with contrastive learning and refined contrastive learning using weak radical conflation. Number of correct Quadraliteral again shown brackets.

5.4.5 Discussion

These comparisons with existing, widely-used Arabic stemmers confirm the worth of the unsupervised learning technique developed in this thesis. In comparison to the stemmers, the contrastive learning technique bases the analysis of a word not just on the affix pattern but also the strength of the resulting root. Thus, character sequences weighted highly as a consequence of a strong co-occurring affix pattern would be likely root candidates, whereas no such gauge for potential roots is available in rule-based stemmers. Also, the technique automatically learns a meaningful ordering of morphemes in the lexicon, thus eliminating the manual task of setting the right order affix/pattern application in the stemmers. In addition, there is no arbitrary choice of affixes and patterns hand coded into the system which determines the scope of application of the stemmers; instead the unsupervised learning techniques discover patterns based on their occurrence in the corpus. Even if there is a rare pattern in the corpus, it may be correctly recognized as a consequence of the weights assigned to potential roots that occur with other patterns. But it would not be analysed by the stemmers if it were omitted from their encoded pattern lists.

These advantages are reflected in the evaluation results. The affix ambiguity problem is reduced due to the learned ordering. For example, for the Khoja stemmer, the word *mrjAn* is analysed to *rjn*, applying the pattern *m - - A -* instead of removing the suffix *An;* the ambiguity between the affixes prevents it yielding the correct root *mrj*. But in the contrastive learning algorithm this ambiguity is removed; the roots *rjn* and *mrj* with weights 1.40743e-05 and 8.64987e-05 and patterns *m - - A -* and *- - - An* with weights 0.000193684 and 0.000103935 lead to the correct analysis ⟨*mrj , - - - An* ⟩ with aggregate weight 1.67534e-08, compared to the incorrect analysis ⟨*rjn , m - - A -* ⟩ with lower weight, 1.46281e-09. The algorithm's assignment of weights to the root and pattern reflects their significance in the corpus, whereas no such knowledge is available to the dictionary based stemmers.

5.5 Conclusion

This chapter concludes the description and evaluation of an unsupervised approach to learning the intercalated morphology of Arabic. Starting off with the basic link analysis based algorithm, a refinement method is first formulated to further improve the results. This iterative rescoring procedure helped increase performance without compromising unsupervised learning. Next, moving beyond triliteral roots, morpheme count normalization methods were introduced to allow recognition of roots without restricting morpheme size. This allowed the correct analysis to be chosen for any of root size starting from single character substrings. Finally a comparative evaluation with existing, widely used stemmers establishes the true significance of the unsupervised contrastive learning technique which outperforms the rule-based tools in terms of root identification accuracy.

The work reported in this thesis has explored several issues but there are further areas that warrant investigation. The refinement procedure lacks a theoretical foundation and could be studied further in term of graph connectivity. It might be possible to identify groupings of morphemes that are connected in the graph to reveal paradigm-like associations between root and pattern morphemes.

The normalization procedure looked into root variety count normalization but no investigation into pattern variety counts was carried out. This may further enhance the ability to capture the right analysis given balanced pattern counts. Also, while extending morphological analysis to whole words, a simple approach was used to induce the analysis of words, where null patterns were assigned weights of the highest non-null patterns (section 5.3.4). Other approaches could be investigated for this sub-task.

Although the current approach does give some idea about when to and when not to analyse words, a more in-depth investigation might be worthwhile investigating. There could be an automatic procedure to determine a threshold such that words with scores below a certain threshold value are left unanalysed. Other approaches include making use of the Minimum Description Length (MDL) principle to decide whether a particular analysis would reduce the total size of representation of the morphemes.

The unsupervised learning techniques proposed in this thesis were designed to be parameter free and independent of language specific choices. Yet, the techniques perform comparably or better than existing manual tools. In principle, it would be possible to use the same techniques across different datasets and languages without the need to understand the structure of the language.

Chapter 6

Conclusions

6.1 Introduction

The aim of the research reported in this thesis was to develop and investigate techniques for learning the non-concatenative morphology of Arabic in an unsupervised manner. Making the techniques unsupervised means that the developer does not need to know the linguistic structure and morphological rules of the language. Thus, no manual labour is required for coding such rules; nor is there a need to obtain annotated datasets for training supervised learning models for learning word structure. To be used in practical applications, an unsupervised learning approach must give performance comparable to systems based on manually developed rules or supervised learning. While effective unsupervised learning methods have been built for concatenative morphology it has not previously been shown whether effective systems can be built for non-concatenative morphology.

The research questions stated at the beginning of the thesis were:

> *Can the non-concatenative morphology of Arabic be learnt effectively with performance reasonably close to that of linguistic resources and tools? To what extent can the devised approach be independent of manual settings and language specific parameters?*

In order to answer these questions, two techniques were devised to learn lexicons of roots and patterns. The first technique was inspired by an existing one based on Maximum Entropy modelling adapted for unsupervised learning, which was originally used to identify affixes sequentially appended to a stem (concatenative morphology). The second technique learns the lexicons using a simpler yet more efficient approach based on mutually recursive count updates of co-occurring root and pattern morphemes. The more effective of the two methods was the latter, the contrastive learning approach. This was then extended, resulting in a robust procedure for producing a ranked list of

root and pattern morphemes, which are then used to produce an analysis of a word into its root and pattern morphemes. Careful steps were taken to keep the techniques free of any parameter settings or language specific information. The final extended technique was compared with two rule-based Arabic stemmers, through an evaluation on data from the Quranic Arabic Corpus. The unsupervised learning approach gave comparable or better performance than the manually built tools, in terms of accuracy of root identification.

This chapter highlights the strengths and novelties of the implemented unsupervised learning techniques, as reflected in the empirical findings. It also highlights limitations that are difficult to address in unsupervised approaches. Since this research is a preliminary attempt to address the problem of non-concatenative morphology, there are several aspects that deserve further explanation in order to determine the full potential of such unsupervised methods.

6.1.1 Chapter Organization

Section 6.2 discusses the contributions of the research. Limitations are discussed in section 6.3, and aspects that are missing pointed out in section 6.4. Based on these shortcomings, section 6.5 outlines possible future work. Section 6.6 concludes.

6.2 The Strengths and Contributions

This research addresses the unsupervised learning of non-concatenative morphology for naturally written undiacritized Arabic text, in which a word is broken down into *all* possible root and pattern combinations. For a word of length *n*, the number of combinations of root and corresponding pattern pairs amounts to $2^n - 1$, an exponential number. This model could only be feasibly applied to stemmed, undiacritized words which are relatively short. Experiments to learn the morphology of vowelled text by Rodrigues & Cavar (2007) and Xanthos (2007) have had to use heuristics to reduce the search space of possible analysis before applying their respective unsupervised learning techniques. For instance, Xanthos uses Sukhotin's vowel identification algorithm to first

narrow down potential possible patterns before applying an MDL based unsupervised learning technique. However, this initial pruning of the possible analysis set is likely to result in sub-optimal morphology learning accuracy.

The contrastive learning approach, developed in Chapter 4 and extended in Chapter 5, is a novel method of (unsupervised) learning of the morphology of Arabic which is designed to simultaneously learn roots and patterns, exploiting their mutual inter-dependence. It employs a mutually recursive procedure which gives an optimized morpheme weighting. This strategy is in contrast to most contemporary approaches to unsupervised morphology learning which are based on data compression techniques. Such compression or MDL-inspired approaches have been criticized for their weak theoretical and practical basis (Hammarström, 2007): intuitively, there is little link between data compression and what linguists conceive as morpheme units; experimentally, such compression based methods usually depend on thresholds and supervised parameters.

Approaches that are free of language specific parameters and thresholds are more applicable to other languages in the Semitic group and also to other Arabic dialects, which all exhibit similar root and pattern structure. This was a central concern in this work in order to make it suitable for other similar applications. In particular, some other unsupervised learning techniques make use of inter-radical distance thresholds and are limited to learning triliteral roots (e.g. Elghamry, 2004; Rodrigues & Cavar, 2005).

Most previous work aims to learn affixes by using only information from affix or other substring counts, but not whole lemma (stem/root) counts. The approach pursued in this thesis is distinctive in that it learns affixes and lemmas simultaneously, making use of information from both types of morphemes. In this way, the analysis procedure is strengthened by not just good affix candidates but also plausible lemma candidates in a word.

The approach produces a ‘natural’ ordering of morphemes in the lexicons according to their prevalence in the vocabulary dataset. This is one of the strengths of the algorithm over manually-based approaches. Manually built tools for Arabic morphological analysis usually rely on a list of affixes and an arbitrary ordering of these affixes to be

applied when there is ambiguity in the analysis. This ordering relies on linguistic judgement and may not reflect the actual occurrence of morphemes in the text being analysed. The learning method offers morpheme ranking that reflects the actual morpheme usage in the text.

Most systems for Arabic root identification are limited to identifying triliteral roots and do not provide a means to go beyond to quadraliteral or other sized roots. In this work, an attempt to recognize any size roots is presented which has been successful to some extent in capturing roots of any size. With some further refinements to the system it may be possible to improve the performance on other sized roots beyond triliteral.

In the maximum entropy based learning approach, as well as adapting it to non-concatenative morphology, I worked on correcting the formula for obtaining the log based morpheme score from that originally proposed. Previously, the calculation was contrary to intuition, giving higher weight to unrelated words. I modified the log scoring formula to invert the incorrect scoring trend while also introducing a measure to give emphasis to the length of related words.

6.3 **Limitations**

As discussed earlier, Arabic word formation includes certain morphophonemic adjustments, such as the weak root radical rule, i.e. changing long vowels in the root to a different long vowel in the actual words. Similarly, at times the long vowel is completely dropped in the final word, leaving only two root radicals from a tri-literal root. Also, reduplicated root radicals are represented with a *shaddah* marker (ّ) which is omitted in undiacritized texts, again resulting in only two radicals in the final word. The unsupervised learning technique is incapable of identifying such roots correctly as a whole, although partial identification may be possible. Thus if two out of the three root radicals are identified correctly the analysis could be classed as a correct, but not the complete solution. To map back exactly to the modified radical would be very hard to accomplish with unsupervised learning.

Unlike manually-based systems, which can be applied to single words at a time in order to obtain a satisfactory result, unsupervised techniques are dependent on having a sizable corpus to extract a lexicons. Hence, methods for unsupervised learning are sensitive to corpus content and size. This sensitivity is somewhat visible in the different performance for the two different normalizations of the QAC (section 5.4). The input needs to contain a large number of morphologically inflected words; uninflected words, such as proper nouns, are like noise for the learning algorithm.

Another drawback of unsupervised systems is computational cost. Although there are no dictionary storage requirements as for manual tools, the procedure for induction requires processor and memory resources. Processing times may be significantly longer than manually-based systems which may only perform string search operations. For the machine learning based approach, with the QAC vocabulary it can take up to a few hours to build a model and apply the model back to obtain nearest neighbour clusters. On the other hand, the contrastive learning approach requires only a couple of minutes to extract the lexicons, depending on the number of iterations performed. Long processing time is the result of the large search space of possible analyses which are exponential in the length of each word. Fortunately, for stemmed, undiacritized Arabic words this search space is considerably reduced. Memory requirements are likewise high when it is necessary to store a list of all possible root and pattern combinations for all vocabulary words.

6.4 **Omissions**

An aspect that was not investigated is the process of deciding when to analyse or to leave a word unanalysed. It was only briefly touched on from one angle when considering the morpheme count of the *null pattern* (section 5.3), where words were left unanalysed if they occurred more frequently as a whole than as a substring. However, this is not a principled solution.

The lexicons derived are ranked lists of all possible morphemes, with the most promising ones appearing at the top. There is a need to automatically determine a cut-off point for the roots and patterns in order to filter out morphemes that are not in the

language. This would reduce the computational cost at each iteration and also provide a means of deciding which words should be morphologically analysed.

With regards to formal evaluation, most morphological analysis systems are evaluated using F1-score, with precision and recall values being recorded over the entire vocabulary and not just inflected words. This kind of evaluation would be possible if the above limitation on making analysis decisions could be addressed.

Applying these techniques to day-to-day written texts, such as newswire, would test their robustness. The QAC, having a vocabulary with relatively few uninflected words, is not completely representative of naturally composed Arabic text in current daily usage. Some researchers (e.g. Rodrigues & Cavar, 2007; Xanthos, 2007) have used only artificially inflected words derived from verb conjugators or the Buckwalter morphological analyser (BAMA) in order to test their root extraction methods. Such word lists do not gauge the true effectiveness of their techniques for practical applications.

A further issue that is not investigated in this thesis is the size of corpus used for morphology learning. The QAC contains approximately 7000 word types. It would be interesting to experiment with smaller random samples of the corpus to determine the impact of corpus size.

6.5 Future Work

Having considered some of the weaknesses and shortcomings, the following areas warrant further investigation:

- Address the analysis decision problem:
    - Formulate a method to determine whether to analyse a word or leave it unanalysed.
    - Automatically determine a value for the morpheme list cut-off.
    - Experiment with different approaches to determine the *null-pattern* count.
    - Evaluate morphological analysis output for the entire corpus vocabulary using F1 score.

- Apply the technique to other corpora such as the Penn Arabic Treebank[10], or the Arabic Gigaword[11]. Unfortunately, the only satisfactory evaluation corpus for non-concatenative morphological analysis is currently the QAC. To produce another gold standard, one might apply an existing morphological analyser that outputs all possible analyses and disambiguate the result by hand.

- Extend the technique to apply it to dialects such as Egyptian Arabic and also to other Semitic languages such as Hebrew. The morphology of Semitic languages has the unique unifying aspect that it is based on a templatic structure, distinguishing it from other language families. The unsupervised technique presented in this thesis could in principle accommodate the variations in patterns of the various Semitic languages. The corpus normalization procedure would however have to be adjusted to remove diacritics appropriately for each language as they have different sets of short and long vowels and also possibly different conventions of omission and inclusions in written text. Other minor regularization might be required, e.g. for Hebrew normalising the *sofit* (final) letters to non-sofit form.

- By treating the corpus used for learning as running text, it may be possible to apply unsupervised pre-processing methods to distinguish word types into nouns, verb, particles, proper nouns etc. and apply the morphology learning technique to word subgroups in order to obtain pattern templates for respective groups, or paradigms. These subgroups may be obtained by applying automatic syntactic categorization, for example as proposed by Clark (2000).

- Compare contrastive learning with the widely used MDL-based approaches to morphology learning. This will give an understanding of how this technique compares to compression based techniques for unsupervised learning.

---

[10] Website: http://www.ircs.upenn.edu/arabic/

[11] Website: http://catalog.ldc.upenn.edu/LDC2011T11

- Exploit the large amount of research and advances made in ranking webpages using Link Analysis Ranking (LAR) algorithms – to which contrastive learning bears much resemblance. One promising avenue of investigation is the SALSA algorithm (Lempel and Moran, 2000), which is essentially the same as the refinement part of contrastive learning (section 5.2), but applies an alternative stationary solution rather than a non-convergent recursive formula. The algorithm would consider each morpheme's count in each connected component of the adjacency graph, rather than the overall structure of the graph.

- The contrastive learning approach could be applied directly to unstemmed data, although in this case overlapping patterns would be underrepresented due to data sparsity. This could be addressed by considering pattern variety count normalization similar to the root variety normalization carried out in section 5.3.

- Contrastive learning could be adapted for concatenative morphology by learning affix-stem pairs instead of roots and patterns. Alternatively, the technique could be adjusted to consider a pair of adjacency graphs: prefix-stem, stem-suffix, with a composite score for each possible analysis.

- Finally, the contrastive learning technique could be used to enhance the performance of existing stemmers when they are applied to a corpus: since the root and pattern lists are known, these morphemes could be scored using the iterative scoring procedure to help order the morphemes.

6.6 Outlook

This research has demonstrated the feasibility of unsupervised learning of non-concatenative morphology, with performance comparable to that of manually based systems. It has the potential to adapt to different languages while assuming no prior knowledge about the language. There is good potential to further develop and improve the techniques and to apply them in new settings.

Bibliography

Al-Kabi, M. N., Al-Radaideh, Q. A., Akkawi K. W. (2011). Benchmarking and assessing the performance of Arabic stemmers. *Journal of Information Science*, 37(111).

Al-Shalabi, R. (2005). Pattern-based stemmer for finding Arabic roots. *Information Technology Journal*, 4(1), 38–43.

Al-Shawakfa, E., Al-Badarneh, A., Shatnawi, S., Al-Rabab'ah, K., Bani-Ismail, B. (2010). A comparison study of some Arabic root finding algorithms. *Journal of the American Society for Information Science*, 61(5), 1015–1024.

Andreev, N. D. (1959). Modelirovanije jazyka na base ego statističeskoj i teoretiko-množestvennoj struktury (Modelling languages on the basis of their statistical and set theoretical structure). In *Tezisy soveščanija pomatematičeskoj lingvistike*, 14–21 *Aprelja 1959 goda*. Ministerstvo vysšego obrazovanija SSSR, Leningrad, pages 15–22.

Andreev, N. D. (1963). Algoritmy statistiko-kombinatornogo modelirovanija morfologii, sintaksisa, slovoobrazovanija i semantiki (Algorithms for statistical-combinatory modelling of morphology, syntax, word formation and semantics). In *Materialy po matemati˘ceskoj lingvistike i mašinomu perevodu: Sbornik II*. Izdatel'stvo Leningradskogo universiteta, Leningrad, pages 3–44.

Andrew, G., Gao, J. (2007). Scalable training of L_1-regularized log-linear models. *International Conference on Machine Learning* (pp. 33–40).

Attia, M., Pecina, P., Tounsi, L., Toral, A., Genabith J. (2011). Lexical Profiling for Arabic. *Electronic Lexicography in the 21st Century*, Bled, Slovenia.
Source of Morphological Patterns: http://www.attiaspace.com/

Baroni, M., Matiasek, J., Trost, H. (2002). Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the Workshop on Morphological & Phonological Learning of at ACL*, pages 48–57.

Beesley, K. R. (1996). Arabic finite-state morphological analysis and generation. In *Proceedings of the 16th International Conference on Computational Linguistics* (COLING '96), volume 1, pages 89-94.

Berger, A. (1997). *The improved iterative scaling algorithm: A gentle introduction*. http://www.cs.cmu.edu/~aberger/pdf/scaling.pdf (last accessed 15th December 2014).

Berger, A., Della Pietra, S. A., Della Pietra, V. J. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39-71.

Berger, A., Miller, R. (1998). Just-in-time language modelling. *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

Bloomfield, L. (1933). *Language*. Henry Holt & Co, New York.

Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P. (2005). Link Analysis Ranking Algorithms Theory And Experiments. *ACM Transactions on Internet Technology*, 5(1). 231-297.

Brin, S., Page, L. (1998). Anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference,* Brisbane, Australia, Apr. 14–18. pp. 107–117.

Cavar, D., Herring, J., Ikuta, T., Rodrigues, P., Schrementi, G. (2004). On Induction of Morphology Grammars and its Role in Bootstrapping. In *Proceedings of the 9th Conference on Formal Grammar*, pages 47–62, 2004.

Cavar, D., Rodrigues, P., Schrementi, G. (2005). Unsupervised morphology induction for part-of-speech tagging. In *U. Penn Working Papers in Linguistics: Proceedings of the 29th Annual Penn Linguistics Colloquium*, pages 47–62, 2005.

Chen, S. F., Rosenfeld, R. (2000). A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Chung, T., Gildea, D. (2009). Unsupervised Tokenization for Machine Translation. In *Proceedings of EMNLP 2009*, pages 718–726, Singapore.

Clark, A. (2000). Inducing syntactic categories by context distribution clustering. In *Proceedings of the Fourth Conference on Natural Language Learning* (*CoNLL-2000 and LLL-2000)*, pages 91-94.

Clark, A. (2001). Learning Morphology with Pair Hidden Markov Models. In *Proceedings of the Student Workshop at the 39th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 55–60, Toulouse, France.

Clark, A. (2002). Memory-based learning of morphology with stochastic transducers. In *Proceedings of the 40th Annual Meeting on of the Association for Computational Linguistics*, pages 513–520, Morristown, NJ, USA.

Clark, A. (2007). Supervised and unsupervised learning of Arabic morphology. In *Arabic Computational Morphology*, volume 38 of Text, Speech and Language Technology, pages 181–200. Springer Netherlands.

Clark, A., Lappin, S. (2010). Unsupervised learning and grammar induction. In *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley-Blackwell.

Creutz, M., Lagus, K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process*., 4(1):1–34.

Curran, J. R., Clark, S. (2003). Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Meeting of the EACL*, pages 91–98, Budapest, Hungary.

Dahlgren, S. (1998). *Word Order in Arabic*. Goteborg: Acta Universitatis Gothoburgensis.

Darroch, J., Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, Vol. 43:pp 1470-1480.

Darwish, K. (2002). Building a shallow Arabic morphological analyser in one day. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 22–29, Philadelphia, July.

Darwish, K., Oard, D. (2007). Adapting Morphology for Arabic Information Retrieval. In *Adapting Morphology for Arabic Information Retrieval*, pp. 245-262, Springer.

Daya, E. (2004). Learning Hebrew roots: Machine learning with linguistic constraints. In *Proceedings of EMNLP04*, pages 357–364, 2004.

Daya, E., Roth, D., Wintner, S. (2008). Identifying Semitic roots: Machine learning with linguistic constraints. *Computational Linguistics*, 34:429–448.

Dayan, P. (1999). Unsupervised learning. *The MIT Encyclopaedia of the Cognitive Science*.

De Pauw, G., Wagacha, P. (2007). Bootstrapping morphological analysis of Gikuyu using unsupervised maximum entropy learning. In *Proceedings of the Eighth Annual Conference of the International Speech Communication Association*. Antwerp, Belgium.

De Pauw, G., Wagacha, P. Abade, D. (2007). Unsupervised induction of Dholuo word classes using maximum entropy learning. In *Proceedings of the First International Conference in Computer Science and Informatics (COSCIT 2007).* Nairobi, Kenya: University of Nairobi.

Dukes, K., Habash, N. (2010). Morphological Annotation of Quranic Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Malta.

Elghamry, K. (2004). A Constraint-based Algorithm for the Identification of Arabic Roots. *Proceedings of the Midwest Computational Linguistics Colloquium*. Indiana University. Bloomington.

Fabri, R., Gasser, M., Habash, N., Kiraz, G., Wintner, S. (2014). Linguistic introduction: The orthography, morphology and syntax of Semitic languages. In *Natural Language Processing of Semitic Languages*. Berlin and Heidelberg: Springer, pp. 3–41.

Farahat, A., LoFaro, T., Miller, J. C., Raey, G., Ward, L. A. (2006). Authority Rankings from HITS, PageRank, and SALSA: Existence, Uniqueness, and Effect of Initialization. *SIAM Journal on Scientific Computing.* Volume 27, Issue 4, pp. 1181-1201.

Ghwanmeh, S., Al-Shalabi, R., Kanaan, G., Khanfar, K., Rabab'ah, S. (2005). An algorithm for extracting the root for the Arabic language. Paper presented at the *Fifth International Business Information Management Association Conference (IBIMA)*, Cairo, Egypt.

Goldsmith, J. (2000). Linguistica: An automatic morphological analyser. In *Proceedings from the Main Session of the Chicago Linguistic Society's thirty-sixth Meeting*, pages 125–139. Chicago Linguistics Society, Chicago.

Goldsmith, J. (2001). Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198.

Goldsmith, J. (2006). An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(4):353–371.

Goldsmith, J., Xanthos, A., Goldsmith, J. (2009). Learning phonological categories. *Language*. *Project Muse*, 85.1:4–38.

Golub, G., Van Loan, C.F. (1989). *Matrix Computations*, Johns Hopkins University Press.

Habash, N. (2010). *Introduction to Arabic Natural Language Processing*. Morgan & Claypool.

Hafer, M. A., Weiss, S. F. (1974). Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10(11-12):371 – 385.

Hammarström, H. (2006a). A naive theory of affixation and an algorithm for extraction. In *SIGPHON '06: Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology*, pages 79–88, Morristown, NJ, USA.

Hammarström, H. (2006b). Poor man's stemming: Unsupervised recognition of same stem words. *Information Retrieval Technology*, volume 4182 of Lecture Notes in Computer Science, pages 323–337. Springer Berlin / Heidelberg.

Hammarström, H. (2007a). A Survey and Classification of Methods for (Mostly) Unsupervised Learning of Morphology. In *NODALIDA, the 16th Nordic Conference of Computational Linguistics*, Tartu, Estonia, pages 25–26..

Hammarström, H. (2007b). Unsupervised Learning of Morphology: Survey, Model, Algorithm and Experiments. In *Proceedings of iNEWS-07Workshop at SIGIR 2007, pages 14-20*.

Hammarström, H. (2009). *Unsupervised Learning of Morphology and the Languages of the World*. Ph.D. thesis, Chalmers University of Technology and University of Gothenburg.

Harris, Z. S. (1955). From phoneme to morpheme. *Language*, 31(2):190–222.

Holes, C. (2004). Modern Arabic: Structures, Functions, and Varieties. *Georgetown Classics in Arabic Language and Linguistics*. Georgetown University Press.

Holes, C. D. (1995). *Modern Arabic: Structures, Functions and Varieties.* London and New York: Longman.

Huang, F., Hsieh, C., Chang K., Lin, C. (2010). Iterative scaling and coordinate descent methods for maximum entropy. *Journal of Machine Learning Research*, 11:815–848.

Khoja, S., Garside, R. (1999). *Stemming Arabic text*. Lancaster, UK: Lancaster University, Computing Department.

Kirchhoff, K., Bilmes, J., Henderson, J., Schwartz, R., Noamany, M., Schone, P., Ji, G., Das, S., Egan, M., He, F., Vergyri, D., Liu, D., Duta, N. (2002). *Novel speech recognition models for Arabic.* Technical Report, Johns Hopkins University.

Kirchhoff, K., Vergyri, D., Bilmes, J., Duh, K., Stolcke, A. (2006). Morphology-based language modeling for Arabic speech recognition. *Computer. Speech and Language*. 20, 4, 589–608.

Klein, D., Manning, C. (2002). Natural language grammar induction using a constituent-context model. In *Advances in Neural Information Processing Systems (NIPS 14)*, pages 35–42.

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment, *Journal of the ACM*, 46, pp. 604-632.

Klenk, U. (1985). Recognition of Spanish inflectional endings based on the distribution of characters. In *Computers in Literary and Linguistic Computing: Proceedings of the Eleventh International Conference [L'ordinateur et les recherches littéraires et linguistiques: actes de la XIe Conférence internationale]*, pages 246–253, Louvain.

Lee,Y., Papineni, K., Roukos, S., Emam O., Hassan, H. (2003). Language model based Arabic word segmentation. *In ACL '03: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 399–406, Morristown, NJ, USA.

Lempel, R., Moran, S. (2000). The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect. *Computer Networks* 33:1-6 , 387-401.

Liang, P., Klein, D. (2009). Online EM for unsupervised models. In *Proceedings of NAACL-HLT*, pages 611-619.

Lipinski, E. (1997). *Semitic Languages: Outline of a Comparative Grammar*. Leuven: Peters.

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL-2002*, pages 49-55.

McCarthy, J. J. (1979). *Formal Problems in Semitic Phonology and Morphology*. Ph.D. dissertation, MIT. (Published by Garland Press, New York, 1985.)

Moukdad, H. (2006). Stemming and root-based approaches to the retrieval of Arabic documents on the Web. *Webology*, 3(1).

Ohno, S., Kidera, S., Kirimoto, T. (2013). Efficient automatic target recognition method for aircraft SAR image using supervised SOM clustering. In *Asia-Pacific Conference on Synthetic Aperture Radar (APSAR)*, pages 601-604.

Pietra, S. D., Pietra, V. D., Lafferty, J. (1995). *Inducing Features of Random Fields*. Technical Report CMUCS-CS95-144, School of Computer Science, Carnegie Mellon University.

Poon, H., Cherry, C., Toutanova, K. (2009). Unsupervised morphological segmentation with log-linear models. In *the Proceedings of NAACL '09: The 2009 Annual Conference of the North American Association for Computational Linguistics*, pages 209–217, Morristown, NJ.

Ratnaparkhi, A. (1998). *Maximum Entropy Models for Natural Language Ambiguity Resolution.* PhD thesis, University of Pennsylvania.

Rodrigues, P., Cavar, D. (2005). Learning Arabic Morphology Using Information Theory. In *Proceedings from the Annual Meeting of the Chicago Linguistic Society*, 41(2):49–58.

Rodrigues, P., Cavar, D. (2007). Learning Arabic Morphology Using Statistical Constraint Satisfaction Models. In *Perspectives on Arabic Linguistics XIX: Proceedings of the 19th Arabic Linguistics Symposium*, Urbana, IL, USA.

Schone, P., Jurafsky, D. (2000). Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 67–72, Lisbon.

Schone, P., Jurafsky, D. (2001). Knowledge-free induction of inflectional morphologies. *In Proceedings of the North American Chapter of the ACL*, pages 183–191.

Snyder, B., Barzilay, R. (2008). Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio.

Sonbol, R., Ghneim, N., Desouki, M.S. (2008). Arabic morphological analysis: A new approach. In *Proceedings of the Third International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA 2008)* (pp. 1-6). Piscataway, NJ:IEEE.

Taghva, K., Elkhoury, R., Coombs, J. (2005). Arabic stemming without a root dictionary. In *Proceedings of ITCC 2005 International Conference on Information Technology: Coding and Computing*.

Thompson, I., Philips, J. (2013). *About world languages: Semitic Branch*. http://www.aboutworldlanguages.com/semitic-branch (last accessed 15th December 2014).

Tsaparas, P. (2004). *Link analysis ranking.* Ph.D. thesis, University of Toronto.

von Mises, R., Pollaczek-Geiringer, H. (1929). Praktische Verfahren der Gleichungsauflösung (Practical methods for solving equations). *ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik.* Volume 9, Issue 2, pp. 152-164.

Watson, J. (2002). *The Phonology and Morphology of Arabic*. Oxford: Oxford University Press.

Wothke, K. (1986). Machine learning of morphological rules by generalization and analogy. In *Proceedings of the 11th Conference on Computational Linguistics*, pages 289–293, Morristown, NJ.

Xanthos, A. (2007). *Apprentissage automatique de la morphologie: le cas des structures racine-schéme*. PhD thesis, University of Lausanne.

Xue, N. (2003). Chinese Word Segmentation as Character Tagging. *International Journal of Computational Linguistics and Chinese Language Processing*, 8(1).

Yarowsky, D., Wicentowski, R. (2000). Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (ACL), pages 207–216, Hong Kong.

Zhang L. (2004). *Maximum entropy modelling toolkit for Python and C++*. Technical Report, Centre for Speech Technology Research of the University of Edinburgh.

Appendix A

Buckwalter Transliteration

A.1 Original Buckwalter transliteration[12]

Table A.1 below gives a one-to-one mapping of most Modern Standard Arabic (MSA)characters in common usage to Latin characters and symbols, as transcribed by Tim Buckwalter[13].

UNICODE			BUCKWALTER	
Decimal	**Hex**	**Glyph**	**ASCII**	**Orthography**
1569	U+0621	ء	**'**	Hamza
1571	U+0623	أ	**>**	Alif + HamzaAbove
1572	U+0624	ؤ	**&**	Waw + HamzaAbove
1573	U+0625	إ	**<**	Alif + HamzaBelow
1574	U+0626	ئ	**}**	Ya + HamzaAbove
1575	U+0627	ا	**A**	Alif
1576	U+0628	ب	**b**	Ba
1577	U+0629	ة	**p**	TaMarbuta
1578	U+062A	ت	**t**	Ta

[12] Source: http://corpus.quran.com/java/buckwalter.jsp

[13] http://www.qamus.org/transliteration.htm

UNICODE			BUCKWALTER	
Decimal	**Hex**	**Glyph**	**ASCII**	**Orthography**
1579	U+062B	ث	**v**	Tha
1580	U+062C	ج	**j**	Jeem
1581	U+062D	ح	**H**	HHa
1582	U+062E	خ	**x**	Kha
1583	U+062F	د	**d**	Dal
1584	U+0630	ذ	*****	Thal
1585	U+0631	ر	**r**	Ra
1586	U+0632	ز	**z**	Zain
1587	U+0633	س	**s**	Seen
1588	U+0634	ش	**$**	Sheen
1589	U+0635	ص	**S**	Sad
1590	U+0636	ض	**D**	DDad
1591	U+0637	ط	**T**	TTa
1592	U+0638	ظ	**Z**	DTha
1593	U+0639	ع	**E**	Ain
1594	U+063A	غ	**g**	Ghain
1600	U+0640	ـ	**_**	Tatweel
1601	U+0641	ف	**f**	Fa

UNICODE			BUCKWALTER	
Decimal	**Hex**	**Glyph**	**ASCII**	**Orthography**
1602	U+0642	ق	**q**	Qaf
1603	U+0643	ك	**k**	Kaf
1604	U+0644	ل	**l**	Lam
1605	U+0645	م	**m**	Meem
1606	U+0646	ن	**n**	Noon
1607	U+0647	ه	**h**	Ha
1608	U+0648	و	**w**	Waw
1609	U+0649	ى	**Y**	AlifMaksura
1610	U+064A	ي	**y**	Ya
1611	U+064B	ً	**F**	Fathatan
1612	U+064C	ٌ	**N**	Dammatan
1613	U+064D	ٍ	**K**	Kasratan
1614	U+064E	َ	**a**	Fatha
1615	U+064F	ُ	**u**	Damma
1616	U+0650	ِ	**i**	Kasra
1617	U+0651	ّ	**~**	Shadda
1618	U+0652	ْ	**o**	Sukun
1648	U+0670	ٰ	**`**	AlifKhanjareeya

UNICODE			BUCKWALTER	
Decimal	**Hex**	**Glyph**	**ASCII**	**Orthography**
1649	U+0671	ٱ	**{**	Alif + HamzatWasl

Table A.1: Buckwalter Transliteration

A.2 Extended Transliteration[14]

Table A.2 below shows the extended transliteration mapping for certain characters used in classical Arabic

UNICODE			BUCKWALTER	
Decimal	**Hex**	**Glyph**	**ASCII**	**Orthography**
1619	U+0653	ٓ	^	Maddah
1620	U+0654	ٔ	#	HamzaAbove
1756	U+06DC	ۜ	**:**	SmallHighSeen
1759	U+06DF	۟	**@**	SmallHigh-RoundedZero
1760	U+06E0	۠	**"**	SmallHighUpright-RectangularZero
1762	U+06E2	ۢ	**[**	SmallHighMeem-

[14] Source: http://corpus.quran.com/java/buckwalter.jsp

UNICODE			BUCKWALTER	
Decimal	**Hex**	**Glyph**	**ASCII**	**Orthography**
				IsolatedForm
1763	U+06E3	ۣ	**;**	SmallLowSeen
1765	U+06E5	ۥ	**,**	SmallWaw
1766	U+06E6	ۦ	**.**	SmallYa
1768	U+06E8	ۨ	**!**	SmallHighNoon
1770	U+06EA	۪	**-**	EmptyCentreLowStop
1771	U+06EB	۫	**+**	EmptyCentreHighStop
1772	U+06EC	۬	**%**	RoundedHighStopWithFilledCentre
1773	U+06ED	ۭ	**]**	SmallLowMeem

Table A.2: Extended transliteration for classical Arabic characters

Appendix B

Undiacritized Pattern List

B.1 Unvowelled Pattern from

Table B.1 below gives the unvowelled conflated patterns from vowelled pattern from Attia's compilation of Arabic Morphology Patterns (Attia *et al*, 2011).

Word Forms (unvowelled)		Pattern
ا...	(...A)	...A
تا...	(...tA)	...tA
تان...	(...tAn)	...tAn
تفعال	(tfEAl)	t - - A -
تفع	(tfE)	t - -
تفعلة	(tfElp)	t - - - p
تفعل	(tfEl)	t - - -
تفعي	(tfEy)	t - - y
تفعى	(tfEY)	t - - Y
تفعيلة	(tfEylp)	t - - y - p
تفعلل	(tfEll)	t - - - -
تفعيل	(tfEyl)	t - - y -
تفاع	(tfAE)	t - A -

Word Forms (unvowelled)		Pattern
تفاعل	(tfAEl)	t - A - -
تفاعي	(tfAEy)	t - A - y
تفاعى	(tfAEY)	t - A - Y
تفاعيل	(tfAEyl)	t - A - y -
تي...	(...ty)	...ty
تين...	(...tyn)	...tyn
آعال	(|EAl)	| - A -
آعل	(|El)	| - -
آعلة	(|Elp)	| - - p
آعية	(|Eyp)	| - yp
اء...	(...A')	...A'
ات...	(...At)	...At
ة...	(...p)	...p

Word Forms (unvowelled)		Pattern
ان...	(...An)	...An
آل	(|l)	| -
أفع	(>fE)	> - -
أفعة	(>fEp)	> - - p
أفعاع	(>fEAE)	> - - A -
إفعاء	(<fEA')	< - - A'
أفعال	(>fEAl)	> - - A -
إفعال	(<fEAl)	< - - A -
إيعال	(<yEAl)	<y - A -
أفعل	(>fEl)	> - - -
إفعل	(<fEl)	< - - -
أفعى	(>fEY)	> - - Y
أفعلة	(>fElp)	> - - - p
أفعية	(>fEyp)	> - - yp
أفعلاء	(>fElA')	> - - - A'
أفعلان	(>fElAn)	> - - - An
أفعول	(>fEwl)	> - - w -
أفاعل	(>fAEl)	> - A - -

Word Forms (unvowelled)		Pattern
أفاعي	(>fAEy)	> - A – y
أفاعلة	(>fAElp)	> - A - - p
أفاعيل	(>fAEyl)	> - A - y -
آفاء	(|fA')	| - A'
آفال	(|fAl)	| - A -
أفال	(>fAl)	> - A -
أفالة	(>fAlp)	> - A – p
أفل	(>fl)	> - -
أفلة	(>flp)	> - - p
أفلاء	(>flA')	> - - A'
أفيال	(>fyAl)	> - yA -
فع	(fE)	- -
فعع	(fEE)	- - -
فععة	(fEEp)	- - - p
فعا	(fEA)	- - A
فعاعل	(fEAEl)	- - A - -
فعاة	(fEAp)	- - Ap
فعائل	(fEA}l)	- - A} -

Word Forms (unvowelled)		Pattern
فعال	(fEAl)	- - A -
فعالا	(fEAlA)	- - A – A
فعالة	(fEAlp)	- - A – p
فعايا	(fEAyA)	- - AyA
فعالل	(fEAll)	- - A - -
فعالي	(fEAly)	- - A – y
فعالى	(fEAlY)	- - A – Y
فعايل	(fEAyl)	- - Ay -
فعاللة	(fEAllp)	- - A - - p
فعاليل	(fEAlyl)	- - A - y -
فعل	(fEl)	- - -
فعي	(fEy)	- - y
فعى	(fEY)	- - Y
فعلة	(fElp)	- - - p
فعلاء	(fElA')	- - - A'
فعلال	(fElAl)	- - - A -
فعلان	(fElAn)	- - - An
فعلالل	(fElAll)	- - - A - -

Word Forms (unvowelled)		Pattern
فعلالية	(fElAlyp)	- - - A – yp
فعلل	(fEll)	- - - -
فعلى	(fElY)	- - - Y
فعول	(fEwl)	- - w -
فعيل	(fEyl)	- - y -
فعلوت	(fElwt)	- - - wt
فعللة	(fEllp)	- - - - p
فعولة	(fEwlp)	- - w - p
فعيلة	(fEylp)	- - y - p
فعللان	(fEllAn)	- - - - An
فعلوان	(fElwAn)	- - - wAn
فعلول	(fElwl)	- - - w -
فعليل	(fElyl)	- - - y -
فعوlل	(fEwll)	- - w - -
فعللية	(fEllyp)	- - - - yp
فعللول	(fEllwl)	- - - - w -
آستفعال	({stfEAl)	{st - - A -
مستفعل	(mstfEl)	mst - - -

Word Forms (unvowelled)		Pattern
فاع	(fAE)	- A -
فاعل	(fAEl)	- A - -
فاعى	(fAEY)	- A - Y
مآعل	(m|El)	m| - -
فاعلة	(fAElp)	- A - - p
فاعول	(fAEwl)	- A - w -
فاعولة	(fAEwlp)	- A - w - p
فال	(fAl)	- A -
فالة	(fAlp)	- A - p
مآلل	(m|ll)	m| - -
ية...	(...yp)	...yp
فل	(fl)	- -
مفع	(mfE)	m - -
مفعاة	(mfEAp)	m - - Ap
ٱفعال	({fEAl)	{ - - A -
فيعال	(fyEAl)	- y - A -
فيعان	(fyEAn)	- y - An
مفعال	(mfEAl)	m - - A -

Word Forms (unvowelled)		Pattern
مستفعي	(mstfEy)	mst - - y
ٱستفعل	({stfEl)	{st - - -
ٱستفعى	({stfEY)	{st - - Y
مستفال	(mstfAl)	mst - A -
ٱستفالة	({stfAlp)	{st - A - p
ٱستفال	({stfAl)	{st - A -
مستفل	(mstfl)	mst - -
ٱستيفال	({styfAl)	{sty - A -
ٱستفل	({stfl)	{st - -
مستفيل	(mstfyl)	mst - y -
ٱتعال	({tEAl)	{t - A -
متعل	(mtEl)	mt - -
ٱتعل	({tEl)	{t - -
متفعل	(mtfEl)	mt - - -
متفعي	(mtfEy)	mt - - y
متفعلل	(mtfEll)	mt - - - -
متفاعل	(mtfAEl)	mt - A - -
متفال	(mtfAl)	mt - A -

Word Forms (unvowelled)		Pattern
ٱفعل	({fEl)	{ - - -
فوعل	(fwEl)	- w - -
فيعل	(fyEl)	- y - -
مفعل	(mfEl)	m - - -
مفعي	(mfEy)	m - - y
مفعى	(mfEY)	m - - Y
ٱفعلعان	({fElEAn)	{ - - - - An
ٱفعوعل	({fEwEl)	{ - - w - -
فوعلة	(fwElp)	- w - - p
مفعلة	(mfElp)	m - - - p
فلعلاة	(flElAp)	- - - - Ap
ٱفعلال	({fElAl)	{ - - - A -
مفعلان	(mfElAn)	m - - - An
فوعلالية	(fwElAlyp)	- w - - A – yp
ٱفعلل	({fEll)	{ - - - -
مفعلل	(mfEll)	m - - - -
مفعول	(mfEwl)	m - - w -
مفعيل	(mfEyl)	m - - y -

Word Forms (unvowelled)		Pattern
يفعول	(yfEwl)	y - - w -
ٱفتعال	({ftEAl)	{ - t - A -
ٱفتعالة	({ftEAlp)	{ - t - A – p
ٱفتعل	({ftEl)	{ - t - -
ٱفتعى	({ftEY)	{ - t – Y
مفتعل	(mftEl)	m - t - -
ٱفتال	({ftAl)	{ - tA -
ٱفتل	({ftl)	{ - t -
مفتل	(mftl)	m - t -
فواع	(fwAE)	- wA -
مفاع	(mfAE)	m - A -
مفاعاة	(mfAEAp)	m - A – Ap
فلاعل	(flAEl)	- - A - -
ٱفطعل	({fTEl)	{ - T - -
فواعل	(fwAEl)	- wA - -
فياعل	(fyAEl)	- yA - -
مفاعل	(mfAEl)	m - A - -
مفطعل	(mfTEl)	m - T - -

Word Forms (unvowelled)		Pattern
مفاعلة	(mfAElp)	m - A - - p
فواعيل	(fwAEyl)	- wA - y -
مفاعيل	(mfAEyl)	m - A - y -
يفاعيل	(yfAEyl)	y - A - y -
مفائل	(mfA}l)	m - A} -
فلال	(flAl)	- - A -
مفال	(mfAl)	m - A -
فلل	(fll)	- - -
فول	(fwl)	- w -
فيل	(fyl)	- y -
مفل	(mfl)	m - -
ٱنفعاء	({nfEA')	{n - - A'
ٱنفعال	({nfEAl)	{n - - A -
ٱنفعالة	({nfEAlp)	{n - - A - p
منفعل	(mnfEl)	mn - - -
ٱنفعل	({nfEl)	{n - - -

Word Forms (unvowelled)		Pattern
ٱنفعى	({nfEY)	{n - - Y
فولان	(fwlAn)	- w - An
فيلان	(fylAn)	- y - An
ٱنفال	({nfAl)	{n - A -
مفول	(mfwl)	m - w -
مفيل	(mfyl)	m - y -
منفل	(mnfl)	mn - -
موفل	(mwfl)	mw - -
ٱنفل	({nfl)	{n - -
مفولة	(mfwlp)	m - w - p
مفيلة	(mfylp)	m - y - p
ون...	(...wn)	...wn
ين...	(...yn)	...yn
و...	(...w)	...w
ي...	(...y)	...y

Table B.1: Undiacritized patterns of Arabic

Appendix C

Processing the Quranic Arabic Corpus

I obtained the vocabulary from the Quranic Arabic Corpus (QAC) tagged with morphological data contained in a text file downloaded from the QAC website[15]. The QAC uses the Buckwalter transliteration, as in section A.1, covering the character set pertaining to MSA; it is extended to transliteration mapping, as in section A.2, catering for classical Arabic text as found in the QAC.

From this dataset I selected only the stems and thereafter filtered diacritical markers shown in Table C.1.

ASCII (Buckwalter)	**Glyph (Corresponding)**
a | u | i | o | ~ | ^ | :	◌َ | ◌ُ | ◌ِ | ◌ْ | ◌ّ | ◌ٰ | ◌ٓ

Table C.1**:** Omitted diacritical markers

I conflated all long vowels with 'humza' marker (◌ٔ | ◌ٕ) to single letter humza (ء), shown in Table C.2. Also, I changed the diacritic marker 'Alif Khanjareeya,' '^' (◌ٰ) to an 'Alif' 'A' (ا).

[15] http://corpus.quran.com/download/default.jsp

Letter with 'Humza'		Conflated to 'Humza'	
ASCII	**Glyph**		
{ | < | > | & | }	ئ | ؤ | إ | أ | آ	'	ء

Table C.2: Conflation of Humza letters to single letter

Table C.3 shows an example of the vocabulary from the first chapter of the Quran, consisting of undiacritized stems (along with the Arabic script shown in the 3rd and 4th columns) used in the experiments after applying the above processing to the stemmed words (shown in the 1st and 2nd columns).

Stemmed Words with Diacritics		Undiacritized Stems	
Buckwalter	**Arabic**	**Buckwalter**	**Arabic**
Somi	سْمِ	Sm	سم
{ll~ahi	ٱللَّهِ	'llh	ءلله
r~aHoma`ni	رَّحْمَٰنِ	rHmAn	رحمان
r~aHiymi	رَّحِيمِ	rHym	رحيم
Hamodu	حَمْدُ	Hmd	حمد
l~ahi	لَّهِ	Lh	له
rab~i	رَبِّ	Rb	رب

Stemmed Words with Diacritics		**Undiacritized Stems**	
Buckwalter	**Arabic**	**Buckwalter**	**Arabic**
Ea`lamiyna	عٰلَمِينَ	EAlmyn	عالمين
r~aHoma`ni	رَّحْمٰنِ	rHmAn	رحمان
r~aHiymi	رَّحِيمِ	rHym	رحيم
ma`liki	مٰلِكِ	mAlk	مالك
yawomi	يَوْمِ	ywm	يوم
d~iyni	دِّينِ	dyn	دين
naEobudu	نَعْبُدُ	nEbd	نعبد
nasotaEiynu	نَسْتَعِينُ	nstEyn	نستعين
{hodi	ٱهْدِ	‘hd	ءهد
S~ira`Ta	صِّرٰطَ	SrAT	صراط
musotaqiyma	مُسْتَقِيمَ	mstqym	مستقيم
Sira`Ta	صِرٰطَ	SrAT	صراط
>anoEamo	أَنْعَمْ	'nEm	ءنعم

Stemmed Words with Diacritics		**Undiacritized Stems**	
Buckwalter	**Arabic**	**Buckwalter**	**Arabic**
gayori	غَيْرِ	gyr	غير
magoDuwbi	مَغْضُوبِ	mgDwb	مغضوب
D~aA^l~iyna	ضَّآلِّينَ	DAlyn	ضالين

Table C.3: Orthographic changes from diacritized to undiacritized text