Abdelhadi Soudi
Antal van den Bosch
Günter Neumann
Editors

TEXT, SPEECH AND LANGUAGE TECHNOLOGY 38

Arabic Computational Morphology

Knowledge-based and Empirical Methods

Springer

Arabic Computational Morphology

Text, Speech and Language Technology

VOLUME 38

Arabic Computational Morphology

Knowledge-based and Empirical Methods

Edited by

Abdelhadi Soudi

Ecole Nationale de l'Industrie Minérale, Rabat, Morocco

Antal van den Bosch

Tilburg University, The Netherlands

Günter Neumann

Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4020-6045-8 (HB)
ISBN 978-1-4020-6046-5 (e-book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springer.com

Printed on acid-free paper

Contents

Preface

One of the advantages of having worked in a field for twenty years is that you have an opportunity to watch research areas grow from infancy into maturity. The present book represents a marriage of two such fields: computational morphology and Arabic computational linguistics.

In the mid 1980s, Koskenniemi had just published his landmark (1983) thesis on Two-Level Morphology. Prior to Koskenniemi there had of course been work on computational morphology dating back all the way to the 1960s, but the field had never been a major focus of research. Koskenniemi changed that by taking the computational framework of finite-state transducers, proposed by Ron Kaplan and Martin Kay, and making it actually work in a real system. Koskenniemi provided a practical implementation of a well-defined computational model, and this in turn led to an explosion of work in finite-state and other approaches to morphology over the ensuing twenty years. Data-driven methods, which gained popularity in the late 1980s took a few years to make an impact on computational morphology, but in the last decade there has been a significant amount of work, particularly in the area of self-organizing methods for morphological induction.

In the mid 1980s, with notable exceptions like early work by Beesley, there was next to nothing being done on Arabic. There simply were not the resources, nor were there very many people who both had the linguistic training and knowledge of Arabic, as well as training in natural language processing. All of this has changed. Now there are quite a few resources for Arabic including the roughly 400 million words of Modern Standard Arabic newswire text in the Arabic Gigaword corpus, the Penn Arabic Treebank, the Prague Dependency treebank, Tim Buckwalter's publicly available morphological analyzer, as well as a growing set of resources for Colloquial Arabic, including the Egyptian, Levantine, Iraqi and Gulf dialects. As evidenced by the contributors to this volume, there are now a large number of computational linguists with a knowledge of Arabic. And perhaps most importantly, there is a widespread interest in the community as a whole in Arabic language processing.

Like all good marriages, the union of computational morphology with Arabic language processing is one fraught with complexity; for Arabic seems almost to have been specially engineered to maximize the difficulties for automatic processing. The famous Semitic "root-and-pattern" morphology defies a straightforward implementation in terms of morpheme concatenation, and this has spawned a wide variety of different computational solutions, many of which are represented in various chapters in this volume. Students of writing systems have speculated that this root-and-pattern morphology was ultimately responsible for the second interesting and difficult property of Arabic (and several other Semitic languages), namely that the writing system is impoverished in that a fair amount of phonological information is simply missing in the script. In its normal everyday use, the script systematically fails to represent not only most vowel information (is درس *DRS /darasa/, /durisa/*, …?), but also information on consonant gemination (is كتب *KTB /kataba/* or */kattaba/*?), as well as both vowel and nunation information in the nominal case system (is ولد *WLD /waladu/, /waladun/, /waladin/*, …?). If this weren't enough, as Tim Buckwalter shows, the advent of Unicode has failed to standardize Arabic encoding, so that in dealing with real texts, one has to be prepared to do a fair amount of low level normalization; to some extent the differences reflect regional variants (such as the use of */alif maqSūra/* for */ya/* in Egyptian texts), but in other cases they reflect the fact that for all of its attempts at rigid design, Unicode still allows for a fair amount of "wiggle room": the same issue comes up in the encoding of South Asian languages using Brahmi-derived scripts.

The chapters in this volume attest both to the wide variety, and to the sophistication of the work being done on the computational analysis of Arabic morphology, both in terms of approaches to morphological analysis, as well as in applications of such work to other areas such as machine translation and information retrieval. To be sure, part of the reason for the increased interest in Arabic language processing is due to greater funding opportunities for work on Arabic, and this in turn has been fueled by various important political events of the past few years. But it would be shortsighted to view this as the sole justification for an increased interest in Arabic. Forms of Arabic are spoken by roughly 250 million people in an area spanning North Africa to the Persian Gulf. It is the official language of over 20 countries. It is a significant minority language of a number of sub-saharan African countries. And there is a large expatriate population spread throughout the world. Arabic is thus one of the world's major languages. History, especially that of the last hundred years, has not been kind to the Arabic-speaking peoples, and they have not had an economic clout proportional to their population. This is bound to change sooner or later, and there will be an increasing need for tools that allow one to use Arabic in the digital world as easily as one can now use English.

The work represented in this book is an important milestone along the path towards that goal. I commend the editors — Abdelhadi Soudi, Antal van den Bosch, and Günter Neumann — and all of the contributing authors on its publication.

I wish to thank Elabbas Benmamoun for helpful feedback on an earlier draft of this preface.

Richard Sproat
University of Illinois at Urbana-Champaign

PART I

Introduction

1

Arabic Computational Morphology: Knowledge-based and Empirical Methods

Abdelhadi Soudi[1], Günter Neumann[2] and Antal van den Bosch[3]
[1] *Ecole Nationale de l'Industrie Minérale, Rabat, Morocco*
`asoudi@gmail.com`
[2] *Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany*
`Neumann@dfki.de`
[3] *Tilburg University, The Netherlands*
`Antal.vdnBosch@uvt.nl`

1.1 Overview

The morphology of Arabic poses special challenges to computational natural language processing systems. The exceptional degree of ambiguity in the writing system, the rich morphology, and the highly complex word formation process of roots and patterns all contribute to making computational approaches to Arabic very challenging. Indeed, many computational linguists across the world have taken up this challenge over time, and we have been able to commit many of the researchers with a track record in this research area to contribute to this book.

The book's subtitle aims to reflect that widely different computational approaches to the Arabic morphological system have been proposed. These accounts fall into two main paradigms: the knowledge-based and the empirical. Since morphological knowledge plays an essential role in any higher-level understanding and processing of Arabic text, the book also features a part on the integration of Arabic morphology in larger applications, namely Information Retrieval (IR) and Machine Translation (MT).

The book is unique in the following ways:

- It is the first comprehensive text that covers both knowledge-based and data-driven approaches to Arabic morphology;
- It provides broad but rigorous coverage of the computational techniques for the processing of Arabic morphology as well as a detailed discussion of the linguistic approaches on which each computational treatment is based;
- Compared and contrary to already published books in the area, the proposed book includes contributions in which authors demonstrate how their approaches

to morphology improve the performance of Natural Language Processing Applications, namely IR and MT, including experiments and results;
- While the book focuses primarily on Arabic computational morphology, the authors do show how their approaches could be extended to other Semitic languages;
- It brings together original and extended contributions from the most distinguished actors in knowledge-based and empirical paradigms as well as in Arabic MT and IR.

First, we offer a brief roadmap for the book. The book is opened by a Preface by Professor Richard Sproat who has a long-standing experience in computational morphology. Chapter 2 introduces the transliteration scheme used in this book to represent Arabic words for readers who cannot read the Arabic script and presents guidelines for pronouncing Arabic given this transliteration. Chapter 3 provides a review of the salient issues in Arabic computational morphology. Among the issues discussed are: the status of non-standard Arabic characters (e.g., Persian characters), problems in orthography relating to non-standard uses of Arabic characters, problems in orthography that affect tokenization, defining standard vs. non-standard orthography, defining contemporary morphological features (e.g., is the energetic verb form archaic, or simply rare?), designing a maintainable system (e.g., what level of specialized knowledge is required to make new entries in the lexicon?), and the need to extend the analysis to written colloquial Arabic, in view of its increasing and widespread use on the Internet.

1.2 Knowledge-based Approaches

Benefiting from the findings of modern phonology and morphology, the contributions in Part 2 of the book, "Knowledge-based methods", present computational treatments built on solid linguistic grounds. This part of the book brings together aspects of phonology, morphology and computational morphology with the aim of providing a linguistically tractable account of Arabic morphology. The following four major linguistic frameworks are presented in the four chapters of this part:

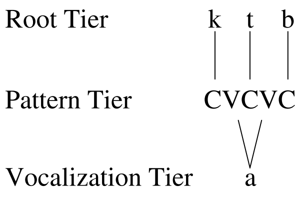
Syllable-based Morphology (SBM): In SBM, morphological realisations are defined in terms of their syllable structure. Although most work in syllable-based morphology has addressed European languages (especially the Germanic languages) the theory was always intended to apply to all languages. One of the language groups that appears on the surface to offer the biggest challenge to this theory is the Semitic language group. In Chapter 4, Cahill presents a syllable-based analysis of Arabic morphology which demonstrates that, not only is such an analysis possible for Semitic languages, but the resulting analysis is not significantly different from syllable-based analyses of European languages such as English and German. While the account presented in this chapter does not require the separation of morphemes à la McCarthy as is described in the previous chapter, the organisation of the lexicon reflects this separation: information about the pattern[1], root and vowel inflections is provided by separate nodes. Interestingly, this chapter also captures the dependencies between the Arabic binyanim forms with only a few equations using DATR's inheritance techniques.

Root-and-Pattern Morphology: The type of account of Arabic morphology that is generally accepted by (computational) linguists is that proposed by McCarthy (1979, 1981). In his proposal, stems are formed by a derivational combination of a root morpheme and a vowel melody. The two are arranged according to canonical patterns. Roots are said to interdigitate with patterns to form stems. McCarthy's analysis differs from Harris' (1941) in abstracting out or autosegmentalizing the vowels from the pattern and placing them on a separate tier of the analysis. Rules of association then match consonants with C slots and vowels with V slots to form the abstract stem.

Harris' segmental analysis consisted of:

Root:	k t b	"notion of writing"
Pattern	_a_a_	
Stem	katab	"wrote"

McCarthy (ibid) autosegmentalizes the vowels from the pattern, as is shown below:



Note that the difference between the segmental analysis and the autosegmental analysis is not just in the notation. The autosegmental approach is introduced to capture some linguistic phenomena in Arabic, such as Spreading, a process that involves consonant copying over intervening phonemes.

McCarthy's autosegmental approach is reflected in most of the computational attempts to model Arabic morphology, especially in the systems written within finite-state morphology (Beesley, 1990, 1996; Kay, 1987; Kiraz, 1994, 2000). Since, to our knowledge, no recent attempts have been made to improve the results of already published work on Arabic Finite-state morphology, there is no contribution in this book within this framework. However, it would be useful to briefly review one of the largest systems ever built for Arabic morphology on the basis of finite-state technology, namely the Arabic morphology system implemented using Xerox finite-state technology.

Using Xerox lexical and rule compilers, Beesley (1998) argues that the interdigitation of semitic roots and patterns is simply an intersection process. It is argued that triliteral roots are represented as $?^*C?^*C?^*C?^*$, where * denotes zero or more

[1] Also called measure or binyan (singular of binyanim).

concatenations of ? (=any symbol) and C represents any consonant. The root *drs* "the notion of studying", for example, can be represented as $?^{*}d?^{*}r?^{*}s?^{*}$. The intersection of this representation with the pattern *CaCaC* produces *daras* "studied":

(1) [drs & CaCaC] (where the square brackets are symbols that delimit the stem and the symbol & denotes the intersection of *drs* and C*a*C*a*C.)
Abstract intersected level = *daras*

The voweling of the pattern can also be abstracted:

(2) Abstract lexical level: [drs & CVCVC] [a]
Abstract intersected level: *daras*

For each root and pattern, a rule of the form in (1) is generated automatically. The generated rule is compiled into its corresponding transducer.[2] The latter maps the string *[drs & CaCaC]* into *daras*.

In order to develop these ideas further, let us consider how hollow verbs are treated. Hollow verbs have a weak middle radical (cf. Chapter 6 for details). In Beesley's system, *qwl* "the notion of saying" is the underlying spelling of the root. This is what appears in the root dictionary. As for all roots, the dictionary lists all the forms that the root can take. For form 1, it is also necessary to specify the stem vowels. Thus, in the Xerox system, the underlying form 1 perfective pattern for *qwl* is *CaCuC*. The underlying interdigitated stem is therefore *qawul*. The third person masculine singular suffix *-a* is added, and the underlying form qawul+a is yielded:

(3) Levels of derivation of *qaAla* "he said"
Upper level: *[qwl & CaCuC] + a*
Intermediate level: *qawul+a*
Fully voweled: *qaAla*

The dictionary is first compiled into a finite-state transducer that recognizes strings like *[qwl & CaCuC]+a*. An algorithm then interdigitates the root and pattern into *qawul* and creates a transducer that maps between *[qwl & CaCuC]+a* and *qawul+a*. The mapping from the intermediate level to the fully voweled level is performed by alternation rules that map *qawul+a* to *qaAla*. These alternation rules are rather complex, especially for the handling of *w* and *y*, but they are normal finite-state alternation rules. All the rules are compiled into a transducer, as well as the lexicon. The two are then combined via a finite-state operation called composition (denoted .o. in regular expressions). In the case at hand, *w* is deleted in the surface word and the vowel *a* is lengthened (and spelled with Alif *A*).

[2] A finite transducer is like an ordinary finite-state automaton except that it considers two strings rather than one. In a transducer, the arc labels are symbol pairs having the form x:y. The first member of the pair (the upper level), x, belongs to the input string, and the second symbol (the lower symbol), y, is part of the output string. If the members are identical, the pair is written as a single symbol.

The result, after composition of the rules, is a single two-level finite-state transducer that maps directly between strings like the following:

(4) Upper level: *[qwl & CaCuC] + a*
Fully voweled: *qaAla*

The intermediate level disappears in the composition. Thus, if we look up *qaAla*, we get *[qwl & CaCuC]+a*, and if we generate from *[qwl & CaCuC]+a*, we get *qaAla*. Generation is just the reverse of analysis. These transducers are inherently bi-directional. Bi-directionality is maintained by a direct mapping of each root and pattern pair to their respective surface realizations.

In Beesley's system, the intersection mechanism requires the application of a rule conveying a linguistic phenomenon to every single stem of the language. According to Kiraz' (2000) computational evaluation of the lexical compilation of Beesley's system, the intersection approach needs m rules of the form in (1) above (i.e., m intersections) to be compiled into their respective transducers (where $r << m < (r \times v \times p)$, r=roots, v=vocalisms, p=patterns). That is, m is far greater than r, but less than r times v times p.

Another problem is that a full recompilation is required for new dictionary entries. Beesley (personal communication) pointed out that, in the Arabic system (and most others at Xerox), there is a premium placed on fast runtime performance.

Although the existing finite-state accounts have tried to show that finite-state and two-level morphology techniques are adequate for Arabic morphology, they fall short of capturing linguistic generalizations. There are two interesting points not dealt with in these models. The first point relates to the syncretism cases exhibited in the Arabic verbal and noun systems.[3] Chapters 5 and 6 of this book provide linguistic evidence that an adequate theory of morphology must incorporate rules of referral in order to account for some kinds of inflectional syncretism.[4] (cf. these chapters for further details). The second point relates to the dependencies between the Arabic verbal patterns. That is some patterns can be derived from other patterns (see Chapters 4 and 5).

Adopting McCarthy's multilinear formalization of Arabic morphology, Al-Najem, introduces an inheritance-based approach that computationally captures the generalizations, dependencies and syncretisms existing in Arabic morphology in Chapter 5. For this, Al-Najem uses the lexical knowledge representation language DATR which enables the definition of inheritance networks in a relatively simple way.[5]

Lexeme-based Morphology (LBM): LBM supports the claim that the stem is the only morphologically relevant form of a lexeme. Chapter 6, by Cavalli-Sforza and Soudi, provides linguistic evidence that the stem is the phonological domain of realization

[3] *Syncretism* refers to any instance of what (Carstairs 1987:91) calls "systematic inflectional homonymy".

[4] A *referral* is defined as the stipulation "that certain combinations of features have the same realization as certain others" (Zwicky 1985:372).

[5] http://www.cogs.susx.ac.uk/lab/nlp/datr/datr.html

rules. It is demonstrated that this claim is appropriate and necessary for capturing the linguistic facts in a computational account. On the one hand, the authors have done this by showing the advantages Lexeme-based Morphology has over the Lexical Morpheme Hypothesis. On the other hand, the authors have provided a computational implementation that allows them to test the approach with a non-fragmented lexicon (i.e., without sub-lexicons: a sub-lexicon for vocalism, a sub-lexicon for roots and another sub-lexicon for patterns). In this approach, the stem and operations on the stem become the focus of the representation. Thus, the approach differs from the previous computational representations of Arabic morphology that have essentially granted equal status to all the constituents of an Arabic word (the root, the pattern and the vocalism) by placing them in separate lexicons.

Stem-based Arabic Lexicon with Grammar and Lexis Specifications: The central claim of this approach is that stem-grounded lexical databases, with entries associated with grammar and lexis specifications, is the most appropriate organisation for the storage of pertinent information for Arabic. In Chapter 7, Dichy and Farghaly provide an in-depth discussion of the role of grammar-lexis relations in the computational morphology of Arabic. After presenting the limits of the pattern and root representation as well as other approaches to Arabic morphology, the authors argue that entries associated with a finite set of morphosyntactic w-specifiers can guarantee a complete coverage of data within the boundaries of the word-form. The contents of this chapter are based on two experiences in Arabic NLP development, that of the DIINAR.1 "*DIctionnaire Informatisé de l'Arabe*", a comprehensive Arabic lexical resource of around 121.000 lemma-entries and that of the lexical database and analyzers embedded in the SYSTRAN Arabic-English translator, a fully automatic transfer system (Dichy et al., 2002).

1.3 Empirical Approaches

A major benefit of the knowledge-based methods is that the rules and constraints for recognizing and classifying the internal structure of words are defined on a precise linguistic basis. Thus, under the assumption that the set of morphological rules and constraints define a linguistically consistent system, then, if a word can be morphologically analysed, we can be sure that the resulting structure is correct. Of course, this requires the computational basis of the analysis to be sound and complete; however, since we also assume this for the computational basis of the empirical methods, we do not consider this as a unique feature of knowledge-based methods. Furthermore, it is also often assumed that the modelled linguistic system is domain independent that is valid and applicable in any domain. As a consequence, the ultimate goal is to implement a linguistic knowledge base -in our case, a morphological system- that covers all possible allowable structures and only these. However, this requires not only that all possible allowable structures are known by the linguist, but that they can be formalized and implemented consistently preferable as non-redundant as possible.

As everybody who has implemented large-scale real-life NLP components might have experienced, such a strict perspective has at least the following drawbacks:

- Ambiguity: The aim of formalizing all possible allowable structures means that an NLP component is to be expected to return all possible analyses (or readings) for a given input for further processing (by a human or another NLP component), as long as the system does not dispose of any decision criteria on how to rank or select between the alternative readings relative to a given domain or application.
- Coverage: Even if a linguistic domain is completely understood, it is extremely challenging to provide a complete implementation of all phenomena from scratch, because it might still be unclear how to represent a certain linguistic entity using the implementation formalism at hand or because not all possible ways and constraints (e.g., about the nature of the input data) are known in order to properly embed the NLP component into a larger application context.

As a solution direction for these kinds of problems, empirical-based methods are explored and developed in computational linguistics since the 1990s, cf. Cardie and Mooney (1999). Empirical methods employ machine learning techniques to automatically extract linguistic knowledge from natural language data directly rather than require the system developer to manually encode the requisite knowledge. Since these methods are by definition data-driven, they actually also learn how to weight between alternative solutions and how to predict useful information (e.g., a missing class label) for unknown entities through a rigorous statistically analysis of the data. In the beginning of the development of the new field of empirical methods for NLP, the proposed approaches have often been considered as alternative or even competing approaches to the corresponding knowledge-based methods, cf. Magerman (1995). However, in recent years the trend has become to consider both approaches more as complementary to each other, and new ways of integrating knowledge-based and empirical-methods are envisaged and actively explored.

Part 3 of the book, "Empirical methods", presents four accounts of data-driven processing models of Arabic morphology. The contributions reflect key advances in the field of machine learning and statistical models applied to natural language.

The Part's first chapter, Chapter 8, by Daya, Roth, and Wintner, acts as a bridge to the second part, Knowledge-based methods, by addressing the question whether the performance of machine-learning-based models of morphology can be boosted by constraining them according to externally coded linguistic knowledge. As argued in Chapter 8, this is indeed the case. The task studied in Chapter 8 is the identification of the root of a wordform; a hard task central to morphological analysis. The chapter describes experiments performed with SNoW (Carlson et al., 1999), based on the Winnow classifier (Littlestone, 1988). In this chapter another bridge is made; results obtained on Arabic are combined with results obtained on Hebrew.

Subsequently, In Chapter 9, by Diab, Hacioglu, and Jurafsky, a completely machine-learning-based account of Arabic morpho-syntax is presented using support vector machines (Cortes and Vapnik, 1995). In this approach, morphology is seen as an integral part of the larger problem of part-of-speech tagging and constituent chunking. Rather than full morphological analysis, the authors focus on clitic segmentation, while a subsequent part-of-speech tagger assigns detailed morpho-syntactic tags to the segmented token sequences. Using the recent annotated data that have become available, i.e. the Arabic Treebank, via the Linguistic Data Consortium, this chapter sets a standard in an integrative machine-learning approach to shallow Arabic parsing.

Chapters 10, by Clark, and 11, by Van den Bosch, Marsi, and Soudi, both present memory-based models, the one in Chapter 10 being essentially unsupervised, and the one in Chapter 11 supervised, drawing on the same Arabic Treebank data as used in Chapter 9. Compared to the other chapters in this part, in Chapter 10 Clark provides a counterpoint in showing the potential strength of unsupervised machine learning techniques, i.e., techniques for which un-annotated training corpora suffice. Clark shows how stochastic transducers, trained with the unsupervised expectation-maximization algorithm (Dempster, Laird, and Rubin, 1977) can be trained to map base forms to inflected forms, even if the examples are not paired for particular words, but are merely collected into two unordered sets of base forms and inflected forms.

In Chapter 11, Van den Bosch, Marsi and Soudi use a similar but supervised memory-based approach (Daelemans and Van den Bosch, 2005), which as in Chapter 9 integrates the task of morphological analysis with part-of-speech tagging. Rather than in Chapter 9, which focuses on clitic segmentation, Arabic morphological analysis is defined as encompassing the segmentation and tagging of all morphemes in a wordform, including spelling changes. This is then formulated as a lattice generation task, which the memory-based classification approach generates in overlapping fragments. Analysis, in other words, is reduced to sequences of classifications that each encode character-by-character segmentation codes, part-of-speech information, and spelling information; post-processing is subsequently needed to construct a lattice that ideally does not overgenerate nor undergenerate too much. Finally, the authors show that their morphological analysis module could be integrated into a part-of-speech tagger.

The issues played out in Part 3 are, in sum:

(i) The integration and co-learnability of morphological analysis with part-of-speech tagging and shallow parsing (constituent chunking).
(ii) The limitations of single-step morphological analysis "by classification".
(iii) The relation of analysis, which is easily rephrased as a classification task learnable by machine learning algorithms, to generation, which is much harder to model in the standard machine learning representation frameworks.
(iv) The role of linguistic knowledge in features or constraints used in the supervised learning of morphological analysis.
(v) The transferability of results attained with machine-learning algorithms between models trained on Hebrew and on Arabic.

1.4 Applications of Arabic Computational Morphology

In the two parts 2 and 3, morphological processing has been considered mainly from a strict isolated or modular point of perspective. In the final part 4, "Integration of Arabic morphology into larger applications", morphological processing is mainly considered as a sub-component of a large-scale end-to-end NLP system, viz. systems for performing Information Retrieval (IR) or Machine Translations (MT). Under such a system—integration perspective, it has to be carefully exploited how effectively a morphological component – which might have been mainly developed with a generic "plug-and-play" design goal – can be embedded into such a larger application environment. In particular, the specific input and output requirements of the application dictate not only constraints on the representation of structure, but also on the needed depth of the structural analysis. For example, in a full—text search engine, the main task of the morphological component might be the part-of-speech tagging and a stem-based segmentation, whereby in the context of a MT system, additionally morphosyntactic information has to be computed in order to support the parsing and generation engines. Furthermore, the interplay with other components which are applied on the same input level has to be considered carefully. Consider, for example, the case of Named Entity recognition (NER), i.e., the classification of token sequences as a person name, a location name, a date expression, etc. Here, it depends on the larger application environment, whether NER is seen as a pre-processor to morphology, as a post-processor or as mutually independent processes. However, this has a direct influence on the scalability and robustness of both components. For example, if NER acts as a pre-processor, then its performance will doubtless influence the performance of the morphology.

The chapters in Part 4 focus on aspects of Arabic morphology as an integral part of IR and MT. Recently, further large-scale applications are in the focus of attention, e.g., cross-lingual open-domain question answering (cf. Al-Maskari and Sanderson, 2006; Awadallah and Rauber, 2006; Hammo et al., 2002) and information extraction (cf. Abuleil, 2004; Florian et al., 2004; Maloney and Niv, 1998), but also the recently launched pilot evaluation for Entity translation as part of the ACE (Automatic Content Extraction) program.[6]

Part 4, "Integration of Arabic morphology into larger applications", addresses key issues relevant to the deployment of Arabic morphological information in information retrieval and machine translation. One of the salient issues discussed (and empirically tested through evaluation) is whether a root-based or a stem-based approach to Arabic morphology would allow effective information retrieval. Chapter 12 demonstrates that light stemming allows remarkably good information retrieval without providing correct morphological analyses. In this context, the authors have addressed the question as to why, given the complexity of Arabic morphology, a morphological analyzer does not perform better than a simple stemmer. It might be the case that for future content-based retrieval systems

[6] http://www.nist.gov/speech/tests/ace/index.htm.

morphological components are at least as important as simpler stemmers in order to support semantic information annotation and access.

Chapter 13 presents a rapid method of developing a shallow statistical Arabic morphological analyzer. The analyzer is concerned with generating possible roots and stems of a given Arabic word along with a probability estimate of deriving the word from each of the possible roots. The use of the generated roots and stems along with their probability estimates as index terms is evaluated in an information retrieval application and the results are compared to index terms generated from a rule-based Arabic morphology tool and an Arabic light stemmer.

With respect to the employment of Arabic morphology in MT, two key points are addressed:

(i) Multiple representations of morphological information in resources for MT: Arabic resources (morphological systems, dictionaries, etc.).often use various morphological representations (e.g., lexeme, stem, root) that are not necessarily compatible with each other, hence the need for machine translation researchers to relate Arabic resources in differing morphological representations. Chapter 14 describes the different representations used by many resources and their usability in different machine translation approaches (symbolic, statistical and hybrids) for Arabic as source language and as target language. A framework addressing this compatibility issue in a specific hybrid MT system is discussed in detail. In this context, the lexeme-and-feature level of representation is motivated.

(ii) The role of Arabic morphology generation in MT: Chapter 15 investigates the impact of Arabic Morphological Generation on the quality of English to Arabic Machine Translation. To this end, the authors have translated thousands of sentences from English to Arabic using an online MT system and have categorized the main morphological information/errors relevant to Arabic MT into various types of features. A detailed scrutiny of the translated sentences, with a focus on morphological information/errors, has revealed that the morphological information captures various linguistic aspects (syntactic, pragmatic, etc.) and hence affects quite heavily the quality of the translation.

1.5 A BLARK for Arabic

Data-driven processing models of Arabic morphology as well as Natural Language processing applications involving Arabic rely heavily on the existence of Language Resources (LRs). During a recent EU project on Arabic language resources, NEMLAR (Network for Euro-Mediterranean Language Resources), two surveys on the availability of Arabic LRs and on industrial requirements were carried out. Several tools and LRs have been identified.[7] Interestingly, the project also worked

[7] http://www.nemlar.org.

out a BLARK (Basic LAnguage Resource Kit) for Arabic.[8] In this context, three important issues are discussed:

(i) *Availability*: the availability of LRs depends on three key factors: (1) Accessibility (an LR that is existent but only company-internal, an LR that is existent and freely usable for precompetitive research, and an LR that is existent and freely usable for both precompetitive research and product development; (2) affordability (resources at a very high cost should not be listed as fully available); (3) customizability (the degree of manipulability of resources).
(ii) *Quality*: the soundness, standard-compliance and interoperability of the LR with other LRs.
(iii) *Quantity*: what counts as a sufficiently large LR (lexicon, corpus etc.).

Based on this BLARK, several LRs have been developed.[9]

References

Abuleil, S. (2004). Extracting names from Arabic text for question-answering systems. In *Proceedings of RIAO'2004*, pp. 638–647, France, 2004.

Al-Maskari, A. and Sanderson, M. (2006). The affect of machine translation on the performance of Arabic-English QA system. In *Proceedings of the EACL'2006 Workshop on Multilingual Question Answering*, MLQA06, pp. 9–14, Trento, Italy.

Awadallah, R., and Rauber, A. (2006). Web-based multiple choice question answering for English and Arabic questions. In *Proceedings of the 28th European Conference on Information Retrieval*, ECIR 2006, pp. 515–518, London, UK.

Beesley, K. (1990). Finite-state description of Arabic morphology. In *Proceedings of the Second Cambridge Conference: Bilingual Computing in Arabic and English.*

Beesley, K. (1996). Arabic finite-State morphological analysis and generation. In *Proceedings COLING'96*, Vol. **1**, pp. 89–94.

Beesley, K. (1998). Consonant spreading in Arabic stems. In *Proceedings of COLING'98*, pp. 117–123.

Cardie, C., and Mooney, R. (1999). Guest editors' introduction: Machine learning and natural language. *Machine Learning*, **11:1–3**, pp. 1–5, 1999.

Carlson, A., Cumby, C., Rosen, J., and Roth, D. (1999). *SNoW user guide.* Technical Report UIUCDCS-R-99-2101, Cognitive Computation Group, Computer Science Department, University of Illinois.

Carstairs, A. (1987). *Allomorphy in Inflexion*. Groom Helm, London.

Cortes, C., and Vapnik, V. (1995). Support vector networks. *Machine Learning*, **20**, pp. 273–297.

Daelemans, W., and Van den Bosch, A. (2005). *Memory-based Language Processing*. Cambridge, UK: Cambridge University Press.

[8] The BLARK initiative was initially launched in the Netherlands with the aim of setting up an organized infrastructure for Dutch-based Human Language Technology.

[9] The BLARK for Arabic as well as the tools and the resources that have been identified and developed are available at the NEMLAR website.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological),* **39:1**, pp. 1–38.

Dichy, J., Braham, A., Ghazali, S., and Hassoun, A. (2002). La base de connaissances linguistiques DIINAR.1 (DIctionnaire INformatisé de l'ARabe, version 1). In A. Braham (Ed.), *Proceedings of the International Symposium on The Processing of Arabic* (April 18–20, 2002). Université de la Manouba, Tunis.

Florian, R., Hassan, H., Ittycheriah, A., Jing, H., Kambhatla, N., Luo, X., Nicolov, N., and Roukos, S. (2004). A statistical model for multilingual entity detection and tracking. In *Proceedings of NAACL/HLT-2004*, pp. 1–8, Boston, MA, USA.

Hammo, B., Abu-Salem, H., Lytinen, S., and Evens, M. (2002) QARAB: A question answering system to support the Arabic language. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, pp. 55–65.

Harris, Z.S. (1941). The Linguistic Structure of Hebrew. In *Journal of the American Oriental Society*, **62**, pp. 143–67.

Kay, M. (1987). Non-concatenative finite-state morphology. In *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark, pp. 2–10.

Kiraz, G. (1994). Multi-tape two-level morphology: A case study in semitic non-linear Morphology. In *Proceedings of COLING'94*, Vol. **1**, pp. 180–186.

Kiraz, G. (2000). A Multi-tiered Nonlinear Morphology using Multi-tape Finite State Automata: A Case Study on Syriac and Arabic. In *Computational Linguistics*, **26:1**, pp. 77–105.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, **2**, pp. 285–318.

McCarthy, J.A. (1979). *Formal Problems in Semitic Phonology and Morphology*. Doctoral Dissertation, MIT.

McCarthy, J.A. (1981). Prosodic Theory of Non-Concatenative Morphology." In *Linguistic Inquiry*, **12**, pp. 373–418.

Magerman, D.M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 43rdAnnual Meeting of the Association for Computational Linguistics*, ACL-95. pp. 276–283. ACL: Ann Arbor, MI, USA.

Maloney, J., and Niv, M. (1998). TAGARAB: A fast, accurate Arabic name recognizer. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pp. 8–15. Montreal, Canada.

Zwicky, A. (1985). How to Describe Inflection. In *Berkeley Linguistic Society*, pp. 372–386.

2

On Arabic Transliteration

Nizar Habash[1], Abdelhadi Soudi[2] and Timothy Buckwalter[3]

[1] *Center for Computational Learning Systems, Columbia University, United States*
habash@cs.columbia.edu

[2] *Ecole Nationale de l'Industrie Minérale, Morocco*
asoudi@gmail.com

[3] *Linguistic Data Consortium, University of Pennsylvania, United States*
timbuck2@ldc.upenn.edu

Abstract: This chapter introduces the transliteration scheme used to represent Arabic characters in this book. The scheme is a one-to-one transliteration of the Arabic script that is complete, easy to read, and consistent with Arabic computer encodings. We present guidelines for Arabic pronunciation using this transliteration scheme and discuss various idiosyncrasies of Arabic orthography

2.1 Introduction

In this chapter, we introduce the transliteration scheme used in this book to represent Arabic words for readers who cannot read the Arabic script. We follow the definition of the terms *transcription* and *transliteration* given by Beesley (1998): the term *transcription* denotes an orthography that characterizes the phonology or morpho-phonology of a language, whereas the term *transliteration* denotes an orthography using carefully substituted orthographical symbols in a one-to-one, fully reversible mapping with that language's customary orthography. This specific definition of transliteration is sometimes called a "strict transliteration" or "orthographical transliteration" (Beesley, 1998).[1]

For Arabic script, as used in writing Modern Standard Arabic, there are many ways to define the orthographic symbol set. The basic Arabic alphabet has 28 letters and eight diacritical marks. However, there are eight additional symbols that can be treated as separate letters and/or special combinations of letter and additional diacritics. One example is the Hamza (همزة hamzaћ) which can be a separate

[1] We do not consider non-one-to-one schemes because of their potential to add ambiguity (Beesley, 1998) or become excessively cumbersome to deal with, e.g., SATTS transliteration ("Standard Arabic Technical Transliteration System," 2006).

letter (ء) or can combine with other letters: أ, ؤ, ئ. As a result, it is possible to define an orthographic symbol set of Arabic where the Hamza is not just a letter but also a diacritic with a limited number of combinations. In fact, standard computer encodings of Arabic, such as CP1256, ISO-8859, and Unicode[2], do not do that. They all consider the additional eight symbols as separate letters.[3]

The Buckwalter Arabic transliteration (Buckwalter, 2001) is a transliteration scheme that follows the standard encoding choices made for representing Arabic characters for computers. The Buckwalter transliteration has been used in many publications in natural language processing and in resources developed at the Linguistic Data Consortium (LDC). The main advantages of the Buckwalter transliteration are that it is a strict transliteration (i.e., one-to-one) and that it is written in ASCII characters. However, the Buckwalter transliteration is not always intuitively easy to read. We address this problem in our transliteration scheme by extending the Buckwalter transliteration scheme to include non-ASCII characters whose pronunciation is easier to remember. For example, instead of Buckwalter's non-intuitive * for ذ /ð/, v for ث /θ/ and K for the diacritic ٍ /in/, we use ð, θ, and ĩ, respectively. Buckwalter transliteration choices that *imitate* Arabic pronunciations are kept unchanged, e.g., b for ب /b/ and k for ك /k/. Since non-ASCII characters are less accessible through standard American keyboards (as opposed to ASCII characters), this is clearly a trade-off between typing/coding simplicity and ease of readability.

To our knowledge, there has been no earlier attempt to create a one-to-one transliteration of the Arabic script that is complete and easy-to-read, and that is consistent with Arabic computer encodings. Almost all of the previously created schemes to represent Arabic characters for western readers have focused on representing phonology and morphology (transcription) or mixing between phonology and orthography, making exceptions for transliteration of morphemes such as the definite article ("Arabic Transliteration," 2006; Buckwalter, 2001). Two transliteration schemes, ISO 233 ("ISO 233," 2006) and El-Dahdah's transliteration (El-Dahdah, 1992), get close to achieving our goal except that they are not consistent with computer encodings. For example, in ISO 233, the Hamza (همزة hamzaħ) is treated as a diacritic that combines with other characters whereas all standard encodings of Arabic treat it as an inseparable letter part.

In the Section 2.2 we introduce our transliteration scheme for Arabic script as used in Modern Standard Arabic. In Section 2.3 we present guidelines for pronunciation of Arabic using this transliteration scheme and address various idiosyncrasies of Arabic orthography.

---

[2] Unicode actually implements both approaches, but the use of Hamza as a diacritic in Unicode is not that common to our knowledge.

[3] Arabic letters also have multiple shapes (allographs) that fully depend on their position in the word, e.g. the letter ع has the forms -ع, -ع–, -ع, and ع in its initial, middle, final and standalone positions, respectively. There are also additional ligatures such as لا for ا+ل. We do not discuss the possibility of defining an orthographic symbol set that considers allographs and ligatures as base symbols since Arabic's simple graphotactic rules are abstracted away in all of its standard encodings. Considering sub-letter dots in Arabic as separate symbols is also not discussed for the same reason.

2.2 This Book's Arabic Transliteration Scheme

Table 2.1 provides the full list of Arabic transliterations used in this book. The first three columns contain the symbols for the characters in Arabic script and contrast our transliteration with the Buckwalter transliteration. Cases where our transliteration differs from Buckwalter's are highlighted. The last four columns

Table 2.1. This book's Arabic transliteration scheme with examples

Characters			Examples			
Arabic	Transliteration	Buckwalter	Arabic	Transliteration	Transcription	Gloss
ء	'	'	سماء	samaA'	/samā'/	*sky*
آ	Ā	|	آمن	Āmana	/'āmana/	*he believed*
أ	Â	>	سَأل	saÂala	/sa'ala/	*he asked*
ؤ	ŵ	&	مؤتمر	muŵtamar	/mu'tamar/	*conference*
إ	Ǎ	<	إنترنت	Ǎintarnit	/'intarnit/	*internet*
ئ	ŷ	}	سائل	saAŷil	/sā'il/	*liquid*
ا	A	A	كان	kaAna	/kāna/	*he was*
ب	b	b	بريد	bariyd	/barīd/	*mail*
ة	ħ	p	مكتبة	maktabaħ maktabaħũ	/maktaba/ /maktabatun/	*Library a library* [nom.]
ت	t	t	تنافس	tanaAfus	/tanāfus/	*competition*
ث	θ	v	ثلاثة	θalaAθaħ	/θalāθa/	*three*
ج	j	j	جميل	jamiyl	/jamīl/	*beautiful*
ح	H	H	حاد	HaAd~	/Hādd/	*sharp*
خ	x	x	خوذة	xuwðaħ	/xuwða/	*helmet*
د	d	d	دليل	daliyl	/dalīl/	*guide*
ذ	ð	*	ذهب	ðahab	/ðahab/	*gold*
ر	r	r	رفيع	rafiyς	/rafīς/	*thin*
ز	z	z	زينة	ziynaħ	/zīna/	*decoration*
س	s	s	سماء	samaA'	/samā'/	*sky*
ش	š	$	شريف	šariyf	/šarīf/	*honest*
ص	S	S	صوت	Sawt	/Sawt/	*sound*
ض	D	D	ضرير	Dariyr	/Darīr/	*blind*
ط	T	T	طويل	Tawiyl	/Tawīl/	*tall*
ظ	Ď	Z	ظلم	Ďulm	/Ďulm/	*injustice*
ع	ς	E	عمل	ςamal	/ςamal/	*work*
غ	γ	g	غريب	γariyb	/γarīb/	*strange*
ف	f	f	فيلم	fiylm	/fīlm/	*movie*
ق	q	q	قادر	qaAdir	/qādir/	*capable*
ك	k	k	كريم	kariym	/karīm/	*generous*
ل	l	l	لذيذ	laðiyð	/laðīð/	*delicious*
م	m	m	مدير	mudiyr	/mudīr/	*manager*
ن	n	n	نور	nuwr	/nūr/	*light*
ه	h	h	هول	hawl	/hawl/	*devastation*
و	w	w	وصل	waSl	/waSl/	*receipt*

(*Continued*)

Table 2.1. (*Continued*)

Characters			Examples			
Arabic	Transliteration	Buckwalter	Arabic	Transliteration	Transcription	Gloss
ى	ý	Y	على	ςalaý	/ςala/	*on*
ي	y	y	تين	tiyn	/tīn/	*figs*
ـَ	a	a	دَهَنَ	dahana	/dahana/	*he painted*
ـُ	u	u	دُهِنَ	duhina	/duhina/	*it was painted*
ـِ	i	i	دُهِنَ	duhina	/duhina/	*it was painted*
ـً	ã	F	كتاباً	kitaAbAã	/kitāban/	*a book* [nom.]
ـٌ	ũ	N	كتابٌ	kitaAbũ	/kitābun/	*a book* [acc.]
ـٍ	ĩ	K	كتابٍ	kitaAbĩ	/kitābin/	*a book* [gen.]
ـّ †	~	~	كَسَّرَ	kas~ara	/kassara/	*he smashed*
ـْ ‡	.	o	مَسْجِد	mas.jid *or* masjid	/masjid/	*mosque*
ـ §	_	_	مَســـْجِد	mas._jid	/masjid/	*mosque*

† Shadda (شدة šad~aħ) is a symbol marking consonant doubling.

‡ Sukun (سكون sukuwn) is a symbol marking lack of vowel. It can be used for contrastive purposes in the transliteration. However, it is not required in this book for the purpose of improving readability.

§ Tatweel (تطويل taTwiyl) or Kashida (كشيدة kašiydaħ) is an orthographic elongation symbol with no phonetic value.

present English-glossed examples in Arabic script, our transliteration, and a phonological transcription. Since Arabic words can be written fully diacritized, partially diacritized or non-diacritized, a transliteration should preserve fully how an Arabic word is constructed. This includes preserving all possible ambiguities. For example, the Arabic word كتب ktb could represent one of many diacritized words with different meanings and pronunciations: the noun كُتُب kutub '*books*' or the verb كَتَبَ katab '*he wrote*' among others. Of course, most naturally occurring Arabic text is not diacritized; however, in this book, diacritized transliterations are always used for readability unless the point is to discuss diacritization ambiguity. In Table 2.1, we show examples in fully diacritized transliteration to contrast with the corresponding transcriptions, but the Arabic text examples are not fully diacritized.[4]

[4] In this book, there are very few cases that slightly deviate from our transliteration scheme:

(i) A special transcription variant is used where the one-to-one transliteration of the Arabic script interferes with the author's explanation of some linguistic phenomena. For example, in Chapter 4 "A Syllable-based Account of Arabic Morphology, the author represents vowel lengthening and gemination by vowel doubling and consonant doubling, respectively. The use of transliteration to represent these phenomena would interfere with the syllabification process. In some cases, the authors use a phonetic transcription.

(ii) Snapshots from LDC resources that use the Buckwalter transliteration are presented in the Buckwalter transliteration. This is done in some of the chapters in the third, empirical part of the book.

2.3 Pronunciation Guidelines

Arabic script, as is used in Modern Standard Arabic, is *mostly* a phonemic system with one-to-one mappings of sounds to letters and diacritics. When fully diacritized, Arabic is *almost* perfectly phonologically reproducible by readers given the following few rules and exceptions:

1. **For most *consonants*,** there is no issue in mapping letters to sounds. Some are easier for English speakers than others. The transcription and transliteration are the same for these cases. Table 2.2 describes how to pronounce these consonants.

Table 2.2. Arabic consonant pronunciation

Arabic	Transliteration	Pronunciation	
ب	**b**	Boy	
ت	**t**	Toy	
ث	**θ**	Three	
ج	**j**	Jordan	
ح	**H**	Voiceless pharyngeal fricative. Sounds like a sharp h.	
خ	**x**	Scottish Loch; Yiddish Chutzpa;	
د	**d**	Door	
ذ	**ð**	The	
ر	**r**	Road	
ز	**z**	Zoo	
س	**s**	Sue	
ش	**š**	Shoe	
ص	**S**	Emphatic s	*Emphasis is a bass effect giving an acoustic impression of hollow resonance to the basic sounds* [0].
ض	**D**	Emphatic d	
ط	**T**	Emphatic t	
ظ	**Ď**	Emphatic ð	
ع	**ς**	Voiced pharyngeal fricative. Sounds like a sharp a.	
غ	**γ**	Parisian French r	
ف	**f**	Film	
ق	**q**	Uvular stop. Sounds like a deep k.	
ك	**k**	Kite	
ل	**l**	Cool	
م	**m**	Man	
ن	**n**	Man	
ه	**h**	Hot	
و	**w**	Would	
ي	**y**	Yoke	

2. **The consonant Hamza** (همزة hamzaȟ) has multiple forms in Arabic script. There are complex rules for Hamza spelling that depend on its vocalic context. For a reader, however, all of these forms are pronounced the same way: a glottal stop as in the value of ‘tt’ in the London Cockney pronunciation of bottle. Table 2.3 relates the different forms of Hamza in Arabic script and our transliteration. The form of the transliteration is intended to evoke the form used in the Arabic variant as much as possible. For instance, a circumflex is used with A (ا), w (و) and y (ي) to create their corresponding hamzated forms Â (أ), ŵ (ؤ) and ŷ (ئ).

3. Arabic has three short vowel **diacritics** that are represented using a, u and i. Arabic also has three nunation diacritics. These are short vowels pronounced followed by an /n/. They are not nasalized vowels. Nunation is a mark of nominal indefiniteness in Standard Arabic. Finally, Arabic has a consonant doubling diacritic which repeats previous consonant and also a diacritic for marking when there is no diacritic. Table 2.4 lists these diacritics, their names, and corresponding transliteration and transcription values. Diacritics are largely restricted to religious texts and Arabic language school textbooks. In this respect, the Arabic writing system depends on the background knowledge of the reader to accurately pronounce the written word—much as a reader in English needs to decide on the basis of context whether “read” is pronounced /rīd/ (present tense) or /rɛd/ (past tense).

Table 2.3. Hamza (glottal stop) forms

Arabic	ء	آ	أ	ؤ	إ	ئ
Transliteration	'	Ā	Â	ŵ	Ǎ	ŷ

Table 2.4. Arabic diacritics

Diacritic	Name	Transliteration	Transcription
َ	فتحة fatHaȟ	a	/a/
ُ	ضمة Dam~aȟ	u	/u/
ِ	كسرة kasraȟ	i	/i/
ً	تنوين فتح tanwiyn fatH	ã	/an/
ٌ	تنوين ضم tanwiyn Dam~	ũ	/un/
ٍ	تنوين كسر tanwiyn kasr	ĩ	/in/
ّ	شدة šad~aȟ	~	Double previous consonant
ْ	سكون sukuwn	.	No vowel

4. **Long vowels and diphthongs** in Arabic are indicated by a combination of a short vowel and a consonant. Table 2.5 lists the various Arabic long vowels and diphthongs together with their transliteration and transcription.

5. **The letter Alif** (ا A) is used to (a.) hold vowels at the beginning of words, (b.) represent the long vowel /ā/, and (c.) mark a couple of morphophonemic symbols in which Alif is not pronounced (See note 8).

6. **The /tā' marbūTa/** (تاء مربوطة tA' marbuwTaħ), ة ħ, is typically a feminine ending. It can only appear at the end of a word and can only be followed by a diacritic. In standard Arabic it is always pronounced as /t/ unless it is not followed by a diacritic, in which case it is silent.[5]

7. **The /alif maqSūra/** (ألف مقصورة Âlif maqSuwraħ), ى ý, is a dotless Ya (ي y). In standard Arabic, it is silent and always follows a short vowel **a** at the end of a word. For example, روى rawaý '*to tell a story*' is pronounced /rawa/.[6]

8. There are few **exceptions** to the guidelines above:
   a. The Arabic **definite article**, ال Al /al/, is a prefix that assimilates to the first consonant in the noun it modifies if this consonant is an alveolar or dental sound (except for ج j). This set of letters is called Sun Letters. They include ت t, ث θ, د d, ذ ð, ر r, ز z, س s, ش š, ص S, ض D, ط T, ظ Ď, ل l, and ن n. For example, the word الشمس Alšams '*the sun*' is pronounced /aššams/ not */alšams/. The rest of the consonants are called Moon Letters; the definite article is not assimilated with them. For example, the word القمر Alqamar '*the moon*' is pronounced /alqamar/ not */aqqamar/.
   b. A silent Alif appears in the morpheme +وا +uwA /ū/ which indicates masculine plural conjugation in verbs. Another silent Alif appears after some *nunated* nouns, e.g., كتاباً kitaAbAã /kitāban/. In some poetic readings, this Alif can be produced as the long vowel /ā/: /kitābā/.
   c. A common odd spelling is that of the proper name عمرو çamrw/ çamr/ '*Amr*' where the final w is silent.

Table 2.5. Long vowels and diphthongs

Arabic	ـَا	ـُو	ـِي	ـَو	ـَي
Transliteration	aA	uw	iy	aw	ay
Transcription	/ā/ (long a)	/ū/ (long u)	/ī/ (long i)	/aw/	/ay/

[5] In modern dialects of Arabic, the */tā' marbūTa/* is always silent except when the noun ending with it is part of an (/*'idāfa*/ إضافة ǍiDAfaħ) compound, in which case it is pronounced as /t/.

[6] In some Arab countries such as Egypt, a common orthographic variation is to use ى ý for the letter ي y in word-final position. Orthographic variation in Arabic is further discussed in Chapter 3.

2.4 Conclusion

In this chapter, we presented the transliteration scheme used in the rest of this book. This transliteration is a one-to-one easy-to-read complete transliteration of the Arabic script consistent with Arabic computer encodings. We also presented guidelines for pronouncing Arabic given this transliteration. We hope that this transliteration scheme will become a standard to follow in the natural language processing research community working on Arabic.

Acknowledgements

We would like to thank Ali Farghaly, Joseph Dichy and Owen Rambow for helpful discussions.

References

Arabic Transliteration. (2006, April 6). In *Wikipedia, The Free Encyclopedia.* Retrieved June 19, 2006, from http://en.wikipedia.org/ wiki/Arabic_transliteration

Beesley, K. (1998). *Romanization, Transcription and Transliteration.* Retrieved June 19, 2006, from the Xerox Research Center Europe web site: http://www.xrce.xerox.com/competencies/content-analysis/arabic/ info/romanization.html

Buckwalter, T. (2001). *Arabic Transliteration.* Retrieved June 19, 2006, from http://www.qamus.org/transliteration.htm

El-Dahdah, A. (1992). *Dictionary of Universal Arabic Grammar*. Beirut: Librairie du Liban.

ISO 233. (2006, June 18). In *Wikipedia, The Free Encyclopedia.* Retrieved June 19, 2006, from http://en.wikipedia.org/wiki/ISO_233

Standard Arabic Technical Transliteration System. (2006, April 6). In *Wikipedia, The Free Encyclopedia.* Retrieved June 19, 2006, from http://en.wikipedia.org/wiki/SATTS

3

Issues in Arabic Morphological Analysis

Timothy Buckwalter
Linguistic Data Consortium, University of Pennsylvania
timbuck2@ldc.upenn.edu

Abstract: The salient issues facing contemporary Arabic morphological analysis are summarized as predominantly orthographic in nature, although the issue of how to integrate morphological analysis of the dialects into the existing morphological analysis of Modern Standard Arabic is identified as the primary challenge of the next decade. Issues of orthography that impact morphological analysis stem in part from the successful deployment of the Unicode standard and the subsequent increase in usage of the expanded Arabic character set, including what are properly Persian and Urdu characters. Additional orthographic issues impacting morphological analysis arise from the persistent and widespread variation in the spelling of letters such as *hamza* and *tā' marbūTa*, and the increasing lack of differentiation between word-final *yā'* and *alif maqSūra*. The tokenization of Arabic input strings is also affected by orthography, as typists often neglect to insert a space after words that end with a non-connector letter. An increasing number of archaic morphological features and dated lexical items can be observed in Web-based Islamic publications and cannot be overlooked in contemporary analysis. Finally, the accuracy and completeness of current Arabic morphological analysis can be questioned in light of the almost complete absence of annotation for lexically-determined features of gender, number, and humanness

3.1 Introduction

This chapter is a review of issues in Arabic morphological analysis that have gained prominence in just the last decade as a result of specific worldwide advancements in information science and technology. Among these developments are the successful deployment and widespread use of the Unicode character set, which has greatly extended the set of available characters for representing Arabic electronically, thus impacting the orthography of the language. Thanks to the proliferation of personal computers and the widespread success of Web publishing, Arabic texts are now authored primarily in electronic format and are often published without passing first through the traditional scrutiny of copy editors and other standardizing and normalizing filters. These raw published texts are today readily available for computational analysis, and they reveal various features that are relevant to morphological analysis. The most salient feature is orthographic variation, and much of it derives from purely mechanical factors, such as the manner in which specific letter combinations are displayed on different computer platforms. Other forms of orthographic variation are less artificial and more a reflection of true idiosyncratic and regional spelling tendencies, although the design of specific Arabic glyphs, such as *hamza* in combination with various characters functioning as orthographic props or "chairs," appears to influence typists' habits as well.

The greatest impact that personal computers and Web publishing have had on Arabic morphological analysis today is the change they are bringing about in the language itself. What Hans Wehr (1979) called "modern written Arabic" has long been synonymous with Modern Standard Arabic. Today, however, modern written Arabic includes increasing quantities of dialectal Arabic. A simple Web search of high-frequency dialectal words will yield thousands of Web pages in which dialectal and standard Arabic commingle in written form as they do in spoken form in real life. This emerging modern written Arabic manifests usage ranging from informal to formal and makes appropriate use of both dialectal Arabic and MSA—and some hybrid forms—to reflect changing social registers. Although the orthography of the dialects is far from standardized, increasing popular use and dissemination on the Web are resulting in observable and measurable orthographic conventions. In the discussion that follows we will review in more detail these basic issues that affect Arabic morphological analysis, beginning with issues involving the nature of the input text which impact the pre-processing phase of morphological analysis, such as tokenization and normalization, and concluding with a brief discussion of the nature of the analysis itself, especially the set of lexical and grammatical features in use today. All textual examples that we cite come from actual corpus data.

3.2 Expansion of the Arabic Character Set

The implementation of the Unicode character set on systems used for authoring Arabic texts for publication on the Web has dramatically expanded the set of characters available for representing Arabic electronically, and this has had an impact, both positive and negative, on the orthography of the language, which ultimately impacts morphological analysis as well. The positive impact is seen in more accurate representations of non-Arabic sounds, and in some cases this facilitates disambiguation of what would otherwise be homographs, such as with the word /vān/ (the name "Van" or the type of vehicle commonly called a "van"). This word is now occasionally spelled ڤان VAn (V = ڤ = U+06A4). Normally it would be spelled فان fAn, which would result in an additional possible analysis */fa-'inna/*. The negative impact of easy access to numerous extended Arabic characters in the Unicode character set is seen in new and unpredictable orthographic variation that is linguistically unjustified and hard to detect visually—in fact, much of this variation can only be detected electronically. Before we examine some individual cases of anomalous orthography resulting from the richness of Unicode, we will present some preliminary and basic facts on the Arabic character set.

Table 3.1. ASMO 449 Arabic character set

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
02		ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث	ج	ح	خ	د
03	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ					
04		ف	ق	ك	ل	م	ن	ه	و	ى	ي	ـً	ـٌ	ـٍ	ـَ	ـُ
05	ـِ	ـّ	ـْ													

The basic and minimal character set for representing Arabic textual data in electronic format was defined in the 1980s in the ASMO 449 code page (see Table 3.1), which identified a minimal character set of 36 Arabic letters (0x21-0x3A, 0x41-0x4A), along with a supplementary set of eight short vowels and diacritics (0x4B-0x52) whose usage has always been treated as optional. The fact that these eight short vowels and diacritic needed to be represented with zero-width glyphs may have deterred some early developers from implementing them and dealing with the technical difficulties of displaying them correctly. Even today, successful Web browsers such as Mozilla Firefox will first solve complex aspects of Arabic display, such as bidirectional rendering and multiple character encoding schemes, but will postpone solving the problem of displaying the short vowels and diacritics correctly. These characters are currently displayed in their own dedicated spaces rather than as zero-width glyphs positioned above or below the preceding character (see Figure 3.1).

The eight Arabic short vowels and diacritics have been excluded from input methods designed for portable devices such as mobile phones, which lack space for their keypad display. When cell phones were first Arabized for short text messaging in the late 1990s, the author was assigned the task of designing an Arabic telephone keypad layout for the T9 predictive text input method, whereby tapping the number sequence 5–9–3–8, for example, would automatically spell out the word سلام slAm. (The same numeric sequence spells out the words سكان skAn and صلاة SlAħ, but they are used less frequently than سلام slAm, which is displayed first when using this input method). Although the keypad layout that we proposed clearly showed that there was insufficient room to display the eight short vowels and diacritics (see Figure 3.2), the cell phone manufacturers replied that this was

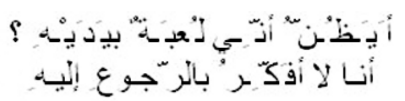

أَيَظُنُّ أَنِّي لُعْبَةٌ بِيَدَيْهِ ؟
أَنَا لَا أُفَكِّرُ بِالرُّجُوعِ إِلَيْهِ

Fig. 3.1. Short vowels and diacritics as displayed in Mozilla Firefox version 1.0

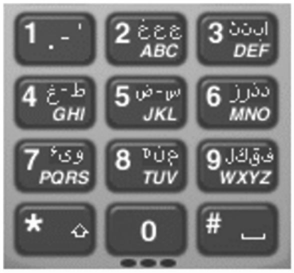


Fig. 3.2. T9 Arabic keypad

not a problem because their consumer research had shown that customers did not need to use short vowels and diacritics. Although it can be debated that the Arabic language can survive without these eight short vowels and diacritics—and character frequency counts of newswire corpora and informal writing show clearly that short vowels and diacritics play only a marginal role in the language—computerization has provided easier access to these characters. Nevertheless, their usage remains associated with specific genres of writing, such as poetry and religious texts, which by their very nature require extensive, if not full vocalization.

Contemporary electronic storage and transmission of Arabic textual data continues to make use of the basic set of 36 characters defined in ASMO 449, although four non-standard Arabic characters were introduced into popular use in the 1990s via platform-specific 8-bit encodings such as Mac Arabic (see Table 3.2) that aimed at providing word processing capabilities for other Arabic-alphabet based languages, chiefly Persian. The four non-standard letters that are occasionally used alongside the standard Arabic character set (see Table 3.3) typically represent sounds that are considered non-native to Arabic, although usage may vary from one region to another in the Arabic-speaking world. For example, whereas in Egypt چ J (U+0686) would represent the non-Egyptian Arabic /j/, as in the name چورچ JwrJ /jūrj/ and the loan word چراچ jrAJ /garāj/, in Iraq the letter چ J (U+0686) would be used to represent the sound /č/, as in the name چلبي Jlby /čalabī/ and the dialectal word شلونچ šlwnJ /šlōnič/. The remaining characters – پ P (U+067E), ڤ V (U+06A4), and گ G (U+06AF) – are used to represent the sounds /p/, /v/, and /g/, respectively. The گ G (U+06AF) is not used in Egypt because /g/ is already the normal pronunciation of ج j (U+062C) in that region of the Arab world.

It is important to note that before these non-standard characters were introduced, Arabic orthography simply made use of their standard counterparts: the letters ب U+0628, ج U+062C, ف U+0641, and ك U+0643 (see Table 3.3). The corresponding non-standard characters پ U+067E, چ U+0686, ڤ U+06A4, and گ U+06AF were adopted not necessarily because the existing orthography was deemed to be inadequate, but simply because the emerging technology made the

Table 3.2. Mac Arabic codepage

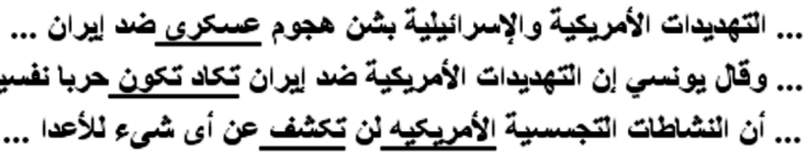
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
80	Ä	NB SP	Ç	É	Ñ	Ö	Ü	á	à	â	ä	ں	«	ç	é	è
90	ê	ë	í	…	î	ï	ñ	ó	»	ô	ö	÷	ú	ù	û	ü
A0	RTL SP	!	"	#	$	٪	&	'	)	(	×	+	،	-	.	/
B0	٠	١	٢	٣	٤	٥	٦	٧	٨	٩	:	؛	>	=	<	؟
C0	*	ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث	ج	ح	خ	د
D0	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	]	\	[	^	_
E0	ـ	ف	ق	ك	ل	م	ن	ه	و	ى	ي	ـً	ـٌ	ـٍ	ـَ	ـُ
F0	ـِ	ـّ	ـْ	پ	ٹ	چ	ە	ڤ	گ	ڈ	ڑ	}	|	{	ژ	ے

Table 3.3. Non-standard Arabic characters

Non-standard character	Standard counterpart
پ P U+067E	ب b U+0628
چ J U+0686	ج j U+062C
ڤ V U+06A4	ف f U+0641
گ G U+06AF	ك k U+0643

new characters more accessible to the common user. A key and obvious factor in determining the use of these characters is the display, or lack thereof, of these extended characters on the text input keyboard itself. It should be noted that although three of these extended Arabic characters (پ U+067E, چ U+0686, and گ U+06AF) are included in the Windows 1256 codepage, they have not been mapped directly to the Arabic keyboard and must be entered via the awkward sequence of Alt + 4-digit number representing their decimal value in the codepage.

Today's Unicode-enabled platforms and word processors have made the entire extended Arabic alphabet character set (U+0600 to U+06FF) available to users, and this has resulted in occasional, and relatively isolated, odd electronic encodings of Arabic text on Internet publications. For example, a character frequency count that we recently conducted using UTF-8 data posted during 2004–2005 on the CNN Arabic website (arabic.cnn.com) showed statistically significant usage of ک U+06A9 (ARABIC LETTER KEHEH), which is used for Persian and Urdu texts. When examining the source data we discovered that this extended character was being used in lieu of ك U+0643 (ARABIC LETTER KAF), and for no apparent reason, because the same text contained as many instances of ordinary KAF as it did of the Persian/Urdu KEHEH. The text data in Figure 3.3 contains both characters, and because the display glyphs are practically identical, the



... التهديدات الأمريكية والإسرائيلية بشن هجوم عسكري ضد إيران ...

... وقال يونسي إن التهديدات الأمريكية ضد إيران تكاد تكون حربا نفسية ...

... أن النشاطات التجسسية الأمريكيه لن تكشف عن أى شىء للأعدا ...

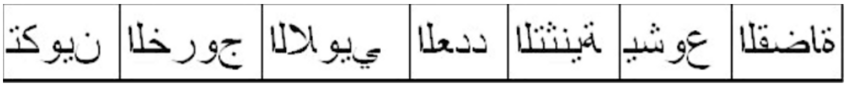
Fig. 3.3. Arabic text with KAF (U+0643) and KEHEH (U+06A9)

underlying encoding difference is hard to detect visually. The words containing KEHEH have been underlined.

Whereas 8-bit encoding formats such as Mac Arabic and Windows 1256 usually ensured that Arabic would be represented with the basic set of 36 characters and eight optional diacritics plus no more than four non-standard Arabic characters, today's multi-byte encoding allows authors and publishers of electronic texts to make direct use of even the very glyphs reserved for rendering on output devises (monitors and printers) and not intended for text storage and text interchange—the so-called Presentation Forms. These glyphs include the initial, medial, final, and separate shapes of each letter for Arabic (U+FE70 – U+FEFF), for Persian, Urdu, and other Arabic-script languages (U+FB50 – U+FBFF), and a large inventory of Arabic ligatures (U+FC00 – U+FDFF).

Arabic Web pages that make use of Arabic presentation forms are quite rare because use of presentation forms requires encoding each line of text in reverse order, which carries with it the assumption that all lines end in carriage return and do not wrap. This so-called "visual" encoding of Arabic, which makes use of display characters or glyphs—as opposed to "logical" encoding, which makes use of abstract characters, not their display glyphs—was used briefly in the early days of Web publishing in Arabic, and has survived in a few legacy Web sites, such as the Al-Ahram mirror site originally created for non-Arabic enabled browsers (www.ahram-eg.com). Although the Al-Ahram example involves visual encoding using a proprietary font and presentation forms, we found at least one example on the Web involving Unicode presentation forms, where the Arabic names of Old Testament books were encoded logically but with presentation forms (see Figure 3.4; observed April 2005 at http://bible.gospelcom.net/versions/).

The easy availability of Unicode extended characters and presentation forms is producing anomalies in Arabic orthography that appear unexpectedly and in a variety of forms that can often be detected only through statistical analyses, such as character frequency counts of electronic texts. These anomalies must be resolved, i.e., corrected or normalized, before the text can be submitted to morphological analysis.



نيوكت	جورخلا	ييولالا	ددعلا	ةينثتلا	عوشي	ةاضقلا

Fig. 3.4. Arabic text encoded logically (left-to-right) with presentation forms

3.3 Orthographic Variation

Orthographic variation poses a challenge to morphological analysis simply because variation in surface orthography expands the inventory of unique character sequences that constitute valid input words. The existence of multiple surface forms for each word creates a need for additional methods for matching surface forms with what is usually a single canonical form in the morphological analysis lexicon. This matching of surface orthography with system-internal canonical forms is done either through orthography-specific two-level morphological rules (Beesley, 2001) or through a combination of pattern matching and generation of orthographic variants (Buckwalter, 2004a). We would like to distinguish two types of Arabic orthographic variation: (1) normal variation, which is caused by the perception among writers and typists of what is orthographically correct or at least permissible, and (2) mechanical variation, which is caused by how specific characters and character combinations are displayed on different computer platforms, which in turn affects a writer or typist's choice of keyboard input characters. We will begin by discussing normal orthographic variation.

Certain types of orthographic variation in Arabic can be observed in all regions and countries where Arabic is written. An example of this variation is the tendency to regard stem-initial *hamza* + *alif* (U+0623, U+0625) and *madda* + *alif* (U+0622) as instances of *alif* + optional diacritics, which means that bare *alif* (U+0627) can substitute for all three (see Table 3.4). In stem-medial and stem-final positions these same characters are treated as optional diacritics only if their removal does not result in ambiguity (see Tables 3.5 and 3.6). The orthographic

Table 3.4. *hamza* + *alif* (U+0623, U+0625) and *madda* + *alif* (U+0622) in stem-initial position

الأول =الاول	إن /أن =ان	وإن /وأن =وان	إلى =الى
آخر =اخر	ألا /إلا =الا	إما /أما =اما	الآن =الان

Table 3.5. *hamza* + *alif* (U+0623, U+0625) and *madda* + *alif* (U+0622) in stem-medial and stem-final positions – ambiguity prevents variation in orthography

سال≠سأل	بدا≠بدأ	كان≠كأن
قرآن≠قران	هنأك≠هناك	بان≠بأن

Table 3.6. *hamza* + *alif* (U+0623, U+0625) and *madda* + *alif* (U+0622) in stem-medial and stem-final positions – lack of ambiguity allows for variation in orthography

متأخر =متاخر	بشأن =بشان	تأسيس =تاسيس	بدأت =بدات
تأييد =تاييد	رأت =رات	تستأنف =تستانف	لتأكيد =لتاكيد

variation observed in Table 3.6 challenges one's definition of typographical error: although the variants with bare *alif* (U+0627) are unambiguous to readers, most Arabic spellcheckers would flag these variants as errors. From a morphological analysis perspective these unambiguous words should be labeled simply as instances of sub-standard orthography, and the ability to analyze them should be regarded as basic robust parsing.

The general perception that the two dots on *tā' marbūTa* (ة U+0629) are diacritics is seen in this letter's orthographic variation with word-final and separate *hā*' (ه U+0629; see Table 3.7), although this use of *hā'* as a substitute for *tā' marbūTa* is restricted primarily to informal written Arabic. Our impression is that there is a tendency to write *hā'* in contexts where the *tā' marbūTa* would not be pronounced, such as in noun-adjective phrases (e.g., مدرسه ثانويه /madrasa θānawiyya/), but to preserve the two dots of the *tā' marbūTa* in *iDāfa* (genitive or possessive) constructions (e.g., مدرسة البنات /madrasatu l-banāt/). We are unaware of any corpus-based research conducted in this area of orthography.

The most significant orthographic variation that occurs in Arabic today involves the so-called Egyptian spelling of word-final *yā'* (ي U+064A), which in Egypt, and in various regions where Egyptian spelling predominates, is typically spelled as undotted *yā'* (see Table 3.8).

In the Unicode character set this word-final undotted *yā'* is represented by U+06CC (ARABIC LETTER FARSI YEH). The Unicode standard (2003, p. 59) states that this letter "yeh" is written with dots in initial and medial positions, in which case it maps to Arabic *yā'* (U+064A), and that in final and separate positions it maps to *alif maqSūra* (U+0649). Systems using 8-bit encoding schemes have implemented this undotted *yā'* in two different ways, which we will refer to by their codepages: Mac Arabic and Windows 1256 (both of which antedated the Unicode standard). The Mac Arabic implementation came first, and it reflects the basic definition of Farsi "yeh" found in the Unicode documentation: the letter is dotted in initial and medial positions, and undotted in final and separate positions. However, because the Mac Arabic codepage also needed to provide a word-final

Table 3.7. *tā' marbūTa* (U+0629) spelled as *hā'* (U+0647)

العربيه =العربية	خليفه =خليفة
المتحده =المتحدة	سعاده =سعادة
غزه =غزة	

Table 3.8. Word-final undotted *yā'*

فى = في	التى = التي
الذى = الذي	هى = هي
أى = أي	دولى = دولي

Table 3.9. Two electronic representations of the word رئيس

Unicode character sequence	Google score (April 2005)
U+0631 U+0626 **U+0649** U+0633	967,000
U+0631 U+0626 **U+064A** U+0633	6,870

dotted *yā'*, which is needed outside the areas where Egyptian orthography predominates, two overlapping character-to-glyph mappings were created: the character known as *yā'* (U+064A) was assigned four glyphs (initial, medial, final and separate: ي ـي ـيـ يـ) all dotted, and the character known as *alif maqSūra* was also assigned four glyphs, two dotted (initial and medial: ـيـ يـ), and two undotted (final and separate: ى ـى). The fact that one can type either *yā'* or *alif maqSūra* to create initial or medial dotted *yā'* is seen as a flexible feature by typists, but this has resulted in mechanical orthographic variation (see below), because words such as رئيس /ra'īs/ can now be spelled two different ways electronically (see Table 3.9), and both spellings are attested on the Web with significant Google scores.

Arabic text data that is created on the Mac platform will probably have instances of *alif maqSūra* in word-initial and word-medial positions, and when this text is ported to platforms where *alif maqSūra* is used only in word-final position these orthographically anomalous words become quite obvious (see Figure 3.5). For morphological analysis it is clear that *alif maqSūra* and *yā'* must be treated as two different characters, regardless of what glyphs are used to display either according to Egyptian or non-Egyptian preferences. Therefore, all instances of initial and medial *alif maqSūra* need to be mapped to *yā'*.

The orthographic variation noted above in so-called Egyptian spelling of undotted word-final *yā'* has undergone an interesting additional and unexpected development since the mid 1990s: the use of word-final dotted *yā'* (U+064A) has been introduced gradually, but it has been extended to all words that should be spelled with *alif maqSūra* (U+0649), such as متي /matta/, موسي /mūsa/, الأعلي /al-'aʕla/ and أخري /'uxra/. Coupled with the continuing normal use of undotted word-final *yā'*, this unfortunate practice of dotting *alif maqSūra* has resulted in orthographic ambiguity for both *yā'* and *alif maqSūra*. The upshot for morphological analysis is that, when processing text that comes from Egypt or any region under the influence of Egyptian orthography, one needs to keep in mind that word-final *yā'* may actually represent *alif maqSūra*, and vice versa (which is the normal case).

There is one additional form of mechanical orthographic variation that should be mentioned, and that is the tendency of certain typists to reverse the normal *alif* + *fatHatān* sequence (U+0627 U+064B) because *fatHatān* + *alif* appears to "look

رىٔس - اسراىٔل - علىه - النهاىٔة - أىضا - الىه - الغذاىٔة - العلىا - الثناىٔة - ىجب

Corrected:

رئيس - اسرائيل - عليه - النهائية - أيضا - اليه - الغذائية - العليا - الثنائية - يجب

Fig. 3.5. Arabic text with *alif maqSūra* in initial and medial positions

Table 3.10. Word-final *hamza*-on-*yā'* orthographic variation

مبادىء ←مبادئ	طوارىء ←طوارئ
فوجىء ←فوجئ	الهادىء ←الهادئ
خاطىء ←خاطئ	لاجىء ←لاجئ

Table 3.11. *hamza*-on-*waw* orthographic variation

موءتمر ← مؤتمر	موءكدا ← مؤكدا
مسوءولين ← مسؤولين	موءخرا ← مؤخرا
موءسسات ← مؤسسات	شوءون ← مسؤول

better," although the reverse could also be argued, as seen in the following orthographic word pairs: أحدًا / أحداً, فلانًا / فلاناً, شيئًا / شيئاً, أيضًا / أيضاً. The fact that glyph design and implementation influences people's typing habits can also be observed in several types of *hamza* + *hamza*-chair combinations, such as word-final *hamza*-on-*yā'* (U+0626; see Table 3.10), and *hamza*-on-*wāw* (U+0624; see Table 3.11). The arrows in the tables show the direction in which orthographic normalization should be implemented. In both cases normalization involves mapping two-character sequences to single characters: U+0649 U+0621 to U+0626, and U+0648 U+0621 to U+0624. Note the relative small size of the *hamza* glyph placed on the *wāw* chair, which makes it difficult to read.

A final case of orthographic variation that merits discussion is the neglected area of "run-on" words, by which we mean the observed habit of writing certain types of two-word combinations without intervening or separating space. We regard this as a tokenization problem and we discuss it in full in the section that follows.

3.4 Tokenization of Arabic Words

By tokenization we mean the process of identifying minimal orthographic units or "words" that can be submitted for individual morphological analysis. The working definition of an Arabic word token is straightforward: Arabic words consist of one or more contiguous "alphabetic" characters (i.e., the set of 36 characters, *hamza* through *yā'*, or Unicode U+0621 through U+064A), the set of eight short vowels and diacritics (U+064B through U+0652), the lengthening character (U+0640), and the set of four extended characters associated mainly with Persian usage (پ U+067E, چ U+0686, ڤ U+06A4, and گ U+06AF). Additional extended Arabic characters may enter the Arabic orthographic mainstream as various non-Arab ethnic groups increase their publishing presence and influence on the Web.

For tokenization to work correctly a certain amount of pre-processing is necessary in order to deal with the use of alphabetical characters in non-alphabetical contexts, such as use of the letter *rā'* (U+0631) as numeric comma (see Figure 3.6), and use of the lengthening character (U+0640) as punctuation (e.g., hyphen or

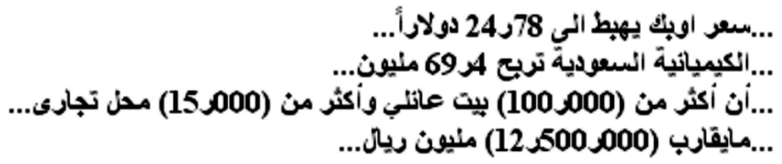

...سعر اوبك يهبط الى 78ر24 دولاراً...
...الكيميائية السعودية تربح 4ر69 مليون...
...أن أكثر من (100ر000) بيت عائلي وأكثر من (15ر000) محل تجارى...
...مايقارب (12ر500ر000) مليون ريال...

Fig. 3.6. Arabic letter *rā'* used as numeric comma

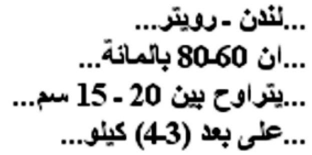

...لندن ـ رويتر...
...ان 60ـ80 بالمائة...
...يتراوح بين 20 ـ 15 سم...
...على بعد (3ـ4) كيلو...

Fig. 3.7. Arabic lengthening character used as punctuation

mdash; see Figure 3.7). Short vowels and diacritics are occasionally used in creative ways as punctuation (in the same manner that Latin lower case "o" is used for bullets), and because of their zero-width display characteristics these Arabic diacritics also have a tendency to become detached from the words they were intended to accompany. Isolated short vowels and diacritics must be treated as punctuation or excluded from the morphological analysis input altogether.

The current approach to Arabic tokenization has overlooked the problem of "run-on" words, which is the writing of two words without intervening space, a condition that typically occurs when the first word ends with any of the thirteen "non-connector" letters: ء آ أ ؤ إ ا ة د ذ ر ز و ى (U+0621-U+0625, U+0627, U+0629, U+062F-U+0632, U+0648-U+0649). Depending on the glyph design (i.e., the perceived width) of the non-connecting letter, the person composing the text may feel free not to insert a space between the non-connector and the first letter of the next word. These "run-on" words are difficult to detect visually, but their presence in digital data is detected immediately and usually flagged as a typographical error. These mistakes are relatively frequent, as witnessed by their Google scores (see Table 3.12).

The most frequent "run-on" words in Arabic are combinations of the high-frequency function words لا /al-/ and ما /mā/ – which end in the non-connector *alif* – with following perfect or imperfect verbs, such as لايزال /lā-yazāl/, مايزام /mā-yazāl/, and مازال /mā-zāl/. The لا /lā/ of "absolute negation" concatenates freely with nouns, as in لابد /lā-budda/ and لاشك /lā-šakka/. It can be argued that these are lexicalized collocations, but their spelling with an intervening space (زال ما , لا يزال and لا بد) is generally more frequent than their spelling as single word units (see Table 3.13).

Proper noun phrases, such as عبدالله / ҫabdallah /, عبدالرحمن /ҫabdurraHmān/, جارالله /jārallah/ and نورالدين /nūruddīn/ are also written with or without intervening

Table 3.12. Run-on words and their frequencies

Google score			Run-on
April 2006	April 2005	March 2004	words
17,100	4,420	846	مديرعام
16,200	1,270	719	وزيرالخارجية
658	352	162	ملياردولار
919	493	158	الدكتورمحمد
703	358	130	وقدتم

Table 3.13. 1-word and 2-word frequencies of run-on words

Google score (4/2006)		Google score (4/2005)		Run-on
as 2-words	as 1-word	as 2-words	as 1-word	words
3,540,000	717,000	412,000	44,300	لايمكن
4,420,000	1,850,000	792,000	87,100	ماهو
2,210,000	2,010,000	188,000	276,000	لابد
1,120,000	192,000	106,000	15,500	لاتزال
1,190,000	155,000	97,500	7,680	ماحدث
1,160,000	210,000	96,500	8,470	مايتعلق
928,000	356,000	83,500	30,300	لاشك
910,000	749,000	83,400	67,800	مازالت
170,000	30,500	12,700	3,760	لاريب

space. These name compounds may be regarded as lexicalized units, but syntactically they are also *iD fa* constructions, and should be treated accordingly as two separate word tokens. Some run-on words can have more than one reading, although this is extremely rare, as in فقدتم fqdtm, which could be read as two words, /fa-qad tamma/, or as a single word, /faqadtum/. The proper solution to this problem is to pre-process input strings and decouple run-on words (Buckwalter, 2004b).

3.5 Archaic Lexical Items and Morphological Features

Morphological analysis of Arabic typically means the analysis of Modern Standard Arabic, which implies the exclusion of archaic vocabulary, orthography and morphological features associated exclusively with the literature and religious texts of the Classical period of Arabic literature. Also excluded from MSA are written forms of the vernacular, although this situation is changing rapidly and will probably become the greatest challenge in contemporary Arabic NLP (see below, "Integrating the dialects in Arabic morphological analysis"). It is not uncommon for MSA texts dealing with religious topics, or with political topics in a religious context, to include quotations from the Qur'an and the Hadith, and these may be a source of archaic lexical items and rare morphological features. Although the orthography of religious quotations could be archaic as well, it is customary to use contemporary orthography (e.g. صلاة SlAћ, rather than صلوة Slwћ).

An example of an archaic lexical item that was used in MSA context not too long ago and disseminated widely in the media is the word ضيزى Dyzý, which is not found in any dictionary of Modern Standard Arabic. This word comes from Qur. 53:22 (al-Najam) /tilka ’iðan qismatun Dīza/, “This, therefore, is an unjust division.” This verse was alluded to in a taped speech by Usama bin Laden which was broadcast widely by the media in November 2002. The implication of this event for morphological analysis of contemporary Arabic is that MSA can be expected to include occasional quotations from the established corpus of traditional religious texts (i.e., Qur’an and Hadith), and that it is advisable to extend the lexical coverage of morphological analysis to such texts, especially since corpus-based lexicography is able to detect the usage and frequency of these archaic lexical items. The phrase قسمة ضيزى qsmħ Dyzý /qismatun Dīza/, for example, is now relatively well attested on the Web.

Certain archaic morphological features are restricted to religious texts and one does not find these features used in new MSA contexts. An example of this kind of archaic feature is the use of direct and indirect object pronoun clitics, as in the word زوجناكها zwjnAkhA /zawwajnākahā/, from Qur. 33:37 (al-Ahzab), “We gave her to you as a wife.” Because these morphological features are limited to specific lexical items in their source religious texts, they should probably be treated as exceptions in the morphological analysis lexicon.

We will conclude by discussing two verbal features that have been categorized as archaic. The first concerns the usage of two alternative forms for the jussive mood of the doubled verb: the short assimilated form of the stem (e.g., يمر ymr /yamurra/) or the long unassimilated form (e.g., يمرر ymrr /yamrur/). Some textbooks and references cite only the assimilated form, يمر ymr /yamurra/, and this is correct because the unassimilated form is not attested in contemporary Arabic and is now considered archaic (Badawi et al., 2004, p. 65). The second feature concerns use of the energetic form, which is rare but not archaic, and requires some attention because it is typically confused in morphological analysis with the feminine plural form. Badawi et al. (2004, pp. 441–2) provide several citations, and we encountered the following citation during the morphological annotation and POS tagging of the first 700,000 words of newswire in the Penn Arabic Treebank (Maamouri et al., 2004). The actual citation is: لا ينخدعن أحد بأنه صديق أو حليف لأمريكا /lā yanxadiçanna ’aHadun bi-’anna-hu Sadīqun ’aw Halīfun li-’amrīkā/. Native informants assure us that the energetic is not uncommon, and can be heard in forceful statements such لا يقومن أحد /lā yaqūmanna ’aHad/ and لا تقولن /lā taqūlanna/. Until we have tagged a substantially larger corpus we cannot assess adequately the extent to which the energetic form is used in contemporary Arabic.

3.6 Lexicon Design and Maintenance

After all aspects of morphological analysis have been adequately addressed, the only way to improve the quality of the analysis is by improving the lexicon. The lexicon can be enhanced in terms of its lexical coverage, by adding new words and new meanings to old words, and also by increasing the level of grammatical detail that is described. We are familiar with two major different types of morphological analysis lexicons: the Xerox lexicon (Beesley, 2001), whose entries are based on root and pattern morphemes, and our own lexicon (Buckwalter, 2004a), whose entries make use of word stems. In the argument over which method represents the correct approach to analyzing a Semitic language such as Arabic, it should be mentioned that although root and pattern morphology is pervasive in the language, approximately seven percent of the entries in the lexicon contain no discernable pattern morpheme (and thus no discernable root morpheme, although Arabs are often capable of extracting root candidates from many non-Semitic words), and that these words must be treated with a stem-based approach. It should also be mentioned that if root and pattern morpheme information is encoded appropriately in the lexicon, it can be reported in the analysis output, regardless of which approach one uses in the analysis. The major difference, of course, is that in a system based on roots and patterns, these morphemes are used in the analysis mechanism itself. The combination, or interdigitation, of these two morphemes produces the equivalent of a stem entry in the lexicon, but not necessarily the equivalent of that stem's canonical surface orthographic form—hence the need for two-level morphology.

Lexicon entries in a two-level morphology system represent not the familiar normalized surface orthography of traditional Arabic dictionary entries, but rather their abstract lexical level. For example, the words *maktaba*, *majalla*, and *maqāla*, are entered in the Xerox lexicon as *maktaba*, **majlala*, and **maqwala*, respectively. Furthermore, because root and pattern morphemes are entered in separate dictionaries, the actual lexicon entries consist of root morphemes (*k-t-b*, *j-l-l*, and *q-w-l*, in this example) and pattern morphemes (*mafçala*, in this example). In the Xerox lexicon Arabic prefixes and suffixes are assigned individual entries in lexicons that group items belonging to the same morpheme class, i.e., items that exhibit the same morphotactic behavior. Hence, in addition to the main lexicons of roots and pattern morphemes, there are various lexicons for morpheme classes such as prepositions, conjunctions, verb subject and object markers, the definite article, noun case endings, and verb mood markers. The morphotactic constraints are implemented via rules that state in which sequence the various lexicons can be accessed by the morphological analysis engine (Beesley et al., 1989).

Our design of a stem-based system of morphological analysis was motivated in part by the complexities we experienced with developing the Xerox lexicon (which was known as the Alpnet lexicon at the time), and with the intricacies in defining the morphotactic constraints. Therefore, in designing our own system we pursued an alternate design that (1) simplified lexicon management by representing lexical items according to their normalized surface orthography, and (2) greatly simplified the morphotactic component of the system by representing all valid concatenations of prefixes and suffixes in the lexicon entries themselves. Maintaining three lexicons—of prefixes, stems, and suffixes—is vastly simpler than maintaining a dozen or more. But the greatest advantage in not using a two-level morphology approach is that lexicon entries are not orthographic abstractions but rather the familiar canonical forms of printed dictionaries. This means that dictionary maintenance need not require a thorough knowledge of Arabic derivational morphology, which few native speakers learn as well as non-native Arabists, who are often known for their ability to cite the form number of any given verb, regular or irregular, followed by its respective active and passive participles and verbal noun.

3.7 Integrating the Dialects in Arabic Morphological Analysis

Modern written Arabic is exhibiting a growing influx of dialectal Arabic, largely as the result of unedited and uncensored publication on the Web. Significant quantities of dialectal Arabic in written form can readily be found on the Web by searching for high-frequency dialectal words (see Table 3.14). Whereas some dialectal words are common to all major dialects (e.g., اللي /illī/), a search for groups of words used only in a given dialect will lead to Web pages with text written mostly in that dialect. For example, a search for the words بدي /biddī/, شو /šū/, ليش /lēš/ and هلق /halla'/ can be used to locate written samples of Levantine Arabic, and a search for النهارده /an-nahār-da/, حاقول /H-a'ūl/, دلوقتى /di-l-wa'tī/ and محدش /ma-Hadš/ can be used to find Web pages with Egyptian Arabic. The text on these Web pages typically contains a mixture of colloquial Arabic and MSA. If the text is a transcription of speech, such as an interview from a television show, it often becomes clear that some sections of the transcript could be sufficiently formal to warrant the use of MSA case endings in the morphological analysis, but that other sections would sound awkward (too formal) with these features present, and that sections abounding in colloquial lexical items and constructions would preclude the presence of almost all case endings (see Figure 3.8). The main point is that these samples of modern written Arabic reflect the full range of registers one hears in modern spoken Arabic, without any clear-cut division between dialect and MSA.

The morphological analysis of Arabic dialects is complicated by the relative absence of orthographic standards and by orthographic variation among different dialects. However, the rising use of dialectal Arabic on the Web is changing this situation and some orthographic conventions are taking shape. For example, it can already be observed that while Egyptians prefer to spell the word /ba'ūl/ "I say" with the *alif* subject marker (باقول bAqwl), Levantine speakers prefer to write it without the *alif* (بقول bqwl). The verbal paradigms of the dialects differ radically from MSA paradigms (e.g., imperative forms such as قول /qūl/, روح /rūH/, شوف /šūf/, and خلي /xallī/) and include additional affixes that increase the complexity of the morphotactic component of morphological analysis (e.g., ماقلتلكش /ma-'ulti-llak-š/ and حيبقالك /Ha-yib'ā-lak/). In addition, the dialectal lexicon makes use of many MSA lexical items, such as خلاص /xalāS/, لازم /lāzim/,ممكن /mumkin/, ماشي /māšī/, باين /bāyin/, يعني /yaçnī/, مال /māl/, صار /Sār/, and راح /rāH/, but with very

Table 3.14. High-frequency dialectal words

احنا	احدعشر	احدعش	اتنين	ابوي	آني
اطنعش	ازيك	ازاي	اربعطعشر	اربعطعش	اخويا
امبيرح	امبارح	النهارده	اللي	اكو	اطنعشر
اوكيه	اوكي	انتي	انتوا	انتو	امتى
باقول	باخذ	ايوه	ايمتى	ايش	ايد
برا	بدي	بجد	بتقول	بتاعة	بتاع
بكره	بكرة	بعدين	بس	برضو	برضه
تسعطعش	تاني	بينات	بيقول	بيبقى	بلكي
حاقول	جوا	جاي	تقول	تسوي	تسعطعشر
خوش	خمسطعشر	خمسطعش	خلي	خالص	حيبقى
رح	راح	دى	دلوقتى	دغري	دايما
شغلات	سطعشر	سطعش	سبعطعشر	سبعطعش	زي
شلون	شكو	شقد	شفته	شفتك	شغلة
شوي	شونك	شون	شو	شنو	شلونك
عشان	عايش	طب	صار	شي	شوية
فيش	عندكو	عم	عليا	علمود	علشان
كويس	كمان	كلش	كده	كام	فين
ليه	ليش	لعد	لسه	كويسين	كويسة
مش	محدش	مبارح	مال	ماكو	ماعرفش
منو	مكانش	مفيش	معلش	معايا	معاك
هاي	نسوي	مين	مية	مو	منيح
هيكي	هيك	هون	هلق	هلا	هسا
يجي	يابا	وين	وياك	ويا	واخد
			يم	يلا	يعني

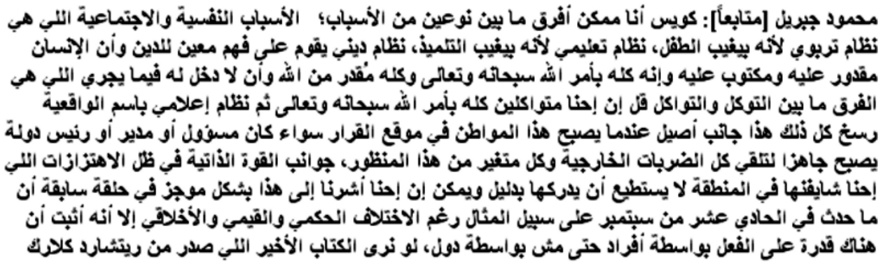

محمود جبريل [متابعاً]: كويس أنا ممكن أفرق ما بين نوعين من الأسباب؛ الأسباب النفسية والاجتماعية اللي هي
نظام تربوي لأنه بيغيب الطفل، نظام تعليمي لأنه بيغيب التلميذ، نظام ديني يقوم على فهم معين للدين وأن الإنسان
مقدور عليه ومكتوب عليه وإنه كله بأمر الله سبحانه وتعالى وكله مُقدر من الله وأن لا دخل له فيما يجري اللي هي
الفرق ما بين التوكل والتواكل قل إن إحنا متواكلين كله بأمر الله سبحانه وتعالى ثم نظام إعلامي باسم الواقعية
رسخ كل ذلك هذا جانب أصيل عندما يصبح هذا المواطن في موقع القرار سواء كان مسؤول أو مدير أو رئيس دولة
يصبح جاهزا لتلقي كل الضربات الخارجية وكل متغير من هذا المنظور، جوانب القوة الذاتية في ظل الاهتزازات اللي
إحنا شايفنها في المنطقة لا يستطيع أن يدركها بدليل ويمكن إن إحنا أشرنا إلى هذا بشكل موجز في حلقة سابقة أن
ما حدث في الحادي عشر من سبتمبر على سبيل المثال رغم الاختلاف الحكمي والقيمي والأخلاقي إلا أنه أثبت أن
هناك قدرة على الفعل بواسطة أفراد حتى مش بواسطة دول، لو نرى الكتاب الأخير اللي صدر من ريتشارد كلارك

Fig. 3.8. Sample of modern written Arabic (www.almanara.org/Audio/2-2-05.htm)

different dialect-specific meanings and grammatical functions. Special attention needs to be given to homographs—items read one way in MSA and another way in the dialects—such as نص /naSS, nuSS/, عليا /ςulya, ςalayya/, and بقدر /bi-qadrin, ba'dar/. In order to sort out dialectal and MSA features in the analysis, it may be necessary to maintain separate lexicons and analysis modules for each dialect. The analysis of MSA data that occurs in a predominantly dialectal context is also not free of complications. For example, should it be vocalized internally according to MSA canonical lexical forms or according to how it is pronounced? What is the vocalization of the MSA word منطقة, for example, in view of its numerous pronunciations: /minTaqa, manTiqa, munTiqa/? These are just a few of the questions that arise when dialectal Arabic commingles with what is traditionally labeled as MSA. In fact, the presence of dialectal Arabic in a text immediately brings into question whether the remaining non-dialectal portion of the text should bear any case endings and mood markers at all in its corresponding morphological analysis.

3.8 Adequacy and Accuracy of Current Morphological Analysis

By "current morphological analysis" we mean the output of the two systems with which we are familiar: the Xerox two-level morphology system (Beesley, 2001) and our own (Buckwalter, 2004). (Other systems are described in this publication, but we have not yet had access to adequate output data samples in order to evaluate them). The Buckwalter system has received considerable exposure and scrutiny because of its use in the Penn Arabic Treebank project (Maamouri et al., 2004). Some incisive criticism of both the Xerox and Buckwalter systems has been provided by Otakar Smrž (in prep.), of the Prague Arabic Dependency Treebank research group at Charles University, especially in areas concerning the lexicon's coverage of gender, number, and humanness features. Essentially, the Xerox and Buckwalter analyzer lexicons provide the default gender and number part-of-speech labels for noun suffixes based on their form without any regard to their actual semantic value or function in the word. For example, the suffix ة (U+0629) is labeled FEM_SG regardless of whether it occurs in the word مدرسة /madrasa/ (fem. sg. non-human), خليفة /xalīfa/ (masc. sg. human) or مغاربة /maγāriba/ (masc. pl. human). This can only be remedied by systematically entering the necessary information on gender, number and humanness for each lexical item.

The set of grammatical features that are currently encoded in the Buckwalter morphological analyzer lexicon have been to a great extent determined by the requirements of the Penn Arabic Treebank project. The Buckwalter system was originally designed for simple word identification, as a first step towards generating lemmatized concordances for use in lexicography. Since its adoption for use in treebanking it has undergone considerable change, mainly in the addition of POS tags, with fairly precise distinctions applied especially to function words (Maamouri et al., 2004). However, it is surprising that many of the traditional grammatical categories that are discussed in all treatments of Arabic morphology have not been needed for successful treebanking, at least as defined in the phrase-structure Penn Treebank model. It is anticipated that forthcoming pedagogical applications of the Buckwalter system will soon require the implementation of the following traditional grammatical labels:

- Gender, number, and humanness.
- Active and passive participles and verbal nouns. These categories are already labeled and cross-linked in the Xerox system, although verbal nouns of Form I need to be linked explicitly to their respective verb.
- Elative. This category needs to be linked to its related adjectival form (e.g. اكبر /'akbar/ → كبير /kabīr/).
- Instance noun, unit noun, and collective noun.
- Verb features such as transitive, intransitive, grammatical collocations, etc.

3.9 Conclusion

We expect the next decade of Arabic morphological analysis to be challenged by added complexity in the input data, as the orthography undergoes unpredictable usage of Unicode characters beyond the basic set required for Arabic, and as the written language itself is transformed by a steady influx of dialectal forms, forcing morphological analysis to deal with diglossic texts. The output analysis itself will be enriched by the complexities and challenges of the input data, and new grammatical features will be added to increase the level of detail and accuracy of the description. Developments in automated syntactic analysis will also influence the priorities that are followed in developing and improving morphological analysis algorithms and lexicons. Arabic morphological analysis as a discipline is still in its early stages.

References

E. Badawi, M.G. Carter, and A. Wallace. 2004. *Modern Written Arabic: A Comprehensive Grammar*. Routledge, London.

Kenneth R. Beesley. 2001. Finite-state Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001, In *EALC 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pp. 1–8, Toulouse, France, July 2001.

Kenneth R. Beesley, S. Newton, and T. Buckwalter. 1989. Two-Level Finite-State Analysis of Arabic Morphology, In *Proceedings of the Seminar on Bilingual Computing in Arabic and English*, no pagination, University of Cambridge, U.K., September 1989.

T. Buckwalter. 2004a. *Buckwalter Arabic Morphological Analyzer Version 2.0.* Linguistic Data Consortium, catalog number LDC2004L02 and ISBN 1-58563-324-0.

T. Buckwalter. 2004b. Issues in Arabic Orthography and Morphology Analysis. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages, COLING 2004*, pp. 31–34, Geneva, August 2004.

M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. Paper presented at the NEMLAR International Conference on Arabic Language Resources and Tools, Cairo, Sept. 22–23, 2004.

Otakar Smrž. in prep. *Functional Arabic Morphology. Formal System and Implementation*. Ph.D. thesis, Charles University in Prague.

The Unicode Consortium. 2003. *The Unicode Standard, version 4.0*. Boston, Addison-Wesley.

H. Wehr. 1979 *A Dictionary of Modern Written Arabic*. 4th edition, edited. by J. Milton Cowan. Wiesbaden, Harrassowitz.

PART II

Knowledge-Based Methods

4

A Syllable-based Account of Arabic Morphology

Lynne Cahill
Natural Language Technology Group, University of Brighton
lynneca@sussex.ac.uk

Abstract: Syllable-based morphology is an approach to morphology that considers syllables to be the primary concept in morphological description. The theory proposes that, other than simple affixation, morphological processes or operations are best defined in terms of the resulting syllabic structure, with syllable constituents (onset, peak, coda) being defined according to the morphosyntactic status of the form. Although most work in syllable-based morphology has addressed European languages (especially the Germanic languages) the theory was always intended to apply to all languages. One of the language groups that appears on the surface to offer the biggest challenge to this theory is the Semitic language group. In this chapter a syllable-based analysis of Arabic morphology is presented which demonstrates that, not only is such an analysis possible for Semitic languages, but the resulting analysis is not significantly different from syllable-based analyses of European languages such as English and German

4.1 Introduction

Approaches to the morphology of the semitic languages have tended to assume that different mechanisms are required to account for a system which, on the face of it, appears very different from typical European morphology such as affixation and ablaut. However, the syllable-based approach to morphology does not require the radically differentiated accounts of such morphological systems. In this approach to morphology, morphological realisations are defined in terms of their syllable structure, with the values of syllabic constituents defined according to a range of possible factors including morphosyntactic features, phonological context and lexical information. It transpires that defining semitic morphology in terms of Syllable Based Morphology (henceforth SBM) requires very similar mechanisms to those required for defining the morphology of European languages. The chief difference between the mechanisms required to define, for example, the various ablaut processes in German and English, is of degree rather than nature. That is, semitic languages may require more different constituents to be defined for each inflected form, but the actual definitions are identical.

It should be stressed that SBM was developed with the intention of being able to define morphological alternations of a wide variety of types from a wide variety of languages from across the world. The theory assumes a model of lexical representation that defines all word forms in terms of their syllable structure.

In this chapter, we present an account of the triliteral verbal morphology of Classical Arabic verbs. We do not set out to provide a comprehensive account of the verbal morphological system of Arabic, as many aspects do not differ in any significant way from the systems of European languages (for example, the affixation processes marking person and number). We begin by giving a brief description of the theory of syllable-based morphology, with examples from English and German. We then outline the Arabic data which we aim to cover. Next, we describe our syllable-based account of Arabic. We then compare this account with previous accounts of English and German, finally giving our conclusions.

4.2 Syllable-based Morphology

The theory of syllable-based morphology is described in Cahill and Gazdar (1997, 1999a, 1999b). The fundamental assumption is that morphological alternations can all be defined in terms of the syllabic structure of a stem. A stem is assumed to consist of a linear string of syllables, and syllables within a string are identified by simple indexing from either end. Syllables are simply numbered from one end or the other. So, for English and German, both of which exhibit extensive suffixation, together with adaptations to the right-hand end of stems, we assume a system of counting from the rightmost, or final, syllable. Their syllable strings therefore have the final syllable as syl1, the penultimate syllable as syl2 and so on. This ignores higher level structures such as feet or tone groups, as well as lower level structures such as mora. However, as argued in Cahill (1990), the indexing appears to be sufficiently powerful to define in elegant terms a wide range of morphological alternations from languages as diverse as English, Bontoc, Sanskrit and Arabic.

The internal structure of a syllable is that given in Pike and Pike (1947), which we take to be relatively uncontroversial. That is, we assume that a syllable consists of an onset and a rhyme. An onset consists of a number of consonants from 0 to 3 (the exact number possible depends on the phonotactic constraints of the language in question). A rhyme consists of a peak (or nucleus) and a coda. A peak is either a vowel or a syllabic consonant. We assume here that long vowels and diphthongs have the same syllabic status as short vowels, although this is not an assumption that will lead to correct analyses of precise phonotactic constraints in many languages. However, this assumption has no implications for the analysis of Arabic presented here and so it is unnecessary to discuss this further. A coda is a number of consonants from 0 to 4 (again, dependent on the phonotactic constraints of the language).

Another assumption of the account presented here is that the syllable structure definitions are embedded within a lexicon structured as a default inheritance hierarchy. For this we use the lexical knowledge representation language DATR (Evans and Gazdar, 1996). This enables the definition of default inheritance networks in a relatively simple and elegant way. In this chapter, we focus on those parts of DATR which are necessary for the exposition of the account of Arabic morphology. It should be stressed that the use of DATR is not essential to the definition of the morphology in a syllable-based way, but merely a convenient way of expressing the information.

DATR allows us to define nodes in a hierarchy which are linked by inheritance paths. So a very simple inheritance network might be something like:

```
Mammal:
    <legs> == 4
    <fur> == yes
    <young> == live.
Human:
    <> == Mammal
    <legs> == 2.
Platypus:
    <> == Mammal
    <young> == egg.
```

Here we define a few simple facts about mammals. We then define two subtypes of mammal, both of which inherit *by default* from Mammal (via the <> paths). They each have one piece of information that is not inherited from the node for Mammal, but which needs to *override* the default value from Mammal. These equations consist of a path (enclosed in angle brackets) on one side and a value on the other side (of the ==). The path consists of zero or more *attributes*.

We assume that lexemes are typically the bottom-most nodes in the hierarchy, and they inherit information from nodes above them in the hierarchy. A fairly typical account of English verbs, for example, would involve nodes representing the inflectional subclasses (*sing-sang-sung, bring-brought* etc.) each of which would in turn inherit from a Verb node. The kind of information that is involved depends very much on the intended application, but might be semantic, syntactic, morphological or phonological. In the case of syllable-based morphology, we are interested in morphology and (to a limited degree) phonology.

The most simple type of morphological alternation is affixation. This is handled in a slightly different way from other alternations, but only in the sense that it is adding material to the stem structure, rather than making adaptations to the existing structure. Affixation is treated simply as concatenation of two or more strings of syllables. Reduplicative affixation is treated in the same way, but has elements of the affix determined (in whole or in part) by elements of the stem.[1]

[1] It can be argued that reduplication is in fact just an extreme example of context sensitivity such as that exhibited by the English plural suffix *-s*. In the case of the English suffix, the voicing feature of the suffix is determined by the final segment of the stem, whereas in the case of reduplication, every feature of the segments of the affix is determined by some element of the stem.

An affix is assumed to have the same structure as a stem, i.e. a linear sequence of syllables. In practice, most affixes are monosyllabic, or even single segments, but the assumed structure is capable of defining all types of affixation.

Other types of morphological alternation involve defining different values for various constituents of syllables within the stem. A vowel alternation such as the umlaut seen in German (*Haus – Häuser*) or any of the ablaut alternations seen in English (*meet – met, sing – sang, choose – chose*) can be defined by specifying different values for the peak in the stem. For example, the lexical entry for *meet* has the following values defined by default[2]:

```
Meet:<> == Verb
    <syl1 onset> == m
    <syl1 peak> == i:
    <syl1 coda> == t.
```

This defines the peak by default to have the value /iː/ in all forms of the verb. The peak value for the past tense form can then be defined as follows:

```
    <syl1 peak past> == E
```

This states that the peak of the past tense and participle forms is /ɛ/. Note that the DATR language allows us to define both forms by means of underspecification. That is, we can further specify either the tense or participle forms by adding this attribute to the definitional path:

```
    <syl1 peak past tense> == E
```

Any constituent of the syllable string can be defined in this way, dependent on the morphosyntactic features that the form realises. We can use two main types of rule: rules of realisation and rules of referral. The rules above are all rules of realisation. That is, they define explicitly how the form is realised (phonologically). Rules of referral, on the other hand, define relations between forms. They involve rules such as the following default rule for German verb forms:

```
    <phn form second> == <phn form third>
```

This rule states that (by default) all second person forms are the same as third person forms. Rules of referral can refer to the local node (as above) or to the global node:

```
    <phn form second> == "<phn form third>"
```

This will refer back to the original node (or lexeme) that was queried. These notions will be explained further as they become relevant for the Arabic account.

4.3 The Arabic Data

The data we will cover in this chapter is from Standard Arabic. We will include the forms for the various different binyanim, as well as the perfective and imperfective active and passive forms. We will not address bi- and quadriliteral roots, even though

[2] Note that, in the DATR code, we make use of the SAMPA computer readable phonetic alphabet (Wells, 1989).

the latter do occur in Classical Arabic. The aim of this chapter is to demonstrate that there is nothing about Arabic morphology that requires special mechanisms, rather than to present a fully comprehensive account of the morphological system of Arabic.

We do not include minimal coverage of the inflections for person and number. As they consist primarily of simple affixation, they do not present any problems for our account. The data we aim to account for is shown in Table 4.1. The full set of forms as generated by the SBM account is shown in Appendix B. The data we cover here is from a single verb, "to write". The forms provided in Table 4.1 are not all actual forms, as not all of the different binyanim actually occur for all verb roots. We, therefore, do not provide glosses for these forms. The meanings of the different binyanim are all related to the stem meaning. For example, the third binyan carries the meaning "to correspond".

The most important observation about the Arabic verbal system is that there appear to be three morphemes that combine to produce a single form. The first of these is the root morpheme, which consists of a skeleton of consonants. In most cases, these are three consonant, or triliteral, roots. These three consonants usually appear in all forms of the verb in question and usually appear in the same order. The second morpheme is the vowel component, which represents the inflection for the word form (active/passive etc.). The final morpheme involved is the CV template, which defines the arrangement of the consonants and vowels. So, a form like *kattab*, which is the perfective active form of the verb "to write", consists of the stem morpheme *ktb*, the verbal inflection *aa* and the template morpheme *CVCCVC*. In order to fully specify the form, the ordering of the Cs and Vs needs to be stated. There are many accounts in the literature of ways of ensuring that the correct consonants get mapped

Table 4.1. Complete set of verb stems for *k-t-b* "to write" (from McCarthy, 1981, 381)

Binyan	Perfective		Imperfective		Participle	
	Active	Passive	Active	Passive	Active	Passive
I	katab	kutib	aktub	uktab	kaatib	maktuub
II	kattab	kuttib	ukattib	ukattab	mukattib	mukattab
III	kaatab	kuutib	ukaatib	ukaatab	mukaatib	mukaatab
IV	ʔaktab	ʔuktib	uʔaktib	uʔaktab	muʔaktib	muʔaktab
V	takattab	tukuttib	atakattab	utakattab	mutakattib	mutakattab
VI	takaatab	takuutib	atakaatab	utakaatab	mutakaatib	mutakaatab
VII	nkatab	nkutib	ankatib	unkatab	munkatib	munkatab
VIII	ktatab	ktutib	aktatib	uktatab	muktatib	muktatab
IX	ktabab		aktabib		muktabib	
X	staktab	stuktib	astaktib	ustaktab	mustaktib	mustaktab
XI	ktaabab		aktaabib		muktaabib	
XII	ktawtab		aktawtib		muktawtib	
XIII	ktawwab		aktawwib		muktawwib	
XIV	ktanbab		aktanbib		muktanbib	
XV	ktanbay		aktanbiy		muktanbiy	

to the correct slots (the eighth binyan flop rule is a particularly nice example!) (McCarthy 1981, 1990). However, our account does not require any special rules to ensure this, as the consonant and vowel values are defined for all forms in exactly the same way.

The system appears to lend itself to such a separation of the morphemes. The root morpheme provides the underlying sense of the forms, the vowel morpheme provides the inflectional form and the template provides the derived form, or binyan. This in turn provides more information about the interpretation of the sense. The account presented here, while not requiring a separation of morphemes, retains this separation of the kinds of information provided. The organisation of the lexicon reflects the separation, with information about the binyan provided by a set of nodes designed for that purpose, information about the root provided by what we would consider the true lexeme nodes and information about the vowel inflections provided by a set of inflectional nodes accessed by all verbs in the lexicon. This organisation is analogous to the organisation of verbs in English and German, with the slight exception of the binyan nodes. These, however, can be viewed as similar to derived forms of words that use fully productive and transparent processes, such as the *un-* prefix in English.

4.4 A syllable-based Account of Arabic

In this section we will present our account of Arabic morphology in three sections. The first section will describe the overall approach, and in particular, the definition of the lexeme entries. The second section will look at the verbal inflections realised as vowel patterns. The third section will describe the derivation of the different binyanim.

4.4.1 The Overall Approach

First, we must examine the verbal forms and determine exactly which parts are determined by what. A fully inflected form of an Arabic verb may consist of prefixes, a stem and suffixes. The suffixes are person and number agreement markers while the prefixes may indicate things like conjunctions. The stem is comprised of arrangements of consonants and vowels which indicate the root, the tense and the mood. We will not look in any detail at the agreement prefixes and suffixes, but we will concentrate on the stem, to which these affixes may attach.

The stem in question consists of the root consonants, the tense/mood vowels and in some cases prefixes that indicate tense and mood.[3] In our account, we distinguish between agreement prefixes and tense prefixes, which always come closest to the root and form part of the stem to which the agreement affixes attach.

The basic lexeme entries in our account of Arabic define the consonantal roots. However, in our theory, all roots, stems and affixes must be defined as syllable

[3] As will be explained below, we choose to define elements that come before the initial root consonant as prefixes, rather than defining a different syllable structure.

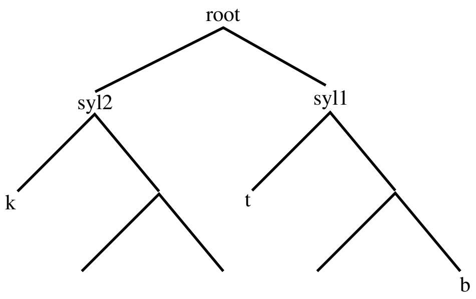


Fig. 4.1. The default structure of the form *katab*

strings. We must therefore define a default syllable structure for triliteral stems, with default positions for the stem consonants. We assume the most simple structure, as exemplified by the stem *katab*. The standard phonotactics of Arabic require syllables of CV (preferred) or CVC structure. Our root must be divided into two syllables, because there are three consonants, and syllables in Arabic may have a maximum of two consonants. The syllable boundary must be before the middle consonant, as syllables with a VC structure are not allowed. This therefore gives us a syllable structure as in Figure 4.1. Note that this structure has no values specified for either peak, nor for the coda of the first syllable. There is an underlying assumption in our theory that any constituents whose values remain unspecified when all information (from the lexeme, inflection and derivation parts) is taken into account are 0. Note also that we count the syllables from the right-hand end, so the final syllable is labelled as syl1, the penultimate syllable is syl2 and so on. For the majority of forms, this is not significant as they all have two syllables. However, there are forms (binyanim 5 and 6) which add a syllable. As they both add the syllable at the front, it makes more sense to count from the right-hand end so as not to disrupt the syllables that are shared (in large part) by all binyanim. It should be noted that we opt for an account that involves several cases of incomplete (illegal) syllables, for example, syllables that lack a peak. As we will discuss in Section 4.3, this assumption of a resyllabification process is required in virtually any phonologically based account of morphology in any language.

The underlying syllable structure is defined by default for all languages with the following three nodes:

```
Null:
    <> == .

Syllable:
 <> == Null
 <phn root>  == <phn syl1>

 <phn $syll form>  == "<phn $syll onset>"
                      "<phn $syll rhyme>"
```

```
   <phn $yll rhyme> == "<phn $yll peak>"
                       "<phn $yll coda>".

 Disyllable:
     <> == Syllable
     <phn root>  == <phn syl2> <phn syl1>.
```

The first of these simply provides the ultimate default value, as discussed above, as being a zero realisation. The second node defines the default syllable structure, as well as the default value for a root as being monosyllabic. The obvious default for English and German is monosyllabic. This is not necessarily the same for Arabic, as we know that verbs and nouns virtually all have disyllabic (or at least, triliteral) roots. However, as there are plenty of monosyllables in Arabic (e.g. function words), the default value is valid, as these specifications can be given at the nodes for nouns and verbs. Attributes that begin with a $ symbol are variables. The variable $yll here ranges over the values syl1, syl2, syl3 etc., indicating that the statements here apply to any syllable.

The statements simply say that the phonological form of a syllable is the value of the onset followed by the value of the rhyme. The rhyme, in turn, is the value of the peak followed by the value of the coda.

For Arabic, we can define the default verb structure to be disyllabic, with the following definition as one part of the information defined for verbs:

```
 Verb:
     <> == Disyllable
```

The values to be specified for the lexeme *katab* can then be defined simply as follows:

```
 Katab:
     <> == Verb
     <phn syl2 onset> == k
     <phn syl1 onset> == t
     <phn syl1 coda> == b.
```

In fact, however, we abstract from this to define the default root for verbs as follows[4]:

```
     <phn syl2 onset> == Root:<c1>
     <phn syl1 onset> == Root:<c2>
     <phn syl1 coda>  == Root:<c3>
```

Then, for *katab*, we need only:

```
     <c1> == k
     <c2> == t
     <c3> == b
```

The nature of the default inheritance we assume means that any of the values defined in any of these nodes can be redefined (overridden). This will be seen extensively in the definition of the derived forms (or binyanim) below.

[4] The node name `Root` here refers to whichever lexeme node is being queried.

So, to sum up, at the top of the inheritance hierarchy we have a number of nodes that define the overall structure of the account. The very top is identical to the hierarchies for English and German. This part defines the default structure of syllables, and defines roots as being monosyllables by default. This part of the hierarchy also defines a word as consisting of a root and a suffix, again by default.

The next part of the hierarchy, the Verb node, defines the default values for Arabic verbs, including the default definition of verbs as disyllabic as already mentioned. This performs two main functions. The first of these is to define default values for the vowels for some of the inflected forms. The second is to map definitions relating to the fifteen different binyanim to individual nodes providing these definitions, for example[5]:

```
<bin1> == "Bin1:<>"
```

The next level of the analysis is the set of binyan nodes, which define the structure (position of the consonants) and the vowels for the various binyan forms. These nodes also define tense prefixes, which consist of either a single vowel or a consonant and a vowel.

Finally, the lowest level of the hierarchy comprises the lexeme nodes. For our purposes, these nodes consist of nothing more than the three consonants of the root, but in a lexicon for a real application, they would include syntactic and semantic information relating to that lexeme.

4.4.2 The Vowel Inflections

In many languages, including English and German, many morphological alternations are characterised by vowel changes or ablaut. The main difference between these ablaut processes and the vowel patterns seen in Arabic morphology is that typically English and German ablaut processes only affect a single vowel (or peak) in a stem. In fact, most of the words that undergo ablaut processes in these languages are monosyllabic anyway. However, we do have one instance of multiple vowel change in English, with the word *woman/women*. Although orthographically only the second vowel changes, phonologically both vowels change:

/wʊmən/ – /wɪmɪn/

Of course, we would not want to claim that a single example means that this is a common thing in English morphology, but the fact that it exists at all does at least demonstrate that any comprehensive account of English morphology will need to account for such phenomena. We could, of course, simply say that this is a lexical exception which cannot (or should not) be accounted for in a rule-governed way. However, we feel that this is a get-out that is not really satisfactory. The forms in question are not random suppletive forms, but simply forms that exhibit an unusual set of changes.

[5] The quotes here indicate that the node `Bin1` now becomes the global node. This is why the equations above require reference to the original query node via `Root`.

It is worth stressing at this point that, although cases of disyllabic stems where both vowels exhibit alternations are rare, cases of more than one constituent in the stem changing, or other combinations of alternation types are not unusual in English and German. German has nouns whose plural is formed with a combination of umlaut and a suffix (*Haus – Häuser*). English has nouns whose plural is formed with a combination of a suffix and a change in the final consonant (*wife – wives*).

So we need to define, for the main verbal inflections, two vowel slots. In the most basic form, the perfective active form of binyan I, the two vowels are both /a/. We can define these values very simply, as follows:

```
Verb:
    <> == Disyllable
    <phn syl2 peak> == a
    <phn syl1 peak> == a
```

These lines tell us simply that unless otherwise specified, the vowels in both syllables have the value /a/.

For some specific inflections, we give the different default vowels as follows:

```
    <phn syl2 peak perf pass> == u
    <phn syl1 peak perf pass> == i
    <phn syl1 peak imp act> == i
    <phn syl1 peak part act> == i
```

These lines tell us that the vowels for the perfective passive are /u/ and /i/, giving *kutib*. The imperfective active has the second vowel specified here as /i/ and the active participle form specifies the second vowel as /i/. These values are merely defaults, and do not necessarily represent any individual form. In fact, for binyan I, the first vowel in each of the last two cases is zero (the forms *aktub* and *uktab*).

In addition to the vowel specifications, we also define here the tense prefixes:

```
    <tense prefix imp act> == a
    <tense prefix imp pass> == u
    <tense prefix part> == m u
```

These definitions complete the default values.

4.4.3 The Binyanim

The different binyanim are defined in terms of the positions of the vowels and consonants. The first few are all defined in their own terms, but there is a certain amount of inheritance possible, especially in the later binyanim. The exact inheritance structure we use is given in Figure 4.2.

This figure shows that binyanim 1 – 5, 7 and 8 all inherit from the Verb node, with binyan 10 inheriting from binyan 4 and binyan 6 inheriting from 5. Binyanim 9 and 12 inherit from 8, binyan 13 inherits from 12, 11 and 14 inherit from 9 and finally binyan 15 inherits from binyan 14. With these inheritance patterns it is possible to

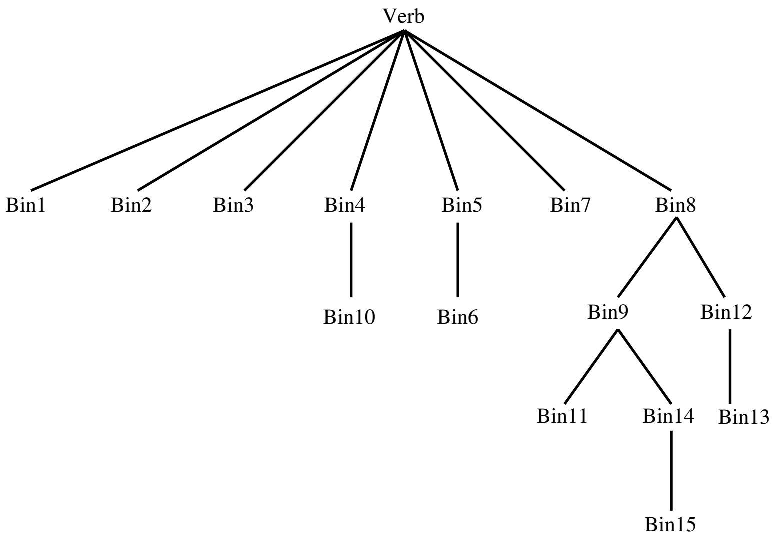


Fig. 4.2. The inheritance structure of the binyanim

define all of the binyanim forms with relatively few equations. Let us now look at how we do this.

The forms for binyan 1 are as follows:

perfective		imperfective		participle	
active	passive	active	passive	active	passive
katab	kutib	aktub	uktab	kaatib	maktuub

The binyan I form is the most basic (semantically), the most widely used and, as we might expect from the more frequently used areas of a morphological system, the one with the most irregularities. We analyse the forms as follows. Any elements that occur before the initial root consonant (in this case, /k/) are considered to be prefixes. Thus the imperfective forms have prefixes /a/ and /u/, and the participle passive form has the prefix /ma/. The three consonants all occur only once, and in the correct order, so we do not need to say any more about those. The vowels, on the other hand, show a large amount of variation. We see the following six patterns:

a	a
u	i
0	u
0	a
aa	i
0	uu

The first pair is the default value for verbs anyway, so does not need to be specified here (see below). The second pair, likewise, is specified at the Verb node. The third pair is specified here, and the zero value for the fourth pair is also covered in the same definition, specifying the zero for both imperfective forms (by underspecification). The /a/ of this pair is inherited from Verb. The participle active form is /aa/. The final pair above is given explicitly in full. In addition to the vowel pairs above, we need to define the prefixes. For this, we need to specify two that are different from the definitions given at Verb. These are the two participle forms. For the other prefixes, we just need to inherit the Verb definitions.

The whole node for binyan I forms is given below:

```
Bin1:
    <> == Verb
    <phn syl1 peak imp act> == u
    <phn syl2 peak imp> ==
    <phn syl2 peak part pass> ==
    <phn syl2 peak part act> == a a
    <phn syl1 peak part pass> == u u
    <tense prefix part act> ==
    <tense prefix part pass> == m a.
```

There are a number of points that deserve some comment here. The first line here says (uncontroversially) that this node inherits by default from the Verb node. The next line tells us that in the imperfective active form, the peak of the second syllable is /u/. The next line tells us that the first peak of this (and the imperfective passive form) is zero. These lines, together with the default values above give us the form /aktub/ for the first binyan imperfective active forms. There are five more equations, all of which relate to the participle forms. The participle passive form has a null vowel in the first syllable (like the imperfect forms) and could indeed be defined with a rule of referral:

```
    <phn syl2 peak part pass> == <phn syl2 peak imp>
```

We have not opted to do this, as it does not save anything, and is not a mapping that we want to apply in any other context.

The other binyan definitions are much simpler to describe. Binyan 2 needs the middle consonant to be doubled. We do this by specifying that the coda of the initial syllable is the second root consonant. The prefix of the imperfective active form is also defined to be /u/.

```
Bin2:
    <> == Verb
    <phn syl2 coda> == Root:<c2>
    <tense prefix imp act> == u.
```

The third binyan form has the first vowel doubled and defines the prefix of the imperfective active form to be the same as that for Binyan 2. The doubled vowel is defined as two "copies" of the vowel as defined at the Verb node. It is arguable whether this is actually a sensible way to define this value. Would it not be simpler and more efficient to just state that its value is /aa/? The answer is that, in this case, this would be a more sensible approach, but in a situation where the same mapping applied to more than one value for that path, the generalisation would be worth making. The fourth binyan has a switch of consonants, with the glottal stop /ʔ/ becoming the onset of the first syllable throughout and the initial root consonant becoming the coda of the initial syllable. This gives forms like /ʔaktab/ and /ʔuktib/. Although we might want to consider the glottal stop part of a prefix, as it comes before the initial root consonant, this analysis does not fit well with the remainder of the forms. If we take the approach we have taken, we can inherit all of the other definitions from the nodes higher up in the hierarchy.

```
Bin3:
    <> == Verb
    <phn syl2 peak> == Verb Verb
    <tense prefix imp act> == Bin2.

Bin4:
    <> == Verb
    <phn syl2 onset> == '?'
    <phn syl2 coda> == Root:<c1>
    <tense prefix imp act> == Bin2.
```

Binyan 10 inherits from binyan 4, with two differences. The first sees the two consonants /st/ replacing the first consonant, instead of the glottal stop. The second is the prefix for the imperfective active form, which comes direct from Verb, rather than from binyan 4.

```
Bin10:
    <> == Bin4
    <phn syl2 onset> == s t
    <tense prefix imp act> == Verb.
```

The binyan 5 forms are interesting, in that they appear at first glance to require an additional prefix of /ta/:

perfective		imperfective		participle	
active	passive	active	passive	active	passive
takattab	tukuttib	atakattab	utakattab	mutakattib	mutakattab

However, it can be seen from the imperfective and participle forms that this "prefix" attaches to the root, after the tense prefixes. We therefore account for this set of forms by defining this root as a trisyllabic root, with the additional syllable consisting of /t/, together with the vowel of the second syllable. There are only two other small elements that need definition here: the final peak of the imperfective active form is /a/ (rather than the /i/ that we might expect from the other binyanim), and the coda of the second syllable is the same as for binyan 2. There might be an argument that we should make binyan 5 inherit from binyan 2. This is a valid position, but we have not chosen to do this, because, although many forms are virtually the same as the binyan 2 forms, with the additional syllable, there are two forms which require different specifications, and so the definitions would not benefit from being inherited from binyan 2.

```
Bin5:
    <> == Verb
    <phn root> == Trisyllable
    <phn syl3 onset> == t
    <phn syl3 peak> == "<phn syl2 peak>"
    <phn syl1 peak imp act> == a
    <phn syl2 coda> == Bin2.
```

Binyan 6 similarly looks very much like binyan 3, but with the extra syllable. However, we define this as inheriting from binyan 5, with three small differences. The peak of the additional syllable is always /a/, rather than being dependent on the second peak. The middle consonant is not doubled here, so the coda value is referred back to the Verb node, and the second peak is doubled, which it inherits from binyan 3.

```
Bin6:
    <> == Bin5
    <phn syl2 coda> == Verb
    <phn syl3 peak> == a
    <phn syl2 peak> == Bin3.
```

The seventh binyan forms are also defined as trisyllabic, with the extra syllable being defined as having only a coda, /n/. This binyan picks up the default values from Verb for the vowels (which are significantly different from the binyan 1 forms).

```
Bin7:
    <> == Verb
    <phn root> == Trisyllable
    <phn syl3 coda> == n.
```

Binyan 8 is the start of a more interesting set of inheritances involving seven of the remaining sets of forms. Binyan 8 itself is very simply defined as inheriting from Verb, with just one difference: the second consonant is added to the first onset (after the first consonant). This gives the forms like /ktatab/.[6] Binyan 9 inherits this information, but changes the onset of the second syllable to be the third root consonant, rather than the second. The forms for binyan 11 are similarly related to binyan 9 forms, but with the single difference that the initial vowel is doubled, which is inherited from binyan 3.

[6] The onset of this form is not a legal onset in Arabic syllables. However, these forms will have other affixes added and a process of resyllabification will result in the surface forms having legal syllable structures.

```
Bin8:
    <> == Verb
    <phn syl2 onset> == Verb Root:<c2>.

Bin9:
    <> == Bin8
    <phn syl1 onset> == Root:<c3>.

Bin11:
    <> == Bin9
    <phn syl2 peak> == Bin3.
```

Binyanim 14 and 15 also inherit from binyan 9. The forms for binyan 14 simply add the consonant /n/ as the initial coda (/ktanbab/) and for 15, the final coda is also changed to be /y/ (/ktanbay/).

```
Bin14:
    <> == Bin9
    <phn syl2 coda> == n.

Bin15:
    <> == Bin14
    <phn syl1 coda> == y.
```

The forms for binyanim 12 and 13 are also inherited from binyan 8. The binyan 12 forms differ from those of binyan 8 only in having a /w/ as the first coda (/ktawtab/), while 13 changes the second consonant also to /w/ (/ktawwab/).

```
Bin12:
    <> == Bin8
    <phn syl2 coda> == w.

Bin13:
    <> == Bin12
    <phn syl1 onset> == w.
```

4.5 Comparison Between Accounts of Arabic and English and German

One of the most striking features of this account of the verbal system of Arabic is that the equations used look, for the most part, no different from the equations required to define English and German morphology. The comprehensive accounts of the verbal morphology of English and German that have been implemented in a syllable-based framework have structures that are very similar at the top. That is, they define a default structure with syllables counted from the right-hand end of the stem. They define a default structure consisting of a root and a suffix (with possible prefixes, in

German). Many definitions for the sub-regular classes of verbs in both English and German define values for peaks.

One aspect of the English and German accounts that does not occur in (this fragment of) the Arabic verbal system is the question of "active affixes" (Cahill and Gazdar, 1999). We thus define affixes whose precise realisation may vary depending on morphosyntactic or phonological context. Although our account does not look at such details in the Arabic system, and so this may well turn out to have similar features, what is interesting is that the areas of the Arabic account that appear to differ most from the English or German, namely the binyan definitions, do resemble the active affix definitions more than the stem definitions. The definition of different consonants and different syllable positions for consonants does not happen very much within the stems in English and German, but it does happen within the affixes. For example, German verb suffixes are defined as varying between *-e, -te, -et, -est, -test* etc.

One aspect of the Arabic account which may appear to present problems is also a potential problem for the accounts of English and German. This is the question of resyllabification. The roots, affixes and final forms are all defined in terms of full syllable structures, but in many cases, the final syllable structure is different from the structures at earlier stages of the derivation. To cite a simple example, the English word *inform*, has the consonant /m/ as the final coda. However, when it has the suffix *-ation* added, the /m/ moves to the onset of the following syllable. This is a very simple example of the kind of thing that will always have to be dealt with in any account of morphology that assumes underlying phonological structures. We therefore feel that having to account for situations like that in binyan 7, where we have introduced a syllable that consists of only a coda /n/, is no more problematic than dealing with the suffixation example above.

4.6 Conclusions

We have presented an account of Arabic verbal morphology within the theory of syllable-based morphology. This approach allows us to use precisely the same mechanisms as used for defining the morphological systems of languages such as English and German. In addition to this fact, it can be seen from the code defining all the different binyanim (see Appendix A) that these widely varying, but obviously related, sets of forms can be defined very simply with only a few equations within a fairly simple inheritance network.

We have not provided accounts of the other variations found within Semitic languages, notably the quadriliteral and biliteral roots and the so-called weak forms, where one of the root consonants is either /y/ or /w/. However, we are confident that accounts of these forms could be simply defined within the overall framework defined above. The quadriliteral and biliteral forms display similar form sets from the triliteral roots. The quadriliteral roots have a restricted set of binyanim, all of which have positions for all four consonants. The biliteral roots simply have to specify which of the two root consonants takes the place of the (missing) third consonant in each form. The situation with weak forms is actually very much the kind of phenomenon that syllable-based morphology has been used to address within English and German morphology, namely phonologically-conditioned variation. Again, to take a simple example, the English plural suffix *-s* has three different phonetic realisations, depending on the phonological form of the final coda of the stem. This kind of variation is equivalent to the variation seen in weak forms in Arabic.

In conclusion, we believe that this account of Arabic, in addition to being a linguistically interesting and computationally efficient way of representing the verbal system of Arabic, demonstrates the wide applicability of the theory of syllable-based morphology.

References

Cahill, L. 1990. "Syllable-based morphology", In *COLING-90*, Vol.3 pp. 48–53, Helsinki.

Cahill, L. and G. Gazdar. 1997. "The inflectional phonology of German adjectives, determiners and pronouns", In *Linguistics* 35:2, pp. 211–245.

Cahill, L. and G. Gazdar. 1999a. "The `PolyLex` architecture: multilingual lexicons for related languages", In *Traitement Automatique des Langues*, 40:2, pp. 5–23.

Cahill, L. and G. Gazdar. 1999b. "German noun inflection", In *Journal of Linguistics* 35:1, pp. 211–245.

Evans, R. and G. Gazdar. 1996. "`DATR`: A language for lexical knowledge representation", In *Computational Linguistics* 22:2, pp. 167–216.

McCarthy, J. 1981. "A prosodic theory of nonconcatenative morphology" In *Linguistic Inquiry* 12, pp. 373–418.

McCarthy, J. and A. Prince. 1990. "Prosodic morphology and templatic morphology" In M. Eid and J. McCarthy (eds) *Perspectives on Arabic Linguistics: Papers from the Second Symposium* pp. 1–54.

Pike, K.L. and E.V. Pike. 1947. "Immediate constituents of mazateco syllables", In *International Journal of American Linguistics* 13, 1947, pp. 78–91.

Wells, J. 1989. "Computer-coded phonemic notation of individual languages of the European Community", In *Journal of the International Phonetic Association*, 19:1, pp. 31–54.

Appendix A: The `DATR` Code

The code listed here is available in the DATR archive at www.datr.org.

```
% The structure of syllables and syllable sequences

# vars $syll: syl1 syl2 syl3.

Null:
    <> == .
Syllable:
 <> == Null
 <phn root>  == <phn syl1>

 <phn $yll form>  == "<phn $yll onset>" "<phn $yll rhyme>"
 <phn $yll rhyme> == "<phn $yll peak>"  "<phn $yll coda>"
 <phn $yll coda>  == "<phn $yll body>"  "<phn $yll tail>".

Disyllable:
    <> == Syllable
    <phn root>  == <phn syl2> <phn syl1>.

Trisyllable:
    <> == Syllable
    <phn root>  == <phn syl3> <phn syl2> <phn syl1>.

% % % % % % % % % % % % % % % % % % % % % % % % % % % % %

% The (default) structure of affixes and words

Affix:
    <> == Syllable.

Word:
    <> == Syllable

    <mor word> == "<phn root form>" "<mor suffix>".

Verb:
    <> == Disyllable
    <mor word> == "<agr prefix>" "<tense prefix>"
                  "<phn root form>" "<agr suffix>"

    % <agr prefix> and <agr suffix> are not defined
    % further here - they will be realised as 0.

    % basic root structure
    <phn syl2 onset> == Root:<c1>
    <phn syl1 onset> == Root:<c2>
    <phn syl1 coda>  == Root:<c3>

    <phn syl2 peak> == a
    <phn syl1 peak> == a

    % default vowels for different tenses
    <phn syl2 peak perf pass> == u
    <phn syl1 peak perf pass> == i
    <phn syl1 peak imp act> == i
    <phn syl1 peak part act> == i

    % prefixes for different tenses
    <tense prefix imp act> == a
    <tense prefix imp pass> == u
    <tense prefix part> == m u

    % mapping different binyan forms to binyan nodes.
    <bin1> == "Bin1:<>"
    <bin2> == "Bin2:<>"
    <bin3> == "Bin3:<>"
    <bin4> == "Bin4:<>"
    <bin5> == "Bin5:<>"
    <bin6> == "Bin6:<>"
    <bin7> == "Bin7:<>"
    <bin8> == "Bin8:<>"
    <bin9> == "Bin9:<>"
    <bin10> == "Bin10:<>"
    <bin11> == "Bin11:<>"
    <bin12> == "Bin12:<>"
    <bin13> == "Bin13:<>"
    <bin14> == "Bin14:<>"
    <bin15> == "Bin15:<>".

% Binyan I - the most basic and with the most
% "irregularities"
Bin1:
    <> == Verb
    <phn syl2 coda imp act> == "<c1>"
    <phn syl1 peak imp act> == u
    <phn syl2 peak imp> ==
    <phn syl2 peak part pass> ==
    <phn syl2 peak part act> == a a
    <phn syl1 peak part pass> == u u
    <tense prefix part act> ==
    <tense prefix part pass> == m a.

% Binyan II - doubled middle consonant, one prefix
% difference
Bin2:
    <> == Verb
    <phn syl2 coda> == Root:<c2>
    <tense prefix imp act> == u.

% Binyan III - doubled first vowel, prefix difference
% as above
Bin3:
    <> == Verb
    <phn syl2 peak> == Verb Verb
    <tense prefix imp act> == Bin2.


% Binyan IV - c1 moved and /?/ added in its place,
% prefix as above
Bin4:
    <> == Verb
    <phn syl2 onset> == '?'
    <phn syl2 coda> == Root:<c1>
    <tense prefix imp act> == Bin2.

% Binyan V - trisyllable, has extra syllable's
% values defined, different final peak, consonant doubling
% as for Bin II.
Bin5:
    <> == Verb
    <phn root> == Trisyllable
    <phn syl3 onset> == t
    <phn syl3 peak> == "<phn syl2 peak>"
    <phn syl1 peak imp act> == a
    <phn syl2 coda> == Bin2.

% Binyan VI - as Bin V but no consonant doubling,
% first peak fixed, second doubled as Bin III.
Bin6:
    <> == Bin5
    <phn syl2 coda> == Verb
    <phn syl3 peak> == a
    <phn syl2 peak> == Bin3.

% Binyan VII - trisyllable, coda only defined for
% first syllable
Bin7:
    <> == Verb
    <phn root> == Trisyllable
    <phn syl3 coda> == n.

% Binyan VIII - adds second consonant to first onset.
Bin8:
    <> == Verb
    <phn syl2 onset> == Verb Root:<c2>.

% Binyan IX - as Bin VIII but second coda is third,
% not second, consonant
Bin9:
    <> == Bin8
    <phn syl1 onset> == Root:<c3>.

% Binyan X - as Bin VI, but with /st/ in place
% of first consonant
Bin10:
    <> == Bin4
    <phn syl2 onset> == s t
    <tense prefix imp act> == Verb.

% Binyan XI - as Bin IX, but vowel doubled as in Bin III
Bin11:
    <> == Bin9
    <phn syl2 peak> == Bin3.

% Binyan XII - as Bin VIII, first coda /w/
Bin12:
    <> == Bin8
    <phn syl2 coda> == w.

% Binyan XIII - as Bin XII, second onset also /w/
Bin13:
    <> == Bin12
    <phn syl1 onset> == w.

% Binyan XIV - as Bin IX, first coda /n/
Bin14:
    <> == Bin9
    <phn syl2 coda> == n.

% Binyan XV - as Bin XIV, final consonant /y/
Bin15:
    <> == Bin14
    <phn syl1 coda> == y.

Write:
    <> == Verb
    <c1> == k
    <c2> == t
    <c3> == b.
```

Appendix B: The Output of the Theory

```
Write:<bin1 mor word perf act> = k a t a b.
Write:<bin2 mor word perf act> = k a t t a b.
Write:<bin3 mor word perf act> = k a a t a b.
Write:<bin4 mor word perf act> = ? a k t a b.
Write:<bin5 mor word perf act> = t a k a t t a b.
Write:<bin6 mor word perf act> = t a k a a t a b.
Write:<bin7 mor word perf act> = n k a t a b.
Write:<bin8 mor word perf act> = k t a t a b.
Write:<bin9 mor word perf act> = k t a b a b.
Write:<bin10 mor word perf act> = s t a k t a b.
Write:<bin11 mor word perf act> = k t a a b a b.
Write:<bin12 mor word perf act> = k t a w t a b.
Write:<bin13 mor word perf act> = k t a w w a b.
Write:<bin14 mor word perf act> = k t a n b a b.
Write:<bin15 mor word perf act> = k t a n b a y.
Write:<bin1 mor word perf pass> = k u t i b.
Write:<bin2 mor word perf pass> = k u t t i b.
Write:<bin3 mor word perf pass> = k u u t i b.
Write:<bin4 mor word perf pass> = ? u k t i b.
Write:<bin5 mor word perf pass> = t u k u t t i b.
Write:<bin6 mor word perf pass> = t a k u u t i b.
Write:<bin7 mor word perf pass> = n k u t i b.
Write:<bin8 mor word perf pass> = k t u t i b.
Write:<bin10 mor word perf pass> = s t u k t i b.
Write:<bin1 mor word imp act> = a k t u b.
Write:<bin2 mor word imp act> = u k a t t i b.
Write:<bin3 mor word imp act> = u k a a t i b.
Write:<bin4 mor word imp act> = u ? a k t i b.
Write:<bin5 mor word imp act> = a t a k a t t a b.
Write:<bin6 mor word imp act> = a t a k a a t a b.
```

```
Write:<bin7 mor word imp act> = a n k a t i b.
Write:<bin8 mor word imp act> = a k t a t i b.
Write:<bin9 mor word imp act> = a k t a b i b.
Write:<bin10 mor word imp act> = a s t a k t i b.
Write:<bin11 mor word imp act> = a k t a a b i b.
Write:<bin12 mor word imp act> = a k t a w t i b.
Write:<bin13 mor word imp act> = a k t a w w i b.
Write:<bin14 mor word imp act> = a k t a n b i b.
Write:<bin15 mor word imp act> = a k t a n b i y.
Write:<bin1 mor word imp pass> = u k t a b.
Write:<bin2 mor word imp pass> = u k a t t a b.
Write:<bin3 mor word imp pass> = u k a a t a b.
Write:<bin4 mor word imp pass> = u ? a k t a b.
Write:<bin5 mor word imp pass> = u t a k a t t a b.
Write:<bin6 mor word imp pass> = u t a k a a t a b.
Write:<bin7 mor word imp pass> = u n k a t a b.
Write:<bin8 mor word imp pass> = u k t a t a b.
Write:<bin10 mor word imp pass> = u s t a k t a b.
Write:<bin1 mor word part act> = k a a t i b.
Write:<bin2 mor word part act> = m u k a t t i b.
Write:<bin3 mor word part act> = m u k a a t i b.
Write:<bin4 mor word part act> = m u ? a k t i b.
Write:<bin5 mor word part act> = m u t a k a t t i b.
Write:<bin6 mor word part act> = m u t a k a a t i b.
Write:<bin7 mor word part act> = m u n k a t i b.
Write:<bin8 mor word part act> = m u k t a t i b.
Write:<bin9 mor word part act> = m u k t a b i b.
Write:<bin10 mor word part act> = m u s t a k t i b.
Write:<bin11 mor word part act> = m u k t a a b i b.
Write:<bin12 mor word part act> = m u k t a w t i b.
Write:<bin13 mor word part act> = m u k t a w w i b.
Write:<bin14 mor word part act> = m u k t a n b i b.
Write:<bin15 mor word part act> = m u k t a n b i y.
Write:<bin1 mor word part pass> = m a k t u u b.
Write:<bin2 mor word part pass> = m u k a t t a b.
Write:<bin3 mor word part pass> = m u k a a t a b.
Write:<bin4 mor word part pass> = m u ? a k t a b.
Write:<bin5 mor word part pass> = m u t a k a t t a b.
Write:<bin6 mor word part pass> = m u t a k a a t a b.
Write:<bin7 mor word part pass> = m u n k a t a b.
Write:<bin8 mor word part pass> = m u k t a t a b.
Write:<bin10 mor word part pass> = m u s t a k t a b.
```

5 Inheritance-based Approach to Arabic Verbal Root-and-Pattern Morphology

Salah R. Al-Najem

Department of Arabic, Faculty of Arts, Kuwait University, P.O. Box: 23558, Safat, Kuwait
alnajem@arts.kuniv.edu.kw

Abstract: This chapter introduces a computational approach to the derivation and inflection of Arabic verbs. The approach attempts to capture generalizations, dependencies, and syncretisms existing in Arabic verbal morphology in a compact non-redundant manner. The approach is represented by a computational implementation in DATR lexical knowledge representation language. Generalizations will be captured through the use of Default Inheritance technique. Dependencies will be handled through the use of Multiple Inheritance technique. Default Inference technique will be used to capture syncretisms

5.1 Introduction

Arabic morphology is a system governed by a number of generalizations, dependencies, and syncretisms that explain the overt aspects of regularity in the derivational and inflectional structure of Arabic. A generalization is a statement about the facts of a language, which holds true in all cases or in nearly all cases (Trask 1993). A dependency is a case in which a (dependent) form is derived from another form. Finally, a syncretism is the case in which two or more morphemes that are morphosyntactically distinct appear identical in form.

In the context of implementing Arabic morphology computationally, these generalizations, dependencies, and syncretisms should be taken into consideration. Capturing such generalizations, dependencies, and syncretisms in a computational implementation of Arabic morphology can save that implementation from redundancy and can make it more compact, which is a vital issue in linguistics and computer science. This chapter introduces a computational approach to Arabic verbal morphology in which these generalizations, dependencies, and syncretisms are used to systematize Arabic in a concise manner. The approach is represented by an implementation written in the DATR lexical knowledge representation language.

A. Soudi, A. van den Bosch and G. Neumann (eds.), Arabic Computational Morphology, 67–88.

5.2 Linguistic Data

Verbs in Arabic are formed from roots consisting of three or four letters (known as radical letters). From these roots, verbal stems are constructed using a number of canonical forms known as measures. Measures are sequences of consonants and vowels that represent word structure. They may also contain stem derivational affixes. Each measure is normally associated with perfective (active and passive), imperfective (active and passive), and imperative patterns, which are used to form perfective, imperfective, and imperative verbal stems. The perfective verbs indicate a completed act, while imperfective verbs denote an unfinished act, which is just beginning or in progress. Measures that are intransitive or express a state of being do not normally associate with passive voice inflection. The stems formed using the above patterns are used to construct verbs through prefixing and/or suffixing inflectional prefixes and/or suffixes.

Verbs in Arabic are either triliteral (having three radical letters) or quadriliteral (having four). The triliteral verbal stems are formed using fifteen verbal measures while the quadriliteral verbal stems use four. Table 5.1 shows seven of these measures with examples that demonstrate how verbal stems are constructed using them. The stems are given in this table without inflectional prefixes and suffixes. For the triliteral verbs, the root *ksr* (breaking) will be used as an example root, while the root *dHrj* (rolling) will be used for the quadriliteral verbs. Roman numbers are used for reference with the prefix Q denoting a quadriliteral verbal measure. It should be

Table 5.1. The verbal measures

No	Measure	Active Perfective	Passive Perfective	Active Imperfective and Imperative	Passive Imperfective
I	faς al	C_1aC_2aC_3	C_1uC_2iC_3	C_1C_2iC_3	C_1C_2aC_3
		kasar	kusir	ksir	ksar
II	faς~al	C_1aC_2~aC_3	C_1uC_2~iC_3	C_1aC_2~iC_3	C_1aC_2~aC_3
		kas~ar	kus~ir	kas~ir	kas~ar
III	faAςal	C_1aAC_2aC_3	C_1uwC_2iC_3	C_1aAC_2iC_3	C_1aAC_2aC_3
		kaAsar	kuwsir	kaAsir	kaAsar
IV	Âafςal	ÂaC_1C_2aC_3	ÂuC_1C_2iC_3	ÂaC_1C_2iC_3	ÂaC_1C_2aC_3
		Âaksar	Âuksir	Âaksir	Âaksar
V	tafaς~al	taC_1aC_2~aC_3	tuC_1uC_2~iC_3	taC_1aC_2~aC_3	taC_1aC_2~aC_3
		takas~ar	tukus~ir	takas~ar	takas~ar
QI	faςlal	C_1aC_2C_3aC_4	C_1uC_2C_3iC_4	C_1aC_2C_3iC_4	C_1aC_2C_3aC_4
		daHraj	duHrij	daHrij	daHraj
QII	tafaςlal	taC_1aC_2C_3aC_4	tuC_1uC_2C_3iC_4	taC_1aC_2C_3aC_4	taC_1aC_2C_3aC_4
		tadaHraj	tuduHrij	tadaHraj	tadaHraj

noted that not all measures are applicable with every root. The roots *ksr* and *dHrj*, for example, do not occur with some measures. For instance, **Âaksar* (Measure IV) is unacceptable in Standard Arabic while *kasar* (Measure I) is acceptable. It should also be noted that some stems undergo certain additional phonological processes. In addition, in this table, I will use CV arrays to represent the perfective, imperfective, and imperative stems (patterns) of each measure.[1]

Measure I (the stem *kasar*) is considered the basic measure (stem) from which the other 14 triliteral verbal stems are derived. The derivation is done throgh various modifications of the meaning associated with that basic stem. Similarly, measure QI (the stem *daHraj*) can be considered a basic form (stem) from which the remaining three quadriliteral verbal stems are derived.

So far, we have considered the derivation of verbs. We turn now to verb inflection. The inflection of verbs in Arabic is mainly achieved through the use of prefixes and suffixes denoting person, number, and gender. These non-stem prefixes and suffixes are affixed to the perfective, imperfective, and imperative stems, which are produced using the verbal measures from roots as shown in Table 5.1. The reader should note that further changes in stem structure, such as radical letter rejection, vowel rejection, and radical letter transformation, are also applied in the inflection process when we deal with some roots.

As mentioned above, Arabic verbs have perfective, imperfective, and imperative stems. The perfective verbs indicate a completed act, while imperfective verbs denote an unfinished act which is just commencing or in progress. Table 5.2 shows a partial inflection paradigm illustrating the perfective active, imperfective active, and imperative inflection for the second person masculine and feminine.

Table 5.2. The perfective, imperfective, and imperative paradigm

Perfective		Imperfective	Imperative
2nd Person Singular			
Mas	kasar-ta	ta-ksir-u	ksir[2]
Fem	kasar-ti	ta-ksir-iyn	ksir-iy
2nd Person Dual			
Mas	kasar-tumaA	ta-ksir-aAn	ksir-aA
Fem	kasar-tumaA	ta-ksir-aAn	ksir-aA
2nd Person Plural			
Mas	kasar-tum	ta-ksir-uwn	ksir-uw
Fem	kasar-tun~a	ta-ksir-na	ksir-na

[1] So a measure can actually produce up to four different stems.

[2] The imperfective stem *ksir* has the surface form *Ǎiksir* with the epenthetic *Ǎi* prefixed to it.

There are a number of generalizations that can be captured in relation to Arabic verbal inflections. These generalizations are:

(1) Generalizations about Verbal Inflectional Paradigms:

a) Perfective active and passive inflectional paradigms are constructed by suffixing a perfective inflectional suffix (like: {*-tu*}) to a perfective stem (like *kasar*). There are no perfective inflectional prefixes.
b) Imperfective active and passive inflectional paradigms are constructed by prefixing an imperfective inflectional prefix (like:{*ya-*}) to an imperfective stem (like: *staksir*) and suffixing an imperfective inflectional suffix (like {*-u*}).
c) Imperative inflectional paradigms are constructed by suffixing an imperative inflectional suffix (like: {*-aA*}) to an imperative stem (like: *staksir*). There are no imperative inflectional prefixes.
d) The imperative stems are the same as the imperfective active stems.
e) The imperfective active prefixes end with the vowel '*a*.' The imperfective passive prefixes are the same as the imperfective active prefixes except that the prefix vowel becomes '*u*.'

Besides the previous generalizations relating to the formation of verbal paradigms, there is a sub-generalization related to the inflectional paradigms that correspond to the verbal measures II, III, IV, and QI. The inflectional paradigms that correspond to these measures are constructed in the same way that the verbal inflectional paradigms mentioned in (1) are except that the imperfective active prefix will end with the vowel '*u*' instead of '*a*.' In other words, the vowel '*a*' of the imperfective active prefixes is changed to '*u*' when these prefixes are used with the imperfective active stems that consist of exactly two heavy syllables. A heavy syllable is a syllable with a branching nucleus (CVV) or a branching rime (CVC). These dual heavy syllable stems are those imperfective active stems produced using the measures, II, III, IV, and QI, such as the stem *kas~ir* (Measure II).

In addition, there are dependencies between verbal measures. There are, in fact, overt dependencies between the basic verbal measure I and most of the derived triliteral verbal measures. There are also overt dependencies between the basic quadriliteral verbal measure QI and most of the derived quadriliteral verbal measures. We can capture these dependencies by means of the following generalizations:

(2) Dependencies between Verbal Measures:

a) The perfective active and passive patterns (stems) of the triliteral derived measures are constructed with the perfective active and passive patterns (stems) of the basic triliteral measure I ($C_1aC_2aC_3$[3] and $C_1uC_2iC_3$) by

[3] The final syllable vowel (*a*) of Measure I perfective active pattern ($C_1aC_2\mathbf{a}C_3$) is changed to '*u*' or '*i*' according to stem root (see Table 5.1). For example, the perfective active stem formed from the roots *Hsn* and *ςlm* using Measure I are ***Hasun*** ($C_1aC_2\mathbf{u}C_3$) and *ςal**i**m* ($C_1aC_2\mathbf{i}C_3$) not * *Has**a**n* and * *ςal**a**m*.

changing the pattern (stem) portion that precedes the final C_2VC_3 syllable of these derived patterns (stems).

The imperfective active and passive patterns of the triliteral derived measures are constructed with the imperfective active and passive patterns of the basic triliteral measure I ($C_1C_2iC_3$[4] and $C_1C_2aC_3$) by changing the pattern portion that precedes the final C_2VC_3 syllable of these derived patterns.

b) The perfective active and passive patterns of the quadriliteral derived measures are derived from the perfective active and passive patterns of the basic quadriliteral measure QI ($C_1aC_2C_3aC_4$ and $C_1uC_2C_3iC_4$) by changing the pattern portion that precedes the final C_3VC_4 syllable of these derived patterns.

The imperfective active and passive patterns of the quadriliteral derived measures are derived from the imperfective active and passive patterns of the basic quadriliteral measure QI ($C_1aC_2C_3iC_4$ and $C_1aC_2C_3aC_4$) by changing the pattern portion that precedes the final C_3VC_4 syllable of these derived patterns.

The changes include the gemination of radical letters, the deletion of vowels, and the insertion of affixes. For example, the Measure II perfective active pattern C_1aC_2 ~aC_3 (such as *kas~ar*) and the Measure VII perfective active pattern $nC_1aC_2aC_3$ (such as *nkasar*) are derived from the perfective active pattern of the basic triliteral measure I $C_1aC_2aC_3$ (such as *kasar*), but at the same time geminating the second radical letter in C_1aC_2 ~aC_3 (*kas~ar*) and prefixing the derivational stem-prefix {*n*−} in $nC_1aC_2aC_3$ (*nkasar*). Notice here that the derived patterns share with the basic pattern the same final C_2VC_3 syllable (*sar*).

c) The *ta*-prefixed verbal measures (V, VI, and QII) also change the vowel of the final CVC syllable of the basic measure I/QI active voice imperfective pattern ($C_1C_2\mathbf{i}C_3$/ $C_1aC_2C_3\mathbf{i}C_4$) from '*i*' to '*a*' in the derived patterns. For example, we can get *takas~**ar*** (Imperf Act, Measure V) and *tadaHraj* (Imperf Act, Measure QII) but not **takas~**ir*** or **tadaHrij*.

The verbal inflectional paradigms also exhibit a number of syncretisms. To show these syncretisms, let us consider the perfective and imperfective paradigms. First, the following is the perfective paradigm formed from the root *ksr* using Measure II: First, Table 5.3 provides the perfective paradigm formed from the root *ksr* using Measure II. As the table shows, there are a number of syncretisms present in the suffixes:

- {*-tu*}, which indicates perfective first singular
- {*-naA*}, which indicates perfective first dual or plural
- {*-tumaA*}, which indicates perfective second dual

[4] The final syllable vowel (*i*) of the Measure I imperfective active pattern ($C_1C_2iC_3$) is changed to '*u*' or '*a*' according to stem root (see Table 5.1). For example, the imperfective active stem formed from the roots *ktb* and *ftH* using Measure I are *kt**u**b* ($C_1C_2\mathbf{u}C_3$) and *ft**a**H* ($C_1C_2\mathbf{a}C_3$) not * *ktib* and * *ftiH*.

Table 5.3. The perfective paradigm

	Mas	Fem
Perfective Active		
Singular		
1st	kas~ar-tu	kas~ar-tu
2nd	kas~ar-ta	kas~ar-ti
3rd	kas~ar-a	kas~ar-at
Dual		
1st	kas~ar-naA	kas~ar-naA
2nd	kas~ar-tumaA	kas~ar-tumaA
3rd	kas~ar-aA	kas~ar-ataA
Plural		
1st	kas~ar-naA	kas~ar-naA
2nd	kas~ar-tum	kas~ar-tun~a
3rd	kas~ar-uw	kas~ar-na
Perfective Passive		
Singular		
1st	kus~ir-tu	kus~ir-tu
2nd	kus~ir-ta	kus~ir-ti
3rd	kus~ir-a	kus~ir-at
Dual		
1st	kus~ir-naA	kus~ir-naA
2nd	kus~ir-tumaA	kus~ir-tumaA
3rd	kus~ir-aA	kus~ir-ataA
Plural		
1st	kus~ir-naA	kus~ir-naA
2nd	kus~ir-tum	kus~ir-tun~a
3rd	kus~ir-uw	kus~ir-na

There is also a partial syncretism expressed by the incorporated (connected) subject pronoun '*aA*,' which indicates the dual number. In this context, the incorporated pronoun '*aA*' is used as a complete suffix, as in {*-**aA***}, and as parts of suffixes, as in {*-at**aA***} and {*-tum**aA***}. The reader will note that the morphosyntactic feature lists, which correspond to the three suffixes, share the dual number. Another partial syncretism is expressed by the incorporated subject pronoun '*at*,' which indicates the third person singular feminine and which appears as a complete suffix ({*-**at***}) and as part of a suffix ({*-**at**aA*}).[5] A third partial syncretism in the perfective paradigm is expressed by '*t*,' which indicates the second person in the suffixes {*-**t**umaA*}, {*-**t**a*}, {*-**t**i*}, {*-**t**un~a*}, and {*-**t**um*}. All these suffixes share the second person.

[5] Notice that '*at*' constitutes a third person dual feminine suffix when it is used with the dual '*aA*' in the suffix {*-**at**aA*}.

Table 5.4. The imperfective paradigm

	Mas	Fem
Imperfective Active		
Singular		
1st	Âa-ksir-u	Âa-ksir-u
2nd	ta-ksir-u	ta-ksir-iyn
3rd	ya-ksir-u	ta-ksir-u
Dual		
1st	na-ksir-u	na-ksir-u
2nd	ta-ksir-aAn	ta-ksir-aAn
3rd	ya-ksir-aAn	ta-ksir-aAn
Plural		
1st	na-ksir-u	na-ksir-u
2nd	ta-ksir-uwn	ta-ksir-na
3rd	ya-ksir-uwn	ya-ksir-na
Imperfective Passive		
Singular		
1st	Âu-ksar-u	Âu-ksar-u
2nd	tu-ksar-u	tu-ksar-iyn
3rd	yu-ksar-u	tu-ksar-u
Dual		
1st	nu-ksar-u	nu-ksar-u
2nd	tu-ksar-aAn	tu-ksar-aAn
3rd	yu-ksar-aAn	tu-ksar-aAn
Plural		
1st	nu-ksar-u	nu-ksar-u
2nd	tu-ksar-uwn	tu-ksar-na
3rd	yu-ksar-uwn	yu-ksar-na

Syncretisms in the imperfective paradigm can be seen by considering the imperfective paradigm formed from the root *ksr* using Measure I, listed in Table 5.4. As this table shows, there are a number of syncretisms present in the prefixes:

- { *ÂV*-}, which indicates imperfective first singular
- {*nV*-}, which indicates imperfective first person
- {*tV*-}, which generally indicates imperfective second person
- {*yV*-}, which indicates imperfective third person.

Additionally, there are syncretisms among the imperfective suffixes, which are evident in the following:

- {-*u*} , which corresponds to half the imperfective morphosyntactic feature lists
- {-*aAn*}, which indicates imperfective dual
- {-*uwn*}, which indicates imperfective masculine plural
- {-*na*}, which indicates imperfective feminine plural

Table 5.5. The imperative paradigm

	Mas	Fem
Singular		
2nd	kas~ir	kas~ir-iy
Dual		
2nd	kas~ir-aA	kas~ir-aA
Plural		
2nd	kas~ir-uw	kas~ir-na

The above paradigm also shows that the incorporated subject pronouns '*aA*' (dual) and '*uw*' (Mas Plural) have been used as part of the imperfective suffixes {-***aA****n*} and {-***uw****n*}. Hence, they represent a kind of partial syncretism here.

Finally, we turn to the imperative paradigm. Table 5.5 lists the imperative inflectional paradigm formed from the root *ksr* using Measure II. In this paradigm, we observe that there is a syncretism represented by the suffix {-*aA*}, which indicates the imperative dual. We also observe, when we compare this paradigm with the perfective paradigm, that there are syncretisms between the imperative and perfective paradigms represented by the suffixes (incorporated subject pronouns) {-*uw*} (Mas Plural), {-*na*} (Fem Plural), and {-*aA*} (Dual), which correspond to the following morphosyntactic feature lists:

- {-*uw*}: Imp 2nd Plural Mas, Perf 3rd Plural Mas
- {-*na*} : Imp 2nd Plural Fem, Perf 3rd Plural Fem
- {-*aA*} : Imp 2nd Dual Mas/Fem, Perf 3rd Dual Mas

When we compare the imperative paradigm with the imperfective paradigm, we see that the incorporated subject pronouns '*aA*'and '*uw*' represent a kind of partial syncretism if we consider the imperative suffixes {-***aA***}and {-***uw***} on the one hand and the imperfective suffixes {-***aA****n*} and {-***uw****n*} on the other. The incorporated subject pronouns '*aA*'and '*uw*' in the imperative and imperfective suffixes indicate Dual and Mas Plural, respectively. We also notice that the imperative suffix (incorporated subject pronoun) {-*na*} (Imp 2nd Plural Fem) represents a syncretism with the imperfective suffix {-*na*} (Imperf 3rd Plural Fem). The two suffixes indicate Plural Fem.

5.3 The Computational Approach

In a good computational approach to Arabic verbal morphology, the above generalizations, dependencies, and syncretisms should be employed. Capturing such generalizations, dependencies, and syncretisms in a computational implementation of Arabic morphology can save the system from redundancy and make it more compact.

To show how such generalizations, dependencies, and syncretisms can be used, in this chapter, I will introduce a computational approach to Arabic verbal morphology, which is constructed with DATR.[6] DATR is a lexical knowledge representation language based on the inheritance technique.[7] The approach will capture the above generalizations, dependencies, and syncretisms about Arabic verbal morphology in a compact non-redundant manner. The approach adopts the multilinear formalization of Arabic morphology introduced by McCarthy (1981, 1982), which organizes Arabic verbal stem structure using multiple layers of representation known as tiers. This approach has been also applied in Al-Najem (1998) to capture generalizations, dependencies, and syncretisms about Arabic nominal morphology using DATR.

The abovementioned generalizations will be handled in the implementation by encoding the generalizations once they are in generalization nodes higher in the inheritance hierarchy (inheritance network). Then, using default inheritance, other nodes lower in the inheritance hierarchy inherit, by default, information from these generalization nodes, and override, in some cases, some of the inherited default information. Thus, nodes lower in the inheritance hierarchy will inherit generalization information, which has been encoded once higher in the hierarchy, without re-encoding this information.

The dependencies between verbal measures will be systematized through stem partitioning and through the multiple inheritance technique. In this context, a node in this implementation that represents a derived (dependant) measure will inherit a shared stem portion (the final CVC syllable) from the node representing its corresponding original measure. The node representing the derived form defines the remaining portion, which precedes the final CVC shared portion.

Syncretisms in inflectional paradigms are handled mainly using default inference technique. Using a default interference technique, the implementation can infer, from a single (usually underspecified) statement about the inflectional paradigm, more statements about that paradigm. These statements are implicitly inferred, by default, from the original single statement without re-encoding them explicitly.

The following is a demonstration of how generalizations are handled in the implementation. The abovementioned generalizations about verb inflection will be encoded in our implementation in the node `Verb`, which contains the following information

```
Verb:

<$vform syn cat> == verb

<$vform mor stem> == "<$vform p l>" "<$vform p r>"

<$vform mor stem imp> == "<$vform mor stem imperf act>"

<$vform mor> == "<$vform mor stem>" Verb_Infl_Aff:<mor
```

[6] For more details about the computational approach introduced in this chapter and for details on other computational approaches to Arabic morphology, see Al-Najem (1998).

[7] For an introduction to DATR, see Evans (1990) and Evans and Gazdar (1990, 1996).

```
suf>

<$vform mor imperf> == Verb_Infl_Aff:<mor pre imperf>

Imperf_Pre_V:<a> "<$vform mor stem imperf>"

Verb_Infl_Aff:<mor suf imperf>

<$vform mor imperf pass> == Verb_Infl_Aff:<mor pre

imperf pass> Imperf_Pre_V:<u> "<$vform mor stem imperf

pass>" Verb_Infl_Aff:<mor suf imperf pass>.
```

Notice here that the imperfective prefix, in this implementation, consists of an imperfective prefix consonant inherited from the node `Verb_Infl_Aff` and an imperfective prefix vowel inherited from the node `Imperf_Pre_V`. This allows for the generalization that the imperfective active prefix vowel is normally '*a*' but is changed to '*u*' when imperfective active prefixes are used with imperfective active stems of Measures II, III, IV, and QI. In addition, as previously indicated, the passive imperfective prefixes are the same as the active imperfective prefixes except that the vowel '*u*' is used instead of '*a*.' The imperfective prefix begins with a consonant (*'n', 'Â', 'y'*, or '*t*'), which has no variation with respect to a measure or a voice (imperfective active/passive). The variation in the imperfective prefixes occurs just to the prefixes' vowel ('*a*' or '*u*'), as we have already seen, according to the measure and the voice. Thus, instead of explicitly encoding the prefixes with the two variations of vowel, which yields two sets of prefixes: {*na, Âa, ya, ta*} and {*nu, Âu, yu, tu*}, we have only encoded the consonants of the prefixes in the node `Verb_Infl_Aff`, and the vowel is chosen by inheritance from the node `Imperf_Pre_V`.

The sixth statement of `Verb` partially overrides the third statement, declaring that the imperfective passive inflection paradigms are constructed by prefixing an imperfective passive prefix ending with the vowel '*u*' to an imperfective passive stem and suffixing an imperfective suffix. The fifth and sixth statements, using default interference, provide the imperfective active and passive paradigms. In other words, they allow the generation of all the 36 active and passive imperfective verb forms.

The node `Verb`, which encoded generalizations about verbal paradigm formation, occurs high in the inheritance hierarchy. It will be inherited, by default, several times by other nodes lower in the inheritance hierarchy without re-encoding the generalizations of `Verb` again in those nodes. Deault inheritance permits us to encode these linguistic generalizations in a compact, non-redundant manner. Note also that the use of default inference in the last three statements of `Verb` is crucial to allow the generation of 78 verb forms, that is, virtually all the Arabic morphology verb forms.

The lower nodes inheriting from `Verb` represent measures, and these nodes produce the inflectional stems[8] that correspond to each measure from radical letters inherited globally from root nodes. These stems will be used in the formation of the inflectional paradigms. A difficulty could arise here. The node `Verb` states that the (default) way of constructing the imperfective active paradigm is by prefixing an imperfective active prefix ending with the vowel '*a*' to an imperfective active stem and suffixing an imperfective suffix. This holds for most of the measures. However, some measures construct the imperfective active paradigm by prefixing an imperfective active prefix ending with the vowel '*u*' instead of '*a*.' These are Measures II, III, IV, and QI, as previously stated. The use of the vowel '*u*' in imperfective active prefixes represents a sub-generalization (sub-regularity) emerging from the generalizations (regularities) of the node `Verb`, which encoded the generalizations previously stated in (1). So, we need a way to define this sub-generalization in a node that can be inherited by specific nodes representing Measures II, III, IV, and QI. This is achieved by defining an intermediate node[9] (`Verb2`), which inherits, by default, all the information of the default (regular) node `Verb` but overrides the inherited statement that encodes the imperfective active paradigm by changing the original prefix vowel '*a*' to '*u*.'

```
Verb2:

<> == Verb

<$vform mor imperf> == Verb_Infl_Aff:<mor pre imperf>

Imperf_Pre_V:<u> "<$vform mor stem imperf>"

Verb_Infl_Aff:<mor suf imperf>.
```

Thus, while the node `Verb` represents the default way of constructing the verbal paradigms, `Verb2` represents the sub-generalization (sub-regularity) that is related to the vowel of the imperfective active prefixes used with the imperfective active stems of Measures II, III, IV, and QI.

So far, we have considered how generalizations about verb formation can be encoded in our implementation. Now, we turn to generalizations relating to the dependencies between verbal measures. Recall in this context, as stated in (2), that the patterns of most of the triliteral verbal measures are normally derived from the corresponding patterns of the triliteral verbal measure I by means of changing the pattern portion that precedes the final C_2VC_3 syllable of these derived patterns. In addition, most of the patterns of the verbal quadriliteral measures are derived from the corresponding patterns of the verbal quadriliteral measure QI by changing

[8] Examples of these inflectional stems are the imperfective active and imperfective passive stems.

[9] The use of intermediate nodes is a common way adopted in DATR literature to capture sub-regularities. See Corbett and Fraser (1993) for example.

the pattern portion that precedes the final C_3VC_4 syllable of these derived patterns. This means that to capture the dependencies between the verbal measures, we need to split the verbal stem (pattern) into two parts: the final CVC syllable, which is normally invariably shared between the stems (patterns) of the derived and the original measures and a remainder, which is the part of the stem (pattern) preceding the final CVC syllable.[10] To achieve this, the verbal stem in our implementation will consist of two parts: a right part that represents the final CVC syllable of the stem and a left part that represents the remainder of the stem preceding the final CVC syllable.

To demonstrate how the dependencies between verbal measures are captured in our implementation, we take the following example, involving the dependencies between Measures III (*faAçal*) and VIII (*ftaçal*) on the one hand and the basic verbal Measure I (*façal*) on the other hand. The following is a definition of the node `Façal`, which represents the basic verbal Measure I:

```
% The CV-patterns of the Measure I (façal)

Façal:

<> == Verb

<façal p l> == C:<1> V:<>

<façal p l imperf> == C:<1>

<façal p r> == C:<2> V:<*> C:<3>.
```

The node V below encodes the consonants of a root through global inheritance from nodes representing roots, and the node C encodes perfective and imperfective vocalisms:

```
%The Vocalism Tier
V:

<perf act> == Perf_Act:<>

<perf pass> == Perf_Pass:<>

<* perf act> == <perf act *>

<* perf pass> == <perf pass *>
```

[10] This remainder is not always a single complete free standing syllable. It may become part of a syllabic structure through the insertion of the epenthetic '*ÂV*,' as in **(Âi)nC$_1$a** C_2aC_3 (Measure VII, Perf Act), which yields the syllabic structure **CVCCV**. It may also become part of a syllabic structure by the prefixation of an inflectional prefix like {*ya-*} as in **(ya)C$_1$** C_2iC_3 (Measure I, Imperf Act), which yields the syllabic structure CVC.

```
<imperf act> == Imperf_Act:<>

<imperf pass> == Imperf_Pass:<>

<* imperf act> == <imperf act *>

% "<* ... >" represents a terminal non-default vowel


%Vocalism Generalizations

Perf_Act:

<> == a.

Perf_Pass:

<> == u

<*> == i.


Imperf_Act:

<> == a

<*> == i.



Imperf_Pass:

<> == a.




% The association of the root tier elements (radical

% letters) to the C slots of the pattern tier

# vars $n: 1 2 3 4.

C:

<$n> == "<$n>".
```

The node `Façal` encodes the right parts (final CVC syllables) and the left parts, which correspond to the perfective and imperfective stems (patterns) of Measure I. Notice also that we have not encoded the right and left parts of the imperative stems since the imperative and imperfective active stems are the same as is indicated by the statement (generalization):

```
<$vform mor stem imp> == "<$vform mor stem imperf act>"
```

of the node `Verb`, which is inherited, by default, by the current node. This generalization about imperative stems has been encoded once in `Verb` higher in the inheritance hierarchy, and it is then inherited by the measure nodes lower in the hierarchy. The right and left parts of the perfective and imperfective stems are combined together to form complete perfective and imperfective stems by one statement encoded once in the node `Verb` higher in the inheritance hierarchy, which is inherited, by default, by Façal and the measure nodes lower in the hierarchy:

```
<$vform mor stem> == "<$vform p l>" "<$vform p r>"
```

This line states that a verbal stem consists of a left part followed by a right part. Default inference will fill the gaps, allowing the formation of all the perfective and imperfective stems. For instance, this single, short, and general line will (implicitly) stand for longer more specific lines like the following, which constructs a perfective active stem:

```
<$vform mor stem perf act> == "<$vform p l perf act>"

"<$vform p r perf act>"
```

Then, the node `FaAçal` inherits the right parts (final CVC syllables) of `Façal`, but it defines its left parts in a different way:

```
% The CV -- patterns of the Measure III (faAçal)

FaAçal:

<> == Verb2

<faAçal p r> == Façal:<façal p r>

<faAçal p l> == C:<1> V:<> V:<>.
```

The underspecified line:

```
<faAçal p l> == C:<1> V:<> V:<>
```

states, using default inference, that the (default) left part of all the perfective and imperfective stems (patterns) of `FaAçal` is constructed as the first radical letter of a root followed by a long vowel. The vowel is inherited from the vocalism node `V`.

In addition, the node `FaAçal` inherits, by default, the generalizations encoded in `Verb2`.[11] The reader will, therefore, notice that in this node, we adopt a multiple inheritance by allowing the node `FaAçal` to inherit from two nodes, `Verb2` and `Façal`.

Thus, in the node `FaAçal`, we have captured the dependency between Measures III (*faAçal*) and I (*façal*) using multiple inheritance. We inherited the generalizations about verbs from `Verb2` and inherited the invariant right parts (final CVC syllables) that are shared with Measure I from the node `Façal`. Measure III (*faAçal*) is derived from I (*façal*) except that the left parts of the stems of *façal* are changed to C_1aA (*faA*) (and C_1uw in the perfective passive). In other words, the dependency generally involves the lengthening of the first vowel. Notice here that since we have encoded verb generalizations once in the higher node `Verb`, from which the intermediate node `Verb2` inherits, we do not need to re-encode these generalizations again in the node `FaAçal`. We have simply inherited these generalizations, by default, from `Verb2`. Notice also that we have not re-encoded the right parts (the final CVC syllables) of the stems of `FaAçal` since we encoded these parts once in `Façal` and passed them on from this node. Defining the left parts of `FaAçal` is accomplished in a compact non-redundant manner by using default inference through underspecifying the left part using the statement:

```
<faAçal p l> == C:<1> V:<> V:<>
```

This means that this statement, using default inference, will represent all the left parts of all the perfective and imperfective stems of `FaAçal`. Thus, the dependency between measures III (*faAçal*) and I (*façal*) has been captured by means of multiple inheritance in a compact non-redundant manner, without the need to repeat encoding information which has previously been encoded in higher nodes.

In a similar fashion, the dependency between Measure VIII (*ftaçal*) and I (*façal*) can be captured:

```
% The CV-patterns of the Measure VIII (ftaçal)

Ftaçal:

<> == Verb

<ftaçal p r> == Façal:<façal p r>

<ftaçal p l> == C:<1> Affix:<t> V:<>.
```

This node illustrates that the stems of Measure VIII (*ftaçal*) are derived from the stems of Measure I (*façal*) by inserting the infix $\{t\}$ after the first radical letter in the

[11] As previously seen, the node `Verb2` is a sub-generalization intermediate node, which inherits, by default, the information of the generalization node `Verb`.

left part of the stems of *ftaçal*. The node `Affix` defines derivational affixes used in the CV-pattern tier and is defined below:

```
Affix:

<t> == t

<n> == n

<st> == s t

<Â> == Â

<w> == w.
```

The affixation of derivational affixes is indicated in the pattern tier by inheritance descriptors like `Affix:<t>`, which indicates (inherits) the reflexive infix $\{t\}$.

The other dependencies between verbal measures are captured using a similar method to that used with the dependencies between the previous verbal measures. The inheritance hierarchy below demonstrates how the dependencies between verbal measures are captured in our implementation:

Note that some derived measures inherit from multiple nodes (multiple inheritance). Multiple inheritance is indicated in this diagram by having a derived measure node which inherits from more than one node (receiving more than one IN arrow).

So far, we have considered how generalizations relating to verb formation and generalizations about the dependencies between verbal measures are captured in our implementation. We will now discuss syncretisms in inflectional paradigms and how these are encoded in our DATR implementation.

The verbal inflectional paradigms are handled in our implementation by means of the node `Verb_Infl_Aff`. In this node, we define paths representing morphosyntactic feature lists and assign these paths values representing inflectional affixes (prefixes and suffixes) that correspond to the paths' morphosyntactic feature lists, as we will see shortly. The inflectional paradigms themselves are built through the generalization nodes `Verb` and `Verb2` using statements like the following statements taken from the definition of the node `Verb`:

```
<$vform mor> == "<$vform mor stem>" Verb_Infl_Aff:<mor

suf>

<$vform mor imperf> == Verb_Infl_Aff:<mor pre imperf>

Imperf_Pre_V:<a> "<$vform mor stem imperf>"

Verb_Infl_Aff:<mor suf imperf>

<$vform mor imperf pass> == Verb_Infl_Aff:<mor pre
```

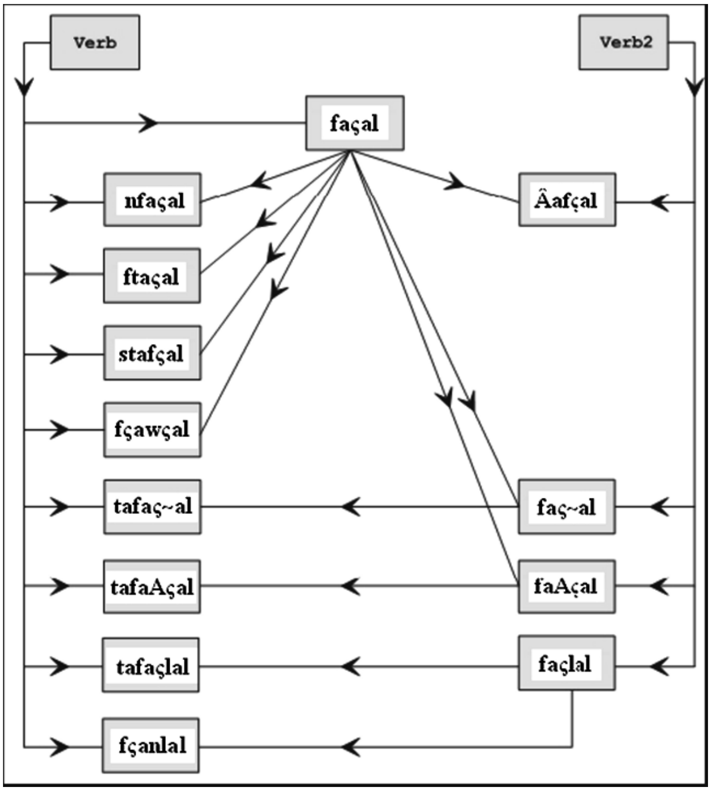


Fig. 5.1. An inheritance hierarchy showing dependencies between verbal measures

```
imperf pass> Imperf_Pre_V:<u> "<$vform mor stem imperf

pass>" Verb_Infl_Aff:<mor suf imperf pass>
```

These three statements, by inheriting the affixes from `Verb_Infl_Aff` and especially through default inference, allow the construction of the whole verbal paradigm of Arabic morphology in a compact, non-redundant manner, as we have already seen from consideration of the node `Verb`.

In our implementation, syncretisms in paradigms are handled mainly using default inference. Default inference represents a useful means of expressing generalizations and encoding repeated information in a compact non-redundant manner. It saves us from re-encoding information which can be inferred, by default, from one (usually underspecified) path. Consider in this context the statement:

```
<mor suf imperf act> == u
```

of the node `Verb_Infl_Aff`, which is the node where inflectional verbal affixes of the inflectional paradigms were encoded. From this single statement, which is underspecified for person, number, and gender, we can infer that, roughly, the default suffix of the imperfective active paradigm is {-*u*}. This is because the suffix {-*u*} is associated with half of the morphosyntactic feature lists of the imperfective active paradigm. Then, we partially override this statement to encode the other imperfective active suffixes, which are associated with the remaining morphosyntactic feature lists of the imperfective active paradigm. From the previous single path (`<mor suf imperf act>`), we can infer, by default, other more specific paths, which include the following:

```
<mor suf imperf act first sing mas>

<mor suf imperf act first dual mas>

<mor suf imperf act first plural mas>
```

These paths are implicitly inferred, by default, from the previous single underspecified path without re-encoding them explicitly. This is a better method than others like local inheritance from (referral to) a specific local path, as in the following:

```
<mor suf imperf act first sing mas> == u

<mor suf imperf act first dual mas> == <mor suf
imperf act

first sing mas>

<mor suf imperf act first plural mas> == <mor suf
imperf act

first sing mas>
```

which is not a compact way in comparison to default inference.

Thus, the previous example clearly shows how default inference represents a useful means to capture syncretisms in paradigms in a compact non-redundant manner. Using a single short (usually underspecified) path, we can infer, by default, multiple longer more specific paths without re-encoding these inferred paths and their values explicitly.

As previously noted, there are some partial syncretisms in the verbal inflectional paradigms, which can be captured in the implementation. Partial syncretisms mainly concern incorporated subject pronouns, which materialize as part of suffixes or as complete suffixes. To handle partial syncretisms in our implementation, we encoded (generalizations about) such incorporated pronouns once in a node called `Part_Sync`:

```
Part_Sync:

<dual> == a A
```

```
<plural mas> == u w

<plural fem> == n a

<sing fem> == a t

<second> == t.
```

Using this node, such incorporated pronouns will not be re-encoded explicitly in the node `Verb_Infl_Aff`, which encodes the syncretism in verbal inflectional paradigms. Instead, we refer to the node `Part_Sync` to inherit those incorporated pronouns, which represent partial syncretisms. An example that shows this is the following statement from the node `Verb_Infl_Aff`:

```
<mor suf imperf act $2nd_3rd $dual_plural> ==

Part_Sync:<$dual_plural> n
```
[12]

Through this statement, the incorporated pronouns '*aA*' (Dual) and '*uw*' (Plural Mas) are inherited from `Part_Sync`. The incorporated pronouns '*aA*' and '*uw*' are parts of the suffixes {***-aA**n*} and {***-uw**n*} associated with the morphosyntactic feature lists :

- {*-aAn*} :

Imperf Act 2nd Dual Mas, Imperf Act 3rd Dual Mas, Imperf Act 2nd Dual Fem, and Imperf Act 3rd Dual Fem.

- {*-uwn*} :

Imperf Act 2nd Plural Mas, and Imperf Act 3rd Plural Mas.
The previous statement encodes this using the node `Part_Sync` and default inference. This statement is partially overridden to encode the other suffixes which are associated with the imperfective active second singular feminine in addition to the imperfective active second and third plural feminine. At the same time, the incorporated pronouns '*aA*' and '*uw*' (and other incorporated pronouns) also form complete suffixes, as can be seen in the following statement from the node

```
Verb_Infl_Aff:

<mor suf perf act third> == Part_Sync:<>
```

[12] The variables `$2nd_3rd` and `$dual_plural` have the following definitions, respectively:

```
# vars $2nd_3rd: second third.
# vars $dual_plural: dual plural.
```

Using this single statement, the incorporated pronouns '*aA*' and '*uw*', in addition to other incorporated pronouns, will form complete perfective active third person suffixes corresponding to morphosyntactic feature lists, including Perf Act 3rd Dual Mas and Perf Act 3rd Plural Mas. The single underspecified paths of that statement will stand for paths such as:

```
<mor suf perf act third dual mas> == Part_Sync:
<dual mas>

<mor suf perf act third plural mas> == Part_Sync:
<plural mas>
```

in addition to others using default inference.

In addition to the above nodes, which define how verbal stems and inflectional paradigms are constructed, the implementation contains other nodes known as the lexical nodes. The lexical nodes are those representing roots and defining the radical letters of these roots. The radical letters will be inherited, using global inheritance, by other nodes in the implementation representing verbal measures to form verbs. The lexical nodes also inherit (select) verbal measures, which can be used to form verbs using the roots represented by those lexical nodes. These nodes are the query nodes of our implementation, and they represent the initial (global) context of inheritance. The queries that will be entered into the implementation will be queries about these lexical nodes. An example of the lexical nodes is the node `KSR`, which represents the root *ksr*.

```
KSR:

<c1> == k

<c2> == s

<c3> == r


% Verb forms

<façal> == Façal

<faç~al> == Faç~al

<nfaçal> == Nfaçal

<tafaç~al> == Tafaç~al
```

So, using a node like this one and through the use of the other abovementioned nodes, we can enter queries like the following and obtain their corresponding results:

```
| ? datr_theorem ('KSR', [façal, mor, imperf, act,
third, dual, mas]).
yaksiraAn
```

```
| ? datr_theorem ('KSR' [faç~al, mor, perf, act, first,
sing, mas]).
kas~artu
```

```
| ? datr_theorem ('KSR' [tafaç~al, mor, perf, act,
third, sing, mas]).
takas~ara
```

```
| ? datr_theorem ('KSR' [façal, mor, perf, pass, third,
sing, mas]).
kusira
```

```
| ? datr_theorem ('KSR', [faç~al, mor, imp, second,
 sing, fem]).
kas~iriy
```

DATR was originally used for generation (synthesis) only, not for recognition (analysis). Information is obtained from DATR theories via queries that cause DATR to *generate* results which take forms like theorem dumps.

5.4 Conclusion

In this chapter, we have computationally systematized generalizations, dependencies, and syncretisms existing in Arabic verbal morphology in a compact, non-redundant manner. Generalizations have been handled through the use of DATR default inheritance, as seen in the node `Verb`. Dependencies have been handled through the use of DATR multiple inheritance, as seen in nodes like `Façal` and `FaAçal`. Finally, syncretisms have been handled through the use of DATR default inference, as seen in the node `Verbs_Infl_Aff`. As has been mentioned earlier, capturing such generalizations, dependencies, and syncretisms in a computational implementation of Arabic morphology can save that implementation from redundancy and can make it more compact.

References

Al-Najem, Salah. "*Computational Approaches to Arabic Morphology*." Diss. Essex University, 1998.

Corbett, Greville and Norman Fraser. "Network Morphology: a DATR A Account of Russian Nominal Inflection." *Journal of Linguistics* 29 (1993): 113–142

Evans, Roger. "An Introduction to the Sussex PROLOG DATR System." *The DATR Papers*. Eds. Roger Evans and Gerald Gazdar . Brighton: University of Sussex, 1990. 63–70.

Evans, Roger and Gerald Gazdar, eds. *The DATR Papers*. Vol. 1. Brighton: University of Sussex, 1990.

Evans, Roger and Gerald Gazdar. "DATR: A Language for Lexical Knowledge Representation." *Computational Linguistics* 22.2 (1996): 167–216.

McCarthy, John. "A Prosodic Theory of Nonconcatenative Morphology." *Linguistic Inquiry* 12 (1981): 373–418.

McCarthy, John. "*Formal Problems in Semitic Phonology and Morphology*." Diss. Indiana University, 1982.

Trask, R. L. *A Dictionary of Grammatical Terms in Linguistics*. London: Routledge, 1993.

6

Arabic Computational Morphology: A Trade-off Between Multiple Operations and Multiple Stems

Violetta Cavalli-Sforza[1] and Abdelhadi Soudi[2]

[1] *Language Technologies Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15217, U.S.A.*
violetta@cs.cmu.edu

[2] *Ecole Nationale de l'Industrie Minerale, Avenue Hadj Ahmed Cherkaoui, B.P. 753, Agdal, Rabat, Morocco.*
asoudi@gmail.com

Abstract: We present a computational approach to Arabic morphology description that draws from Lexeme-Based Morphology (Aronoff, 1994; Beard, 1995), giving priority to stems and granting a subordinate status to inflectional prefixes and suffixes. Although the morphology of Arabic is non-concatenative, we make the process of generating inflected forms concatenative by separating the generation of stems from that of other inflectional affixes. Our approach is implemented in an extension of the MORPHĒ tool (Leavitt, 1994), which has been enhanced in order to provide a representational formalism that embodies Lexeme-Based Morphology theory and minimizes the number of rules required for the description of Arabic morphology

6.1 Introduction

The morphology of Arabic poses special problems for computational natural language processing systems. The exceptional degree of ambiguity in the writing system, the rich morphology, and the unique word formation process of roots and patterns all contribute to making computational approaches to Arabic very challenging. The non-concatenative morphology of Arabic has spurred the development of sophisticated formalisms and computational engines, as well as produced brute force approaches. Among the more elegant formalisms, the finite-state models proposed by (Koskenniemi, 1983), (Kay, 1987), (Beesley, 1996, 1998) and (Kiraz, 1994, 1998, 2000), based on the autosegmental approach of (McCarthy, 1979, 1981), distinguish themselves for their effective computational implementation, but have essentially granted equal status to all the constituents of an Arabic word (the root, the pattern and the vocalism) by placing them in separate lexicons. Our approach differs from these in giving priority to stems in accordance with the theory of Lexeme-Based Morphology (Aronoff, 1994; Beard, 1995) and not treating inflectional prefixes and suffixes as fully-fledged dictionary entries. Using the stem as a basis for morphological transformations we linearize, i.e., make concatenative, the process of morphology generation, by separating the generation of the correct stem from the generation of inflectional prefixes and suffixes. Based on linguistic evidence, we place regularly predictable morphological transformations in rules and non-predictable lexeme-specific information in the lexicon.

We describe the implementation of our approach in the context of MORPHĒ, a morphological description tool (Leavitt, 1994), which we have enhanced with the objective of providing a representational formalism that embodies Lexeme-Based Morphology theory and minimizes the number of inflectional rules required for the description of Arabic morphology. Ongoing enhancements to the tool, and primarily the introduction of inheritance, reduce the representation of the two-step linearization model to a single step, making the morphological description more elegant and increasing system performance.

6.2 Lexeme-Based Morphology

In this section, we briefly outline the premises of the theory of Lexeme-Based Morphology (LBM) (Aronoff, 1994; Beard, 1995) on which our computational model is based. LBM comprises several claims about the nature of morphology and its relation to syntax. The most central claim is the collection of theses that fall under the rubric of the Separation Hypothesis. A standard morphological analysis would claim that a word such as 'cats' consists of two morphemes: the base 'cat' and the plural morpheme '-s'. The base and the plural morpheme would have the following representations:

(i) cat: *<[+ Noun], /kaet/, CAT*, where the first entry represents a set of grammatical features, the second a phonological representation and the third a set of semantic features.
(ii) –s: *<[+ Noun], +PL, /z/, PL*

The basic thesis is that all morphemes, be they **lexemes** such as [*cat*] or grammatical morphemes such as the plural morpheme, are lexical entries consisting of grammatical, phonological and semantic information, that is, minimal grammatical units of analysis.

In a lexeme-based model, however, only lexemes (vocabulary items belonging to the major lexical categories of verb, noun, adjective and adverb) and free morphemes (e.g. detached pronouns) are minimal grammatical elements. Inflectional or derivational morphemes – suffixes, prefixes, infixes and reduplication – are not themselves grammatical elements. Instead, these are merely the phonological expression of operations that apply to basic grammatical elements. For example, given the forms 'cat' and 'cats', we would say that there is a lexeme [cat] that has two word forms 'cat' and 'cats' and that the description of the singular/plural of [cat] is a grammatical word. It follows that affixes and other grammatical morphemes differ from major class lexical stems. The proposed model of Arabic morphology is compatible with this conclusion in that it does not include affixes and lexemes in the same component.

6.3 Arabic Verbal Morphology

The most puzzling problem in the study of Arabic is its verbal system, which is very rich in forms. Arabic verbs are based on three or four radicals (the letters that constitute the skeleton of the verb). An Arabic verb can be conjugated according to one of the traditionally recognized Forms or patterns. There are 15 triliteral patterns (i.e., with 3 radicals), of which at least 9 are in common, and 4 less common quadriliteral patterns (i.e., with 4 radicals), with some quite rare. Within each pattern, an entire paradigm is found: two aspects/tenses (perfect and imperfect), two voices (active and passive) and five moods (indicative, subjunctive, jussive, imperative and energetic). In addition to prefixation and suffixation, inflectional and derivational processes may cause stems to undergo infixational modification in the presence of different syntactic features as well as certain consonants. Verb roots in Arabic can be classified as shown in Fig. 6.1. A primary distinction is made between weak and strong verbs. Weak verbs have a weak consonant ('*w*' or '*y*') as one of their radicals. A handful of doubly weak verbs contain two weak radicals.

In this section, we address two main issues in Arabic verbal morphology. The first relates to weak (**suppletive**) verbs, which undergo stem changes. Weak consonants

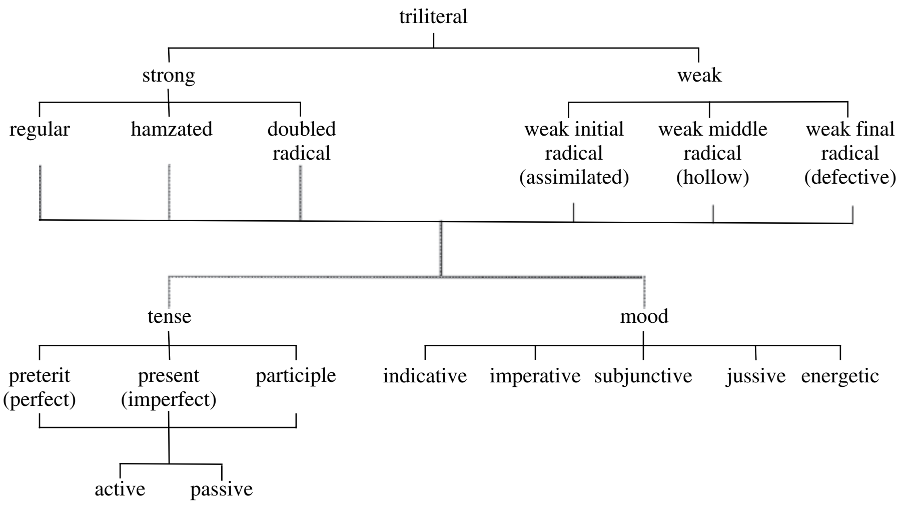


Fig. 6.1. Classification of Arabic triliteral verbal roots and mood tense system

are quite common in triliteral verb patterns, but they do not always cause morphological irregularities. They are seldom present in quadriliteral patterns except for in the second radical position, where they do not give rise to morphological irregularities (Badawi et al., 2004). The second issue relates to apophony attested in Arabic Form 1 verbs. We conclude by describing our generation model and how Arabic verbal morphology is implemented in an enhanced version of MORPHĒ (Leavitt, 1994; Cavalli-Sforza & Soudi, 2003).

6.3.1 Weak Verbs

We focus on triliteral hollow verbs, but a comparable analysis is applicable to other types of weak verbs as well. In both perfect and imperfect tenses, depending on the person, number and gender, hollow verbs are realized by two stems: one long, with a weak middle radical (glide), and one short, where the glide disappears and there is a vowel change.

Hollow Verb Classes:

1. Verbs of the pattern *CawaC* have the perfective stem patterns C$\underline{u}$C and *CuwC* and the imperfective stem patterns *CuC* and *CuwC*. For example, *zaAr* (from [*zawar*]) "to visit" has the perfective *zur* and *zaAr* and the imperfective stems *zur* and *zuwr*. E.g.:
   PERFECT: ***zu***rtu "I visited" and ***zaAr**at* "she visited"
   IMPERFECT: *ya**zur**na* "they (fem.) visit" and *ya**zuwr**u* "he visits"
2. Verbs of the pattern *CawiC* have the perfective stem patterns *CiC* and *CaAC* and the imperfective stem patterns *CaC* and *CaAC*. For example, *naAm* (from [*nawim*]) "to sleep," has the perfective stems *nim* and *naAm* and the imperfective stems *nam* and *naAm*. E.g.:
   PERFECT: ***nim**tu* "I slept" and ***naAm**at* "she slept"
   IMPERFECT: *ya**nam**na* "they (fem.) sleep" and *ya**naAm**u* "he sleeps"
3. Verbs of the pattern *CayaC* have the perfective stem patterns *CiC* and *CaAC* and the imperfective stem patterns *CiC* and *CiyC*. For example, *baAς* (from [*bayaς*]) "to sell" has the perfective *biς* and *baAς* and the imperfective stems *biς* and *biyς*. E.g.:
   PERFECT: ***biς**tu* "I sold" and ***baAς**at* "she sold"
   IMPERFECT: *ya**biς**na* "they (fem.) sell" and *ya**biyς**u* "he sells"
4. Verbs of the pattern *CayiC* have the perfective stem patterns *CiC* and *CaAC* and the imperfective stem patterns *CaC* and *CaAC*. For example, *haAb* (from [*hayib*]) "to fear", has the perfective stems *hib* and *haAb* and the imperfective stems *hab* and *haAb*. E.g.:
   PERFECT: ***hib**tu* "I feared" and ***haAb**at* "she feared"
   IMPERFECT: *ya**hab**na* "they (fem.) fear" and *ya**haAb**u* "he fears"

In generation of the perfective and imperfective, classes 2 and 4 of the verb can be merged because they have the same perfect and imperfect stems. A different grouping is necessary for the passive participle: verbs of classes 1 and 2 behave similarly as do verbs of classes 3 and 4. E.g.:

Class 1: [*zawar*] → *mazuwr* "visited" and Class 2 [*nawil*] → *manuwl* "obtained"
Class 3: [*bayaς*] → *mabiyς* "sold" and Class 4: [*hayib*] → *mahiyb* "feared".

For details on the origin of the glides and the second vowel in the examples above see (Soudi et al., 2001, 2002).

To deal with weak verbs, one might list separate stems in the lexicon or one might derive the stems by rules. The latter solution is preferable because **suppletion** operates at the stem level and involves a single root consonant. It also permits capturing generalizations by providing a unified treatment of strong and weak verbs.

Table 6.1 shows that there are cases of syncretism (opportunities for generalization) in the perfective conjugation paradigms of both strong and hollow verbs. Some combinations of features have the same realization as some others; equivalently, certain inflected verbs have the same word form as others. The number of rules required to capture generalizations depends on how we handle syncretism in the paradigms of strong and hollow verbs: at the whole word form (the stem and the affixes) or simply at the stem level. In the perfective paradigm for non-hollow verbs in Table 6.1, there are three evident instances of syncretism. Such instances are expressed by the following rules of referral:

(i) The first person singular masculine and feminine word forms are identical;
(ii) The first person dual masculine or feminine and first person plural masculine or feminine word forms are identical;
(iii) The second person dual masculine and second person dual feminine word forms are identical.

Given these instances of syncretism, the 18 word forms resulting from the 18 person-number-gender combinations collapse to 13 forms.

Table 6.1. Perfective active conjugation of *katab* "to write" and *zawar* "to visit"

		Strong Verb		Hollow Verb	
Person	Number	Masculine	Feminine	Masculine	Feminine
	singular	*katab-tu*	*katab-tu*	*zur-tu*	*zur-tu*
1st	dual	*katab-naA*	*katab-naA*	*zur-naA*	*zur-naA*
	plural	*katab-naA*	*katab-naA*	*zur-naA*	*zur-naA*
	singular	*katab-ta*	*katab-ti*	*zur-ta*	*zur-ti*
2nd	dual	*katab-tumaA*	*katab-tumaA*	*zur-tumaA*	*zur-tumaA*
	plural	*katab-tum*	*katab-tun~a*	*zur-tum*	*zur-tun~a*
	singular	*katab-a*	*katab-at*	*zaAr-a*	*zaAr-at*
3rd	dual	*katab-aA*	*katab-ataA*	*zaAr-aA*	*zaAr-ataA*
	plural	*katab-uwA*	*katab-na*	*zaAr-uwA*	*zur-na*

Strong verbs show syncretism in the person-number-gender suffixes, with the stem remaining unchanged throughout. Whether we deal with syncretism at the word form level or simply at the stem level, the 13 forms are derivable by 13 rules that add the appropriate suffix to an invariable stem. Hollow verbs and other weak verbs, however, show two kinds of syncretism: in the person-number-gender suffixes and in the stems. By considering syncretism at the whole word form level, we would need 13 more rules (an additional rule for every person, number and gender combination yielding a distinct surface form), for a total of 26 rules, to account for the stem changes in Table 6.1. However, by handling syncretism at the stem level, that is, by separating stem generation from suffix generation, we reduce the 13 additional rules to 3. In the hollow verb paradigm shown in Table 6.1, the first person and second person word forms have the same stem. (i.e., *zur*). To capture this generalization most economically, we: 1) postulate a {+/− 3rd person} feature so that the first and second person can form a coherent class with its associated stem change rule; 2) postulate a default rule that applies to all third person-number-gender combinations; and 3) provide another more specific overriding rule, identical to the rule in 1) to account for the third person plural feminine combination.

Our lexeme-based approach to syncretism corroborates the conclusions in (Zwicky, 1985) and (Stump, 1993), namely that an adequate theory of morphology must incorporate rules of referral in order to account for some kinds of inflectional syncretism. Such a theory also claims that syncretisms do not always encompass whole word forms and that two or more rules of referral may participate in the definition of a single instance of syncretism.

6.3.2 Apophony in Strong Verbs

In this subsection, we briefly address the issue of **apophony** (vowel alternation) in Arabic triliteral Form 1 strong verbs. The perfective stems are typically *façal*, *façil* and *façul*. The first vowel is invariably '*a*', in the active voice; the second vowel (V_2), called stem vowel, is variable. The imperfective stems are *fçal*, *fçil* and *fçul*. In the imperfect, the first vowel drops out (is replaced by lack of vowel '.') and the stem vowel varies. The basic question is whether the perfective-imperfective apophonic alternations are predictable or must be lexically recorded.

Table 6.2 shows that when the stem vowel is '*a*' in the perfective, the imperfective vowel can be '*i*', '*u*', or occasionally '*a*' (the latter occurs mostly in the presence of certain neighboring radicals). When the stem vowel is '*i*' in the perfective, the imperfective vowel is usually '*a*' and occasionally '*i*'. When the stem vowel is '*u*' in the perfective, the imperfective vowel is always '*u*'. Occasionally a verb will have multiple imperfective forms and, rarely, have multiple perfective forms each with its own imperfective form (e.g. *Hasib* "to assume, to compute").

(Guerssel & Lowenstamm, 1996) provides a different account for Arabic apophony, neither of which permits predicting vowel alternations reliably. In our approach we provide information on the imperfective vowel for strong verbs in the lexicon.

Table 6.2. Apophonic alternations attested in Arabic

Stem Vowel	Perfective Stem	Imperfective Stem	Gloss
	*jal**a**s*	*jl**i**s*	to sit down
	*sak**a**n*	*sk**u**n*	to live
a	*kat**a**b*	*kt**u**b*	to write
	*qaT**a**ς*	*qT**a**ς*	to cut
	*fat**a**H*	*ft**a**H*	to open
	*Has**a**b*	*Hs**u**b*	to assume
	*fah**i**m*	*fh**a**m*	to understand
i	*far**i**H*	*fr**a**H*	to be happy
	*sam**i**ς*	*sm**a**ς*	to hear
	*Has**i**b*	*Hs**i**b, Hs**a**b*	to assume
	*sah**u**l*	*sh**u**l*	to become easy
u	*kar**u**m*	*kr**u**m*	to be generous
	*šar**u**f*	*šr**u**f*	to be noble

6.3.3 Generation Model and Implementation

In order to capture the generalizations related to the conjugation of strong verbs and in view of the irregularities exhibited in the paradigms of the different types and classes of weak verbs, the process of Arabic verbal morphology generation should have two steps: first, stem changes are processed, then inflectional prefixes and suffixes. This is indeed the approach we have taken in our work, as illustrated in Section 6.3.3.3 below. Sections 6.3.3.1 and 6.3.3.2 introduce the morphological tool and its enhancements.

6.3.3.1 The Basic MORPHĒ Tool

The MORPHĒ tool (Leavitt, 1994), written in Common Lisp, is based on two major constructs: 1) a Morphological Form Hierarchy (MFH), which relates and distinguishes morphological forms, and 2) transformational rules, attached to leaf nodes of the hierarchy, that operate on the input representation to produce a surface form. To provide a morphological description of a language one must specify a MFH and the corresponding rules. 'Compiling' the morphological description in MORPHĒ produces a Common Lisp program, optionally compiled into object code. The compiled morphological description is used at runtime to generate surface forms.

Runtime Input and Output MORPHĒ's runtime input is a Feature Structure (FS), which describes the lexical item that MORPHĒ must transform. A FS is implemented as a recursive Lisp list. Each element of the FS is a Feature-Value Pair (FVP), where the value can be atomic or complex. A complex value is itself a FS. For example, the FS for generating the Arabic *zurtu* "I visited" is:

```
     ((LEXEME "zawar") (CAT V) (FORM 1) (IMPV HOL)
                                                          (6.1)
   (TENSE PERF) (MOOD IND) (VOICE ACT) (NUMBER SG) (PERSON 1))
```

The feature LEXEME identifies the 'base' form of the lexical item to be transformed and is used as the basis for rule matching. The FVPs in a FS come from one of two sources. Static features, such as CAT (part of speech) and LEXEME, come from the lexicon, which can also contain morphological and syntactic features. Dynamic features, such as tense and number, are set by MORPHĒ's caller. MORPHĒ's output is a string.

The Morphological Form Hierarchy The Morphological Form Hierarchy (MFH) or tree describes the relationship of all morphological forms to each other. The root of the MFH binds all subtrees together. Each internal node of the tree specifies a piece of the FS that is common to that entire subtree. The leaf nodes of the tree correspond to distinct morphological forms in the language. Each node in the tree below the root is built by using a MORPH-FORM declaration that specifies the name of the node, the name of its parent node and the conjunction or disjunction of FVPs that define the node and distinguish it from its parent and siblings. For example, the declaration

```
(MORPH-FORM V-STEM-F1-ACT-PERF-1/2 V-STEM-F1-ACT-PERF    (6.2)
            (PERSON (*OR* 1 2)))
```

says that the node V-STEM-F1-ACT-PERF-1/2 is a child of V-STEM-F1-ACT-PERF and is reached if person feature has value 1 or 2.

Transformational Rules A rule attached to each leaf node of the MFH effects the desired morphological transformations for that node. A rule consists of one or more mutually exclusive clauses. The 'if' part of a clause is a regular expression, which is matched against the string value of the feature LEXEME. The 'then' part includes zero or more operators, applied in the given order. Operators include addition, deletion, and replacement of prefixes, infixes, and suffixes. Applying the relevant clause produces the transformed LEXEME string. For example, the transformational rule that produces the *zur* part of *zurtu* "I visited" is:

Short Stem Rule:

```
(MORPH-RULE V-STEM-F1-ACT-PERF-1/2                                   (6.3)
  ("^%{CONS}(awa)%{CONS}$"       (RI *1* "u"))    ;; CLASS 1
  ("^%{CONS}(a[wy]i)%{CONS}$"    (RI *1* "i"))  ;; CLASSES 2 & 4
  ("^%{CONS}(aya)%{CONS}$"       (RI *1* "i"))    ;; CLASS 3
  )
```

The syntax %{VAR} is used to indicate a variable whose possible values are defined outside the rule, for example:

```
(VARIABLE CONS (... "b" " " "t" " " "j" "H" "x" "d" ...)) (6.4)
```

Enclosing a part of a regular expression in parenthesis associates it with a numbered register, so that an operator can access it for substitution. In the above rule, the first clause says that if the value of the `LEXEME` feature is a consonant, followed by the string "*awa*", followed by another consonant, MORPHĒ should apply the Replace Infix (RI) operator and substitute "*awa*" with "*u*". Hence "*zawar*" becomes "*zur*".

Runtime Process Logic In generation, the MFH acts as a discrimination network. The lexical item to be inflected is pushed down through the hierarchy, by matching features in its FS against the features defining each subtree, until a leaf is reached. At that point, MORPHĒ first checks in the irregular forms lexicon for an entry indexed by the name of the leaf node (as specified by the morph-form declaration) and the value of the `LEXEME` feature in the FS. If an irregular form is found, MORPHĒ outputs it; otherwise it tries to apply a rule attached to the node. If no rule is found or no clause of the applicable rule matches, MORPHĒ returns the value of `LEXEME` unchanged. An input can also fall through the MFH because the combination of FVPs does not lead to a leaf node, in which case the value of `LEXEME` does not undergo any changes.

Limitations of the Basic MORPHĒ Tool A major limitation of the basic MORPHĒ system is that it currently cannot be used to perform morphological analysis. MORPHĒ was developed in the context of a Knowledge-Based Machine Translation system (Nyberg & Mitamura, 1992) whose primary application was translation from English into other languages. MORPHĒ was used for generating target language morphology but not for morphological analysis of English. Hence, while MORPHĒ was planned to perform both analysis and generation, the analyzer was never fully implemented. Whereas the enhancements described in the following section and in Section 6.4.3.1 have addressed several other limitations in the original implementation of MORPHĒ, the addition of an analysis capability will need to wait until after the set of enhancements currently in progress (Section 6.5) is completed.

6.3.3.2 MORPHĒ Enhancements to Support Verbal Morphology

Our initial use of the original MORPHĒ tool to describe Arabic verbal morphology (Cavalli-Sforza et al., 2000; Soudi et al., 2001) and the realization that our approach fits in well with the LBM theory (Aronoff, 1994; Beard, 1995) prompted us to enhance MORPHĒ in a number of ways. Some of the new features in the enhanced MORPHĒ system (**EMORPHĒ**) are designed to explicitly support the LBM theory approach. Others are primarily intended to facilitate the management of a large morphology description. Cumulatively, the enhancements increase modularity and decrease redundancy. All of the enhancements described in this section operate at morphology compilation time and do not affect the MORPHĒ runtime process logic.

Default Rules In the original MORPHĒ system (Leavitt, 1994), a transformational rule could only be attached to leaf nodes. In EMORPHĒ, it is also possible to attach a rule to a pre-leaf node, where it acts as a default rule: if the input FS matches up to the pre-leaf node (the general case) but does not match any of its children (the special cases), the default rule is applied instead. Default rules reduce the number of leaf nodes and avoid the (possibly) complex specification of the complement to special cases. For example, considering hollow verb morphology in the perfect tense, one can attach a default rule that generates a third person long stem (e.g., *zaAr*) to a pre-leaf node and use a more specific rule, attached to a leaf node) for the third person feminine plural to generate a short stem (e.g., *zur*).

Rule Equivalencing The original MORPHĒ system required the MFH to be a tree. If several leaf nodes required the same transformational rule, the rule had to be duplicated. In contrast, EMORPHĒ can avoid rule duplication in one of two ways:

Implicit equivalencing. Different paths in the MFH (i.e., different FVP sequences), can lead to the same node and share information attached to that node. Effectively, the MFH becomes a graph instead of a tree.

Explicit equivalencing. Distinct nodes reached by different paths in the MFH can be explicitly declared to share the same rule by using the declaration syntax:

```
(MORPH-EQUIVALENCE <REFERENCE NODE NAME> <EQUIVALENT NODE LIST>)
```
(6.5)

`<EQUIVALENT NODE LIST>` is a list of one or more names of actual nodes that share a common rule; `<REFERENCE NODE NAME>` is used to attach the rule and can be either the name of an actual node in the hierarchy or a new virtual node. Used in combination with careful design and default rules, rule equivalencing reduces rule duplication, highlights syncretisms, and embodies the rules of referral of LBM theory.

Other Enhancements The original MORPHĒ tool only allowed the use of ROOT as the name of the base form of the lexical item on which transformational rules act, and expected the morphology description to be a single file, which was extremely unwieldy for extensive morphology descriptions. EMORPHĒ allows the user to specify at compilation time, the desired feature name (e.g. `LEXEME`, `STEM`, `ROOT`) and allows a morphology description to be split across multiple files. The only restriction is that a file can make external references only to previously declared information.

6.3.3.3 Arabic Verbal Morphology in MORPHĒ

The linguistic results of Sections 6.3.1 and 6.3.2 suggest that the generation of Arabic verbal morphology be performed in two steps. Figure 6.2 sketches the Arabic MFH and the division of the verb subtree into stem changes and prefix/suffix additions. It also partly fleshes out the perfect tense subtrees for strong and hollow verbs of Form 1 (i.e., pattern *CVCVC*) and shows some of the features used to traverse the MFH.

MORPHĒ is first called with the feature `GEN` (generate) set to stem. After MORPHĒ has traversed the nodes branching from `(GEN STEM)`, the required stem is returned and temporarily substituted for the value of the `LEXEME` feature. The

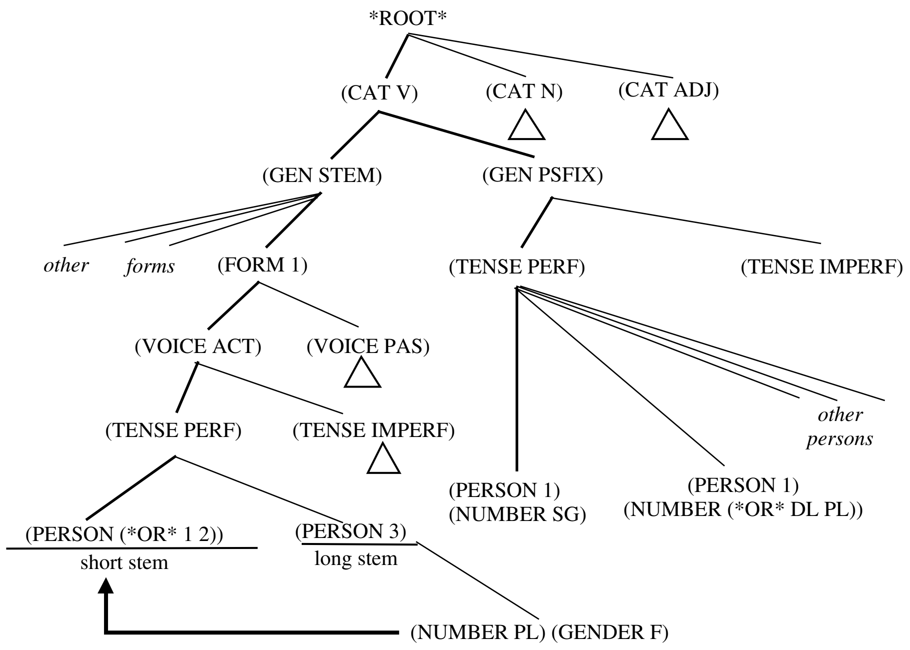


Fig. 6.2. The basic MFH and partially detailed perfective subtrees

second call to MORPHĒ with the feature `GEN` set to `PSFIX` (prefixation and suffixation) applies further rules to the previously computed stem in order to add inflectional prefixes and/or suffixes. The output is a fully inflected verb. (The use of two branches of the tree for the two steps is a constraint of MORPHĒ's current implementation, which does not support multiple trees.)

To demonstrate how the system works, we present the case of hollow verbs (Section 6.3.1) in the perfective and imperfective, relating it to the generation of strong verbs.

Strong and Hollow Verbs in the Perfective As shown in Table 6.1, unlike regular strong verbs, which do not undergo any stem changes in the perfect active voice, hollow verbs use a long stem with a middle *Âalif* (i.e., '*A*') for third person singular and dual (masculine and feminine) and for third person plural masculine (e.g. *daAm* "to last"). The remaining person-number-gender combinations take a short stem whose voweling depends on the underlying root of the verb. These syncretisms are evident in the MFH shown in Figure 6.2: the MFH has a branch for first and second person and a branch for third person. A short stem rule is attached to the first/second person leaf node and a long stem rule is attached to the third person pre-leaf node. The node for third person plural feminine, which requires a short stem, uses a `MORPH-EQUIVALENCE` (represented by the arrow) to refer to the node/rule for first /second person and explicitly embodies a rule of referral. All other cases of syncretism are treated implicitly by using only the shared features that determine the stem to specify the node and its corresponding rule. The MORPHĒ declarations representing these morphological forms and their relationship are given below.

```
(MORPH-FORM V-STEM-F1-ACT-PERF-1/2 V-STEM-F1-ACT-PERF     (6.6)
   (PERSON (*OR* 1 2)))
(MORPH-FORM V-STEM-F1-ACT-PERF-3 V-STEM-F1-ACT-PERF       (6.7)
   (PERSON 3))
(MORPH-FORM V-STEM-F1-ACT-PERF-3-PL-F V-STEM-F1-ACT-PERF-3
                                                           (6.8)
   (NUMBER PLURAL) (GENDER F))
(MORPH-EQUIVALENCE V-STEM-F1-ACT-PERF-1/2                  (6.9)
   (V-STEM-F1-ACT-PERF-3-PL-F))
```

The short stem rule, used by first and second persons and by the third person plural feminine, was given in (6.3); (6.10) shows the default long stem rule for the third person.

Long Stem Rule:

```
(MORPH-RULE V-STEM-F1-ACT-PERF-3                        (6.10)
  ("^%{CONS}A([WY][AI])%{CONS}$" (RI *1* "A")))
```

As mentioned in Section 6.3.1, hollow verb classes 2 and 4 can be merged because they have the same perfect (and imperfect) stems. Inside the short stem change rule, the four different classes of hollow verbs are treated as three separate conditions by matching on the middle radical and the adjacent vowels and replacing them with the appropriate vowel. The long stem is the same for all classes, therefore only one clause is necessary. Strong verbs do not match any clauses in the rules and fall through with no stem changes. Note that there are different ways of expressing the same syncretisms but, in all cases, only 2 stem change rules need to be specified instead of 18 separate and redundant ones.

An example using the verb *nawim* "to sleep" illustrates how the generation process works. Assume the input FS is:

```
((LEXEME "nawim") (CAT V) (FORM 1) (IMPV HOL)    (6.11)
  (VOICE ACT) (TENSE PERF) (PERSON 1) (NUMBER SG))
```

In the first call to MORPHĒ, the `(GEN STEM)` subtree is traversed until the node `V-STEM-F1-ACT-PERF-1/2` is reached. MORPHĒ matches the second clause of rule (6.3) (the short stem rule), and returns the stem *nim* for use in the second call. In the second call, the `(GEN PSFIX)` subtree is traversed until the node labelled `V-PSFIX-PERF-1-SG` (corresponding to the `(PERSON 1)`

`(NUMBER SG)` FVP combination) is reached. MORPHĒ then applies the rule attached to this node, namely:

```
(MORPH-RULE V-PSFIX-PERF-1-SG ("" (+S "tu")))     (6.12)
```

This rule adds the suffix *tu* to the stem *nim* and MORPHĒ returns the fully inflected verb *nim.tu* "I slept". The path through both the stem and psfix subtrees taken in generating this example are shown with thicker lines in Figure 6.2.

Strong and Hollow Verbs in the Imperfective Figure 6.3 shows the imperfect subtree for strong and hollow verbs of Form 1 (pattern *CVCVC*). Strong verbs are treated by three rules branching on the middle radical vowel, given as the value of `IMPV`. The consonant-vowel pattern of the computed stem is shown. For example, for *katab* "to write", the lexicon contains the FVP `(IMPV u)` and the imperfect stem would be *ktub* in the pattern *CCuC*. The imperfect vowel is stored in the lexicon because it is not always determined by the perfect vowel, as is explained in Section 6.3.2 (though, in the presence of certain second and third radicals, the stem vowel is more precisely determined).

For hollow verbs, the imperfective vowel depends on the middle weak radical and the vowel immediately following it in the underlying stem. As for the perfective, it is computed by transformational rules, and is not stored in the lexicon. Hence the feature `IMPV` is only used to distinguish hollow and other kinds of verbs from strong verbs. To show the syncretisms present in this inflectional paradigm, while avoiding visual clutter, in Figure 6.3 we have used the labels "short stem" and "long stem" (e.g., for *nawim* "to sleep" the stems would be *nam* and *naAm* respectively) instead

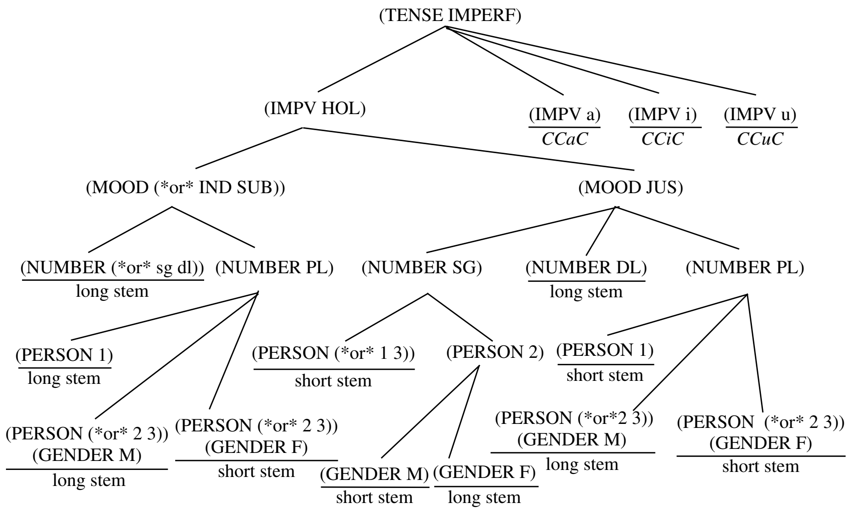


Fig. 6.3. Imperfect stem change subtree for strong and hollow verbs of form 1

of using arrows to represent the `MORPH-EQUIVALENCE` declarations actually used in the morphology description.

The hollow verb subtree shown in Figure 6.3 is not as small for the imperfect as it is for the perfect, since the stem depends not only on the mood but also on the person, gender, and number. It is still advantageous to decouple stem changes from prefixation and suffixation since only two stem change rules are needed for a given voice and prefix and suffix rules are largely shared with other verb forms/patterns and types of verbs.

6.4 The Arabic Noun System

There are three number categories for Arabic nouns (including adjectives): singular (*mufrad*), dual (*muθan∼aý*), and plural (*jamς*). The plural is further divided into sound (*Aljamςu Als∼aAlimu*), the use of which is practically confined (at least in the masculine) to participles and nouns indicating profession and habit, and broken (*Aljamςu Almukas∼aru*) types. Broken plurals are then divided into "plurals of paucity" (*jamςu Alqil∼aħi*), denoting three to ten items, and "plurals of multiplicity" (*jamςu Alkaθraħi*), denoting more than ten items. There are four forms of the plural of paucity and at least 23 forms of the plural of multiplicity (Abu Al-Suud, 1971). Several singulars have more than one plural form. There are also underived nouns with plural or collective sense (usually indicating a group of animals or plants). These are treated as singular but may form a 'singulative' (*Ǎismu AlwaH.daħi*), indicating an individual of the group, by attachment of the suffix '*ħ*' (*tA' marbuwTaħ*).

In this section, we provide both linguistic and statistical evidence against generating broken plurals from the singular or the root. We propose instead a multiple-stem approach to nouns with a broken plural pattern that dispenses with the complex rules required to account for the highly allomorphic broken plural system. For sound nouns we specify the suffix type in the lexicon (Soudi et al., 2002).

6.4.1 The Arabic Broken Plural System

The Arabic broken plural system is highly allomorphic: for a given singular pattern, two different plural forms may be equally frequent, and there may be no way to predict which of the two a particular singular will take. For some singulars as many as three further statistically minor patterns are also possible. The range of allomorphy is, in general, from two to five. For example, a singular noun with the pattern *CVCC* would have one or two of the following plural patterns: *CuCuwC*, *ÂaCCaAC*, *CiCaAC* or *ÂaCCuC*. Examples showing the broken plural of the singular pattern CVCC are as follows:

singular	plural	gloss
wazn	*ÂawzaAn*	measure
kalb	*kilaAb*	dog
ςayn	*ςuyuwn, Âaςyun*	eye

(6.13)

To evaluate the statistical productivity in the Arabic plural system, we, following (Ratcliffe, 1992, 1998), used Levy's (1971) study, reproduced in Table 6.4, which includes statistical information on common plural types, based on Wehr's (1980) dictionary. Singulars based on quadriliteral roots are not included. An older study by Murtonen (1964), based on the dictionary of Lane (1893), gives somewhat different results and makes different assumptions, but both demonstrate that while the association between singular and plural forms is not random, there is no way to predict exactly which plural pattern a singular will take.

The left-most column in the tables indicates the singular patterns and the top row indicates the most frequent plural forms. There are two numbers at every co-ordinate where the singular line crosses the plural one. The first indicates the percentage of the particular singular in relation to all singulars of a given plural. The second indicates the percentage of the particular plural as a proportion of all the plurals taken by a given singular type. By way of example, the intersection of the singular pattern *CaCC* and the broken plural pattern *CuCuwC* in Table 6.4 shows the numbers 73/49. These numbers indicate that 73 percent of all the singulars of a plural pattern *CuCuwC* have a singular of the pattern *CaCC* and that 49 percent of all singulars with the pattern *CaCC* will have *CuCuwC* as their plural pattern.

The allomorphy exhibited by the Arabic broken plural system can be handled by providing the broken plural pattern in the lexicon and a series of rules that operate on the singular noun to generate the plural noun in the morphological component. These rules would act at the internal level to convert the singular stem to the plural stem and at the external level to add the inflectional affixes (e.g., Case affixes: nominative, genitive or accusative suffixes). Alternatively, one could provide the singular and plural stems in the lexicon and then have inflectional morphology act on these stems (Soudi et al., 2002).

The first approach would obviously involve several rules, since nouns with a broken plural pattern have in general complex stem alternants. The multiple-stem approach is more promising. Nouns with a broken pattern commonly display two major stem alternants: the singular/dual allostem and the plural allostem. To capture the fact that there are two forms and that these forms are systematically distributed, the lexeme is given an inventory of two stems, labeled by Number. For generation, only one plural stem suffices, since there is no good criterion for selecting among multiple stems.

6.4.2 Nouns and Inflection

In this section, we look at the inflection of the Arabic noun system and consider some syncretism cases in the noun inflection of Arabic. Tables 6.5 and 6.6 show that the accusative and genitive Cases are realized homonymously in the sound plural but not in the broken plural. (The definite article *Al* is not included in the table.)

In the relevant literature, the main morphological distinction in declension is that between the broken plurals and the rest. The examples in Table 6.5, however, show

Table 6.4. Singular/plural distribution based on (Levy, 1971)

Plural Singular	*ÂaCCuC*	*CuCuwC*	*ÂaCCaAC*	*CiCaAC*	*CiCaC*	*CuCaC*	*-aAt* (sfp)	*CawaACiC*
CaCC	82 /6	73/49	26/27	29/12				
CiCC	12/3	13/23	23/67	5/7				
CuCC	6/2	6/16	17/73	5/9				
CvCvC		6/9	33/85	5/6				
CaCCa				18/18	14/5	6/3	43/74	
CiCCa				1/4	86/84		2/12	
CuCCa				6/8	6/15	94/77	6/8	
CaCvCa				5/18			13/82	
CaACiCa		3/4	½	3/2			5/16	58/84
CaACiC							2/3	42/24 →
CuCuwCa							1/86	→
CaCaACa							7/74	→
CuCaACa							4/87	→
CiCaACa							4/44	→
CaCuwCa							0/14	→
CaCiyCa								→
CuCuwC							1/100	→
CaCaAC							4/27	→
CuCaAC							1/16	→
CiCaAC							6/20	→
CaCuwC				21/16				→
CaCiyC								→

(continued below)

Plural Singular	*CuCCaAC*	*CuCCaC*	*CaCaCah*	*CuCaCa*	*CaCaACiC*	*aCCiCa*	*CuCuC*
CaACiC	← 100/26	100/10	98/14	100/11			5/2
CuCuwCa	←				0/14		
CaCaACa	←				4/26		
CuCaACa	←				1/13		
CiCaACa	←				11/56		
CaCuwCa	←				2/86		
CaCiyCa					70/95		7/5
CuCuwC	←						
CaCaAC	←				1/5	22/52	6/13
CuCaAC	←					8/48	
CiCaAC	←				1/1	48/45	38/34
CaCuwC	←				3/20	3/11	16/63
CaCiyC					6/4	18/17	29/11

Table 6.5. Paradigm of word forms of the sound nouns *muςal~im* "instructor" and *HayawaAn* "animal"

Definiteness	Case	Sound Plural Masculine		Sound Plural Feminine	
		Singular	Plural	Singular	Plural
Indefinite	**Nominative**	*muςal~imũ*	*muςal~imuwna*	*HayawaAnũ*	*HayawanaAtũ*
	Accusative	*muςal~imã*	*muςal~imiyna*	*HayawaAnã*	*HayawanaAtĩ*
	Genitive	*muςal~imĩ*	*muςal~imiyna*	*HayawaAnĩ*	*HayawanaAtĩ*
Definite	**Nominative**	*muςal~imu*	*muςal~imuwna*	*HayawaAnu*	*HayawanaAtu*
	Accusative	*muςal~ima*	*muςal~imiyna*	*HayawaAna*	*HayawanaAti*
	Genitive	*muςal~imi*	*muςal~imiyna*	*HayawaAni*	*HayawanaAti*

that there is another fundamental distinction between two types of sound plurals, related to the spell-out of the Number marker and the Case marker. The exponent of Plural in sound plural masculines is *uwn* (as in *muςal~im**uwn**a*) while in the sound plural feminine it is *aAt* (as in *Hayawan**aAt**u*), using the nominative Case marker as the default Case for sound nouns. Case is realized in the two types of sound plural in different positions with respect to the Plural marker (***u**wna, **i**yna* vs. *aAt**u***, *aAt**i***).

6.4.3 Arabic Noun Morphology in MORPHĒ

The Arabic plural noun system imposes different demands on the morphological representation than the verb system. As discussed above, a minority of nouns form their plural by regular processes of suffixation, but the majority of nouns have one or more broken plural forms, whose pattern is not predictable from the singular pattern. Therefore, drawing from Lexeme-Based Morphology (Aronoff, 1994; Beard, 1995), we choose to give priority to stems and store the broken plural stem in the lexicon.

6.4.3.1 MORPHĒ Enhancements to Support Noun Morphology

With the broken plural stem of nouns in the lexicon, we can employ the same two-stage approach we used for verbal morphology to generate inflected nouns, after introducing a further enhancement to MORPHĒ: **allomorph substitutions**. As

Table 6.6. Paradigm of word forms of the broken noun *rajul* "man"

Definiteness	Case	Singular	Plural
Indefinite	**Nominative**	*rajulũ*	*rijaAlũ*
	Accusative	*rajulã*	*rijaAlF*
	Genitive	*rajulĩ*	*rijaAlĩ*
Definite	**Nominative**	*rajulu*	*rijaA1u*
	Accusative	*rajula*	*rijaAla*
	Genitive	*rajuli*	*rijaAli*

mentioned in Section 6.3.3, transformational rules act upon the value of a feature that, in EMORPHĒ, is specified at morphology compilation time (e.g. `LEXEME`). An allomorph substitution attached to a leaf node indicates that a different feature in the FS should be the source of the string to be transformed. The syntax of an allomorph substitution declaration is:

```
(MORPH-ALLOMORPH <NODE NAME> <FEATURE NAME>)
```
(6.14)

The addition of allomorph substitutions does modify somewhat the runtime process logic of the original MORPHĒ system, as described in Section 6.3.3.1, by introducing the allomorph substitution step. The resulting logic is shown in Figure 6.4.

6.4.3.2 The Noun Morphological Form Hierarchy

Figure 6.5 shows a portion of the computational implementation of noun generation in EMORPHĒ. The MFH is fully fleshed out only for nominative indefinite noun inflection, but the accusative and genitive cases and definite nouns are handled in a parallel fashion. Definite nouns, with FVP `(DEF +)`, have a different suffix and the prefix *Al* which, depending on the initial consonant of the noun, may either have a *sukuwn* or undergo assimilation and cause gemination ('∼') of the initial consonant. Figure 6.5 shows node names for the subset of nodes referenced in the following discussion but only the distinguishing FVPs for the remainder of the nodes.

As for verbs, the noun subtree of the MFH is split into two subtrees, one for producing the correct stem and one for adding the necessary suffixes. Correspondingly, generation of a fully inflected noun requires two calls to EMORPHĒ: the first, to generate the required stem, temporarily adds the FVP `(GEN STEM)` to the FS; the second, to generate the appropriate inflectional prefixes and suffixes, adds `(GEN PSFIX)`.

For singular and dual nouns, there is no stem change, therefore, no branches for `(NUMBER SG)` or `(NUMBER DL)` appear in the MFH under the `(GEN STEM)`

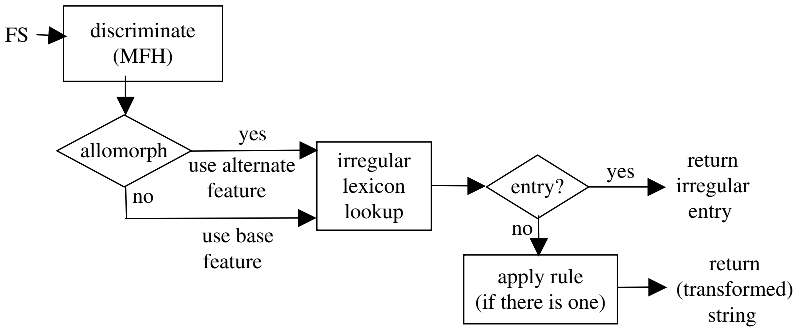


Fig. 6.4. EMORPHĒ's runtime process logic

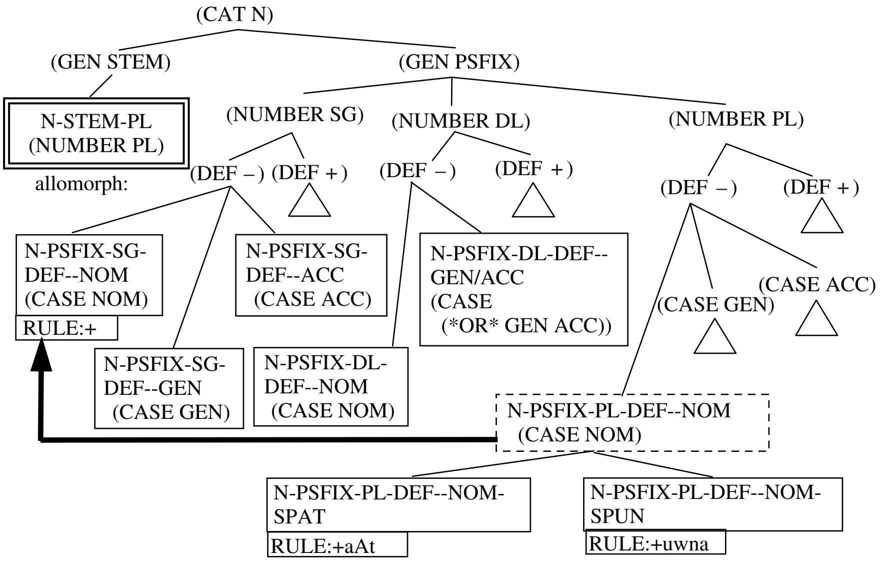


Fig. 6.5. The noun Morphological Form Hierarchy (MFH)

node. The FS just falls through the tree `(GEN STEM)` subtree in the first call, and processing through the `(GEN PSFIX)` subtree proceeds normally during the second call. For plural nouns, an allomorph rule is attached to the node `N-STEM-PL` using the declaration:

`(MORPH-ALLOMORPH N-STEM-PL BPSTEM)` (6.15)

It specifies that, if node `N-STEM-PL` is reached, EMORPHĒ should look in the FS for a feature named `BPSTEM`, which, if present, should be used in subsequent processing instead of the value of the `LEXEME` feature. The second call to EMORPHĒ, traverses the `(GEN PSFIX)` subtree, using the rule of referral attached to `N-PSFIX-PL-DEF--NOM` (represented by the thick arrow) to add inflectional suffixes to broken plurals and regular transformational rules attached to `N-PSFIX-PL-DEF--NOM-UN` and `N-PSFIX-PL-DEF--NOM-AT` for sound plurals. Specific examples of processing are presented below.

Noun with a Sound Plural

Input FS: `((LEXEME "mudar~is")(SP UN) (NUMBER PL)(CASE NOM)(DEF -))`

In the first call to EMORPHĒ, the `(GEN STEM)` subtree is traversed using the feature `(NUMBER PL)` to reach the leaf node `N-STEM-PL`. Since no `BPSTEM` feature is found in the FS, the allomorph substitution is ignored and `"mudar~is"` is returned for use in the second call. Then the `(GEN PSFIX)` subtree is traversed until the node labeled `N-PSFIX-PL-DEF-NOM-SPUN` is reached. The transformational rule in (6.16) produces the result *Almudar~isuwna* "the teachers".

```
(MORPH-RULE N-PSFIX-PL-DEF+-NOM-SPUN          (6.16)
  (""
    (+P "AL")
    (+S "UWNA")
  ))
```

The condition part of the rule is the empty string “”, which matches any input. The operator `+P` prefixes the definite article *Al* and the operator `+S` suffixes *uwna*, a portmanteau morpheme for the Number and Case exponents.

Noun with a Broken Plural

Input FS: `((LEXEME "rajul")(BPSTEM "rijaAl")`
`(NUMBER PL)(CASE NOM)(DEF -))`

In the first call to EMORPHĒ, the `(GEN STEM)` subtree is traversed, reaching the leaf node `N-STEM-PL`. The value of the feature `BPSTEM` – *rijaAl* “men”– is retrieved from the FS via the allomorph substitution attached to that node and is returned for use in the second call. Traversing the `(GEN PSFIX)` subtree on the second call, the node `N-PSFIX-PL-DEF-NOM` is reached. No `SP` feature is found in the feature structure, so MORPHĒ defaults to the rule attached to `N-PSFIX-PL-DEF--NOM`, which refers to the rule attached to node `N-PSFIX-SG-DEF-NOM`:

```
(MORPH-RULE N-PSFIX-SG-DEF-NOM          (6.17)
           (""
           (+S "N")
           ))
```

In Figure 6.5, the thick arrow represents the explicit equivalence:

```
(MORPH-EQUIVALENCE N-PSFIX-SG-DEF--NOM (N-PSFIX-PL-DEF--NOM))
                                                        (6.18)
```

It says that the suffix for an indefinite nominative plural noun is the same as that for an indefinite nominative singular noun. The equivalence works together with default rules, allowing the default rule to be equivalenced to a rule on a different node.

6.5 Work in Progress

While EMORPHĒ significantly enhances the expressiveness and convenience of original MORPHĒ, it still falls short of providing an optimally concise and elegant framework for generating Arabic morphology. It can be further enhanced to both improve runtime efficiency and to more elegantly capture regularities in morphological descriptions. In the two-stage approach used to generate the fully inflected form of a lexical item, redundant work is performed in checking the features in the input FS twice, once for determining the appropriate stem, then again for determining the prefix and suffix. Since the stem subtree is relatively shallow, at least for nouns and for sound and hollow verbs, the cost in time is not high, but avoiding it altogether would be better. In addition, the two-stage process complicates extending EMORPHĒ to perform analysis.

Work in progress addresses this issue by introducing inheritance of rules and other information associated with internal nodes of the MFH, thereby allowing the MFH to be traversed only once while collecting and adjusting the information required to generate the inflected form. The final version of EMORPHĒ will allow internal nodes of the hierarchy to have allomorph substitutions attached to them, and will allow explicit equivalences to use internal nodes of the MFH as reference nodes (implicit equivalencing of subtrees is already supported). In the remainder of this section we sketch, mostly via figures, how the enhancements in progress will affect the representation of Arabic morphology in EMORPHĒ.

Figure 6.6 shows a portion of the MFH for Arabic verb morphology when stem changes and prefix/suffix additions are merged into a single hierarchy using inheritance. For lack of space, the feature names `GENDER`, `NUMBER`, and `PERSON` have been abbreviated to `GEN`, `NUM`, and `PER` respectively.

As an example of how processing works, consider again the feature structure for obtaining the Arabic *zurtu* "I visited" given by (6.1) in Section 6.3.3.1. The short stem rule in (6.3), which yields *zur*, is attached to the internal node labeled with `(PERSON 1)`. Other parts of the MFH where this rule is required make reference to it through explicit equivalencing. As the FS is pushed down through the MFH

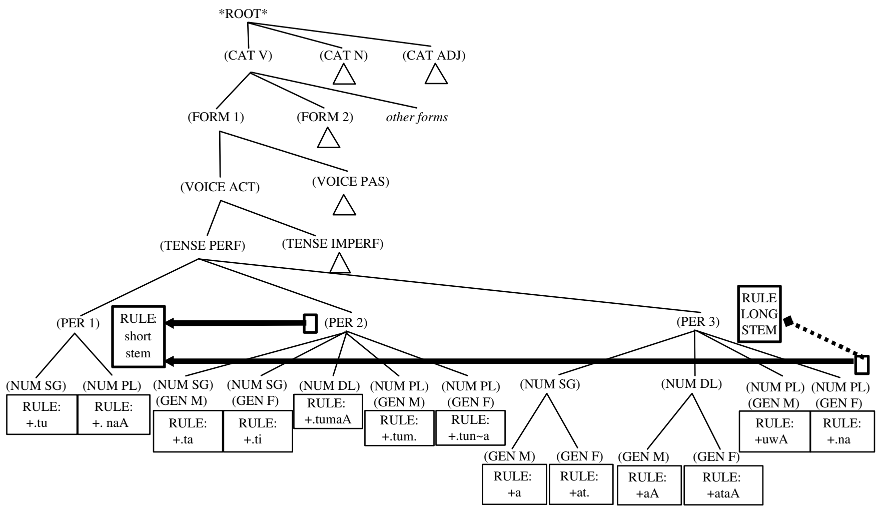


Fig. 6.6. Partially detailed perfective verb MFH with inheritance

and reaches the nodes labeled with `(PERSON 1)`, the stem rule is picked up first. Then, as the FS is discriminated down to the more specific leaf node labeled with `(NUMBER SG)`, the rule that adds the suffix *tu* is also picked up. Both rules are applied to produce the final form *zurtu*. FSs containing the FVP `(PERSON 2)` are processed similarly. For FSs containing the FVP `(PERSON 3)`, the MFH shows a long stem rule attached to the node labeled `(PERSON 3)`, but if the FS also contains `(NUMBER PL) (GENDER F)` the short stem rule referenced via an explicit equivalence overrides the long stem rule.

Noun morphology can also be represented more succinctly with inheritance. Figure 6.7 shows an MFH similar to that shown in Figure 6.5 (Section 6.4.3.2) but with allomorph substitution for broken plurals attached high in the tree and overridden at leaf nodes for sound plurals, and prefixation/suffixation information at the leaf and pre-leaf nodes. For lack of space, the feature `CASE` and its values `NOM`, `GEN`, `ACC`, have been abbreviated to `CS`, `N`, `G`, and `A`, respectively. We note that it is possible to rearrange the noun MFH to exploit inheritance more fully, and we are currently examining this option.

Figures 6.6 and 6.7 show how ongoing enhancements to EMORPHĒ can linearize a non-linear morphology. Stem changes common to two or more word forms are effected by transformational rules attached higher in the MFH and overridden, if necessary, at lower levels. Form-specific prefixation/suffixation rules are attached to nodes lower in the hierarchy, usually leaf or pre-leaf nodes (in the case of default rules). What the figures do not show are the design and implementation implications

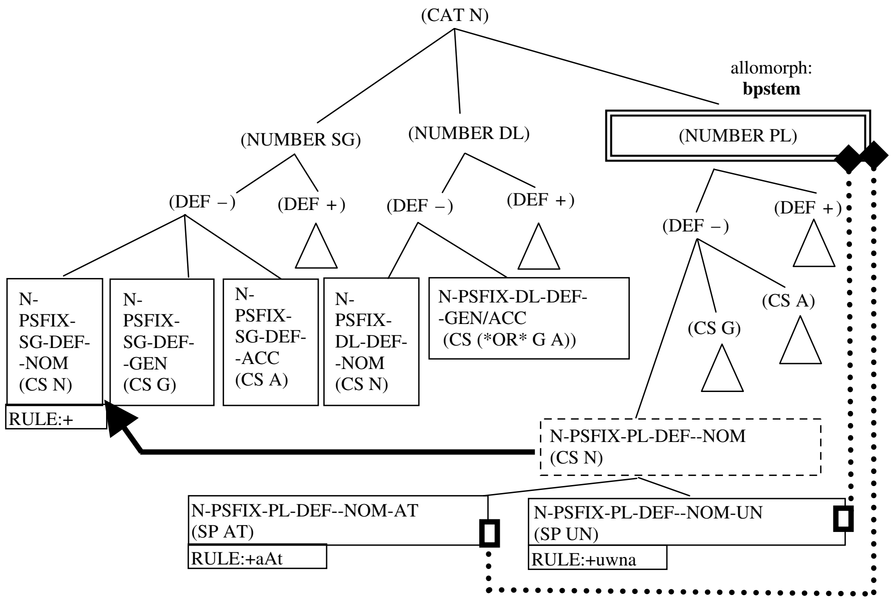

(CAT N)
allomorph:
bpstem
(NUMBER SG)
(NUMBER DL)
(NUMBER PL)
(DEF –)
(DEF +)
(DEF –)
(DEF +)
(DEF –)
(DEF +)
N-PSFIX-SG-DEF--NOM (CS N)
RULE:+
N-PSFIX-SG-DEF--GEN (CS G)
N-PSFIX-SG-DEF--ACC (CS A)
N-PSFIX-DL-DEF--NOM (CS N)
N-PSFIX-DL-DEF--GEN/ACC (CS (*OR* G A))
(CS G)
(CS A)
N-PSFIX-PL-DEF--NOM (CS N)
N-PSFIX-PL-DEF--NOM-AT (SP AT)
RULE:+aAt
N-PSFIX-PL-DEF--NOM-UN (SP UN)
RULE:+uwna

Fig. 6.7. Partially detailed noun MFH with inheritance

of adding inheritance to EMORPHĒ, which recall issues associated with object-oriented programming languages. We also leave discussion of these for future work.

6.6 Conclusions and Future Work

This chapter has presented our approach to generating Arabic morphology from a combination of empirical, theoretical and implementation perspectives. We have reviewed the linguistic framework – Lexeme-Based Morphology (LBM) – that has accompanied the development of the approach, provided and analyzed morphological data in its support, and described its implementation within the Enhanced MORPHĒ (EMORPHĒ) system. EMORPHĒ presents a number of significant extensions with respect to the original MORPHĒ system, targeted at making the system both more suitable for expressing linguistic phenomena found in Arabic and more convenient to use. The result is a morphological description tool that elegantly and concisely captures the transformations undergone by a lexeme during inflection and highlights the characteristics and regularities of Arabic morphology while distinguishing among different behaviors. In concluding, we examine our implementation of Arabic morphology generation in EMORPHĒ and the tool itself, from both linguistic and computational viewpoints.

From a linguistic viewpoint, our approach reflects an empirically and theoretically motivated decision to share the information required to generate fully inflected forms between the lexicon – in the form of multiple stems – and the morphology description – in the form of allomorph substitutions and transformational rules operating on those stems. Allomorph substitutions support alternate stems for Arabic broken plurals. Acting in combination with the broken plural stems in the lexicon, they indicate that a change in the stem is expected, but cannot be reliably predicted, and therefore one must look to lexeme-specific information in the lexicon. In contrast, transformational rules express regular inflectional operations shared by a class of verbs and inflected forms and are lexeme-independent. The specification of transformational rules applicable to particular (classes of) inflected forms is further aided by default rules and explicit equivalencing. Default rules support the representation of morphological operations that are shared by several inflected forms, allowing operations associated with particular combinations of features to be specified as exceptions, while explicit equivalencing vividly displays the syncretisms present in inflectional paradigms.

From a computational viewpoint, EMORPHĒ supports the development of morphological descriptions that are significantly more modular and concise than those possible with the original MORPHĒ tool, and therefore easier to work with and maintain. Multiple morphology files allow a complex morphology description to be broken down into coherent modules with clear relationships to the remaining modules and the sole restriction that references to constructs outside the module require the targets of those references to have been previously declared. Explicit equivalencing allows transformational rules to be attached to only one node in the morphological form hierarchy and be referred to elsewhere, thereby reducing the need for rule duplication in several places in the hierarchy. Syntax changes to transformational rules, related to the introduction of inheritance, will completely eliminate rule redundancy. Inheritance of information from internal morphological hierarchy nodes, in combination with the existing implicit equivalencing mechanism, will permit leaf nodes representing fully inflected forms to inherit information associated with entire subtrees of the morphological hierarchy and allow EMORPHĒ to encapsulate in a single processing pass the two-stage linearization that underlies our approach to generating Arabic morphology. While inheritance will necessarily render compilation of a morphology description more complex and time-consuming, it will result in faster runtime performance and will simplify the extension of EMORPHĒ to perform morphological analysis. Work on inheritance-based EMORPHĒ is underway (Cavalli-Sforza & Soudi, 2006).

At present, our coverage of Arabic morphology is limited to nouns (excluding nouns that do not nunnate) and some verb forms (sound and hollow verbs). Current efforts on extending coverage are driving many of the in-progress enhancements of the system. We expect the fully-fledged EMORPHĒ tool with inheritance will treat not only isolated verb and noun forms but also be able to represent proclitics (e.g., the prepositions bi, li, and ka; the conjunctions wa and fa; and the future particle sa) and enclitics (e.g. the suffixed possessive and direct object pronouns). We also note that, while to date we have focused exclusively on inflectional morphology of Arabic, the same framework can be used to describe derivational morphology as well.

Another future work with respect to EMORPHĒ, will be its extension to perform morphological analysis as well as generation. The currently working version of the tool does not immediately lend itself to reversibility. In generation, a specific FS uniquely describes the path through (E)MORPHĒ's MFH but, in analysis, the path that leads from the input surface form (a string) to the FS output is not necessarily unique. The system must search through several potential paths, discard most of them, and eventually arrive at one or more analyses that are consistent with the input surface form. Hence morphological analysis requires embedding a search strategy into the process. Modifications to support analysis await the completion of inheritance-related work.

References

Abu Al-Suud, A. (1971). *Al-Faisal fi Alwaan Al-Jumuu* [The Distinction among the Colors of the Plurals]. Cairo: Daar Al-maarif.

Aronoff, M. (1994). *Morphology by Itself: Stems and Inflectional Classes*. Cambridge, MA: MIT Press.

Badawi, E., Carter, M.G. & Gully, A. (2004). *Modern Written Arabic: A Comprehensive Grammar*. New York: Routledge.

Beard, R. (1995). *Lexeme-Morpheme Base Morphology: A General Theory of Inflection and Word Formation*. Albany: State University of New York Press.

Beesley, K. (1996). Arabic Finite-State Morphological Analysis and Generation. In *Proceedings of COLING-96* (Vol. *1*, pp. 89–94).

Beesley, K. (1998). Consonant Spreading in Arabic Stems. In *Proceedings of COLING-98* (pp. 117–123).

Cavalli-Sforza, V. & Soudi A. (2003). Enhancements to a Morphological Generator to Capture Arabic Morphology. In *Proceedings of the Eighth International Symposium on Social Communication* (pp. 565–570), Center of Applied Linguistics, Santiago de Cuba.

Cavalli-Sforza, V. & Soudi A. (2006). IMORPHĒ: An Inheritance and Equivalence Based Morphology Description Compiler. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation* (LREC-06, pp.13–18), Genova, Italy.

Cavalli-Sforza, V., Soudi, A. & Mitamura, T. (2000). Arabic Morphology Generation Using a Concatenative Strategy. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (NAACL-00, pp. 86–93), Seattle.

Guerssel, M. & Lowenstamm, J. (1996). Ablaut in Classical Arabic Measure 1 Active Verbal Forms. In Lecarme, J., Lowenstamm, J. & Shlonsky, U. (Eds.), *Studies in Afroasiatic Grammar* (pp. 123–134). The Hague: Holland Academic Graphics.

Kay, M. (1987). Non-concatenative Finite-state Morphology. In *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics* (pp. 2–10), Copenhagen, Denmark.

Kiraz, G. (1994). Multi-tape Two-level Morphology: A Case study in Semitic Non-Linear Morphology. In *Proceedings of COLING-94* (Vol. *1*, pp. 180–186).

Kiraz, G. (1998). Arabic Computational Morphology in the West. In *Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing*, Cambridge.

Kiraz, G. (2000). A Multi-tiered Nonlinear Morphology using Multi-tape Finite State Automata: A Case Study on Syriac and Arabic. In *Computational Linguistics*, *26*(1), 77–105.

Koskenniemi, K. (1983). *Two-level morphology: A General Computational Model for Word-Form Recognition and Production*. Ph.D. dissertation, University of Helsinki.

Lane, E.W. (1863–93). *An Arabic-English Lexicon* (8 volumes). London: Williams and Norgate.

Leavitt, J.R. (1994). *MORPHĒ: A Morphological Rule Compiler*. Technical Report, CMU-CMT-94-MEMO.

Levy, M.M. (1971). *The Plural of the Noun in Modern Standard Arabic*. Doctoral Dissertation, University of Michigan.

McCarthy, J. (1979). On Stress and Syllabification. *Linguistic Inquiry, 10,* 443–465.

McCarthy, J.A. (1981). Prosodic Theory of Non-Concatenative Morphology. *Linguistic Inquiry, 12*, 373–418.

Murtonen, A. (1964). *Broken Plurals, the Origin and Development of the System*. Leiden: E.J. Brill.

Nyberg, E. & Mitamura, T. (1992). The KANT system: Fast, accurate, high-quality translation in practical domains. *Proceedings of COLING-92* (pp. 1254–1258).

Ratcliffe, R.R. (1992). *The Broken Plural Problem in Arabic, Semitic and Afroasiatic: A Solution Based on the Diachronic Application of Prosodic Analysis.* Ph.D. Dissertation, Yale University.

Ratcliffe, R.R. (1998). *The 'Broken' Plural Problem in Arabic and Comparative Semitic: Allomorphy and Analogy in Non-concatenative Morphology*. Amsterdam/Philadelphia: John Benjamins.

Soudi, A., Cavalli-Sforza, V. & Jamari, A. (2001). A Computational Lexeme-based Treatment of Arabic Morphology. In *Proceedings of the ACL-01 Workshop on Arabic Language Processing: Status and Prospects* (pp. 155–162), Toulouse, France.

Soudi, A., Cavalli-Sforza, V. & Jamari, A. (2002). The Arabic Noun System Generation. In *Proceedings of the International Symposium on The Processing of Arabic* (pp. 69–87), University of Manouba, Tunisia.

Stump, G.T. (1993). On Rules of Referral. *Language, 69*(3), 449–479.

Wehr, H. (1980). *A Dictionary of Modern Written Arabic* (Milton Cowan, J., Ed., 4th ed.). Ithaca, NY: Spoken Language Services.

Zwicky, A. (1985). How to Describe Inflection. *Berkeley Linguistic Society*, 372–386.

7

Grammar-Lexis Relations in the Computational Morphology of Arabic

Joseph Dichy[1] and Ali Farghaly[2]

[1] *Université Lumière-Lyon 2, ICAR research lab (CNRS/Lyon 2), 86, rue Pasteur, 69365 Lyon Cedex 07 – France*
Joseph.Dichy@univ-lyon2.fr

[2] *Oracle USA, 400 Oracle Parkway, Redwood Shores, California 94065 – USA*
Ali.Farghaly@oracle.com

Abstract: Grammar-lexis rules and relations ensuring correct insertion of major lexical entries (nouns, verbs and deverbals) play an essential part in the computational morphology of Arabic. This chapter, which is based on the experiences of the DIINAR.1 Arabic lexical resource and related software, and on that of the first version of the SYSTRAN Arabic-English MT system, outlines previous approaches of the computational morphology of the language (Section 2): root and pattern (shortly recalled); lexeme-based; machine learning and statistical; stems, based on roots and patterns, and finally, the stem-based approach, including root and pattern as well as grammar-lexis information. The latter, which is the most compliant to the requirements of machine-translation and other high-level applications, is further developed in Section 3. Authors go on presenting the structure of the Arabic word-form and a mapping of rules and relations accounting for grammar-lexis relations operating within the boundaries of that complex unit. In the Word-Formatives Grammar, rules and relations involving the lexical nucleus of the word-form play a crucial part and are formalised in a computational perspective. The stem either coincides with, or is the core of the nucleus, because lexical entries include two overall categories: in the first, stem and entry coincide; in the second, the lexical entry corresponds to a morphological compound encompassing the stem and a lexicalized extension (in most cases, a suffix which is part of the entry). Correct relations between the lexical nucleus and the other formatives included in the word-form are ensured through morphosyntactic specifiers associated to each entry of the lexical database. These relations, which have been included in the DIINAR.1 database, are both finite in number and exhaustive in coverage. They also allow computational morphology and other applications to rely on a good restriction of the generated lexica: only cliticized or affixed formatives that can effectively be associated with a given lexical nucleus are added, and 'illegal' ones are ruled out. In the DIINAR.1 resource, the effective number of inflected word-forms is 7,774,938 (about nine times less than one would obtain through 'blind' generation). A comprehensive mapping of examples is given. Their compatibility with applications going beyond computational morphology is also outlined

7.1 Introduction

The present chapter is fundamentally concerned with the role, which will be shown to be crucial, of grammar-lexis relations in the computational morphology of the written form of Modern Standard Arabic (henceforth 'Arabic'). Computational morphology is the component of the linguistic engineering of the language that deals with the analysis and/or generation of the grammatical and lexical morphemes encompassed in the boundaries of the word-form, the structure of which proves, in Arabic, to be that of a complex unit. The contents of this contribution are based on two experiences in Arabic NLP development, that of the DIINAR.1 Arabic lexical resource and related analyzers and software, and that of the lexical database and analyzers embedded in the SYSTRAN Arabic-English machine translation system.

DIINAR.1 (*DIctionnaire INformatisé de l'ARabe, version 1*), Arabic acronym ***MaçaAliy*** معالي (*Muçjam Al-çarabiy~aħ Al-Āliy~, /Muçjam al-çarabiyya(t) al-'āliyy/* – معجم العربية الآلي)[1], is a comprehensive Arabic lexical resource of around 120.000 lemma-entries operating at word-form level. It has been completed in close cooperation by IRSIT in Tunis (A. Braham and S. Ghazali), and in France, by the Lumière-Lyon 2 University (J. Dichy) and ENSSIB (M. Hassoun). Main related software are the word-form (or morphological) analyzer developed by M. Ghenima (1998), which was followed by R. Ouersighni's AraParse syntactic analyzer (2001, 2002), R. Zaafrani's Al-Muçal~im (المعلّم) Computer-aided learning system (2002) and R. Abbès's AraMorph morphological analyzer and AraConc concordance software (2004) – all of which have been devised to support the analysis of unvowelled Arabic script, and the generation, when needed, of fully vowelled written words-forms.[2]

The **SYSTRAN's** Arabic-English MT system is a fully automatic transfer system. A first version has been developed at SYSTRAN's offices in San Diego and Paris between 2002 and 2004 by a team of computational linguists and lexicographers including Jean Sénellart, Ali Farghaly, Dina Abu Qaoud, Mats Attnas and Sylvie Poirier.

Both experiences show that grammar-lexis relations are associated to actual lexical entries, and can, subsequently, only be implemented in a stem-based lexical resource (including root and pattern information), as opposed to a resource founded on pure root-and-pattern combination (Dichy & Farghaly, 2003).

[1] Whenever needed, simplified and more traditionally phonological transcription has been added between slashes (//) to the very comprehensive and in many cases original transliteration system reflecting Arabic script introduced in the present volume.

[2] See Dichy, Braham, Ghazali & Hassoun (2002); Abbès, Dichy & Hassoun (2004). Availability: through ELDA, European Evaluation and Language Resources Distribution Agency, 55, rue Brillat-Savarin, 75013 Paris – www.elda.org . Contact: joseph.dichy@univ-lyon2.fr

Section 7.2 begins with a short survey of different approaches to the treatment of Arabic morphology, presenting them from both theoretical perspectives and from a computational viewpoint.

Authors go on (Section 7.3) to present a typology of grammar-lexis relations, which are formalised in a computational perspective. They recall the structure of the word-form in Arabic, focusing on the far less familiar fact that two fields can be distinguished within that unit (presented in Figure 7.2):

[a] the lexical stem, or nucleus formative (NF) – except in word-forms that only include grammatical morphemes –, and
[b] extension formatives (EF-s), which are bound grammatical morphemes.

Rules and relations involved in what can be called a Word-Formatives Grammar (WFG) belong to three general types: [a] NF $\leftrightarrow$ EF and [b] EF $\leftrightarrow$ EF rules and relations, to which [c] $NF_a - NF_b$ morphological derivation links must be added. Rules and/or relations are typified and exemplified, with the purpose of presenting a mapping of grammar-lexis relations at stake in the computational morphology of Arabic.

7.2 Arabic Morphology: Theoretical and Computational Perspectives

The first module of a lexical resource is based on morphological description of the well formed internal structure of morphemes and words in the language in consideration. Grammar-lexis relations are thus dependent on what constitutes the basic units in the morphology, and how these units interact with other morphological entities to form higher and more complex word-form and syntactic structures. In this section, we give a brief account of Arabic morphology, recalling, from both theoretical and computational perspectives, some of the approaches that have dealt with the complexity of that component of the language.

Arabic morphology received a lot of attention from engineers and computational linguists particularly in the early eighties. Pioneering work on the computational morphology of Arabic goes back to the 1970s (Hlal, 1979, 1985a). The retrieval of the consonantal root from fully inflected words represented a challenge both from a theoretical point of view (Farghaly, 1987, 1994; McCarthy, 1981) and from a computational perspective that has, under certain conditions, proved liable to bring forth crucial theoretical advances and a better coverage of linguistic data, which we will endeavour to illustrate.

7.2.1 Arabic Morphology from a Theoretical Perspective

The notion of the morpheme as a meaningful string of segments delimited by the morpheme boundary symbol "+", and containing no internal morpheme boundary, is challenged by the facts of Arabic morphology, which exhibits properties that can be recalled, in very short words, as follows:

– Roots are, strictly speaking, built of three or four consonants. Each root dominates a clustering of Arabic lexical morphemes around a semantic field, which can be single, subdivided or multiple.
– Certain changes in nouns, verbs or adjectives based on these consonantal roots yield derivatives. Some vowel and syllabic patterns seem, subsequently, to be associated with a constant set of meanings.
– Traditional treatment of Arabic morphology – especially in computational morphology – sometimes remains taxonomic, abstracting away from the particular root and citing or generating all possible patterns.

These questions have been presented in the preceding chapters. Let us nevertheless recall a few points. McCarthy (1981) revisited the view according to which an Arabic verb of form I, for example, is better analysed as consisting of two separate linguistic units: a consonantal root and a vocalism (Cantineau, 1950a, 1950b). He proposes that each should be assigned to a different tier. Together, they make a prosodic template. McCarthy also mentions the fact that there are certain constraints that apply to the root: some Arabic roots, for instance, may reduplicate the second radical as in *šad~a,* شدَّ 'to pull' and *haz~a,* هزَّ 'to shake', but never the first.[3] (Such facts have been described at length in medieval Arabic linguistic treatises.) Founding their discussion on the facts of Arabic morphology and other languages, McCarthy and Prince (1996) argue that a templatic morphology based on prosodic theory can better accommodate the properties of the non-concatenative nature of Arabic morphology and that of some other languages. Farghaly (1994) suggests that the Arabic lexicon may consist of underspecified entries to represent the discontinuous nature of Arabic morphemes. Farghaly (1987) argues that an adequate description of Arabic morphology has to recognize three levels: (a) that of the root, which is neither pronounceable nor belongs to any grammatical category, (b) that of the stem, which is pronounceable and has to be a member of the word classes of the language, and (c) that of the inflected word, where inflectional affixes are attached observing well defined rules to form the majority of actual Arabic words.

In the same period, many crucial theoretical and descriptive developments founded on other approaches occurred in the computational morphology of the language.

[3] See, for instance, *Al*-suyuwTiy~ (d. around 1505), *Al-muzhir* (المزهر للسيوطي) – a medieval linguistic treatise known by most readers with general knowledge in Arabic grammar or linguistics, the title of which cannot be relevantly translated.

7.2.2 Arabic Morphology in a Computational and Theoretical Perspective

The fact that Arabic word formation involves not only attaching prefixes and suffixes to stems, but also a large number of infixes with many morphophonemic processes, makes recovering the root and analyzing the internal structure of Arabic words a real challenge for both computer processing and linguistic theory and description. Linguists, engineers and computational linguists took up the task of the analysis and/or generation of Arabic words early on. In this section we present a brief description of the main approaches in the treatment of Arabic morphology.

7.2.2.1 The Root and Pattern Approach

The ‘root and pattern’ approach has already been presented in preceding chapters, and also in Dichy and Farghaly (2003). We will therefore focus very shortly, in this subsection, on historical aspects. The ‘discovery’ of consonantal Semitic roots by Western Semiticists goes back to the XVIIIth-XIXth cent. French traveller and Orientalist Constantin Volney (Rousseau, 1987). Linguistic discussion of the question including many references, can be found, for Arabic in Dichy (1990, 1993), and in Cassuto & Larcher (2000) for Semitic studies in general.[4] The partly traditional notions of ‘root’ and ‘pattern’ should by no means be abandoned, but they need to be limited and submitted to the constraints of formal definition (the set of which is proposed in (Dichy, 2003)). Decisive psycho-cognitive evidence has been given on roots and patterns in Hebrew (Bentin & Frost, 1995; Frost, Forster & Deutsch, 1997), and on the role of roots in the recognition of Arabic written words (Grainger et al., 2003). In the second half of the XXth century (on the whole, after Cantineau (1950a, 1950b)), most linguists and grammarians of Arabic and akin Semitic languages – in the West and in Arab countries alike – came to regard consonantal roots and patterns as basic linguistic components of the morphological description of the languages in consideration. Most researchers and linguistic engineers posited patterns, which are called in Arabic *ÂawzaAn /’awzān/* أوزان (originally: ‘weight, measure, balance, poetic meter’), as presenting formal definitions of well formed Arabic words. These patterns were – and in many projects still are – considered as applicable to any root to generate Arabic lexical entries. D. Cohen (1961) gave a very elegant formulation of this view, which has later been described as a ‘neo-Leibnitzian myth’ (Dichy, 1993). It is nevertheless crucial to note that some of the forms which could be generated by patterns may have never existed in the Arabic language, and represent, as such, lexical gaps in the Arabic lexicon, which can be used to coin new words as needed, instead of

[4] On roots and patterns in the medieval Hebraic tradition, see, among many others, Zwiep (1996), in modern Hebrew dictionaries, Cassuto (2000); in the Arabic tradition, Troupeau (1984); also: Roman (1999), pp. 198–205 – “Brève histoire de la langue arabe”), which includes a strong refutation of the conjecture on roots as non-ordered consonant triples formulated by Ibn Jinniyy (IVth/Xth century), or as bi-consonantal ‘roots’ or ‘etymons’ constituted of non-ordered pairs taken up in the XXth century by G. Bohas.

borrowing foreign words that may violate the morphological and/or phonological rules of Arabic (Fassi Fehri, 1997).

The pioneering work of D. Cohen (1961) introduced a sophisticated representation of Arabic word-form structure, some revisited essentials of which are still in force to-day (Section 7.3 below). Hlal (1979, 1985a), Geith and El Saadany (1987) and many others designed computer systems for the analysis and/or generation of Arabic words relying heavily on the traditional description of Arabic morphology in terms of roots and patterns. The main approach, which has been followed with some variations, was to compile a dictionary of Arabic roots and dictionaries of affixes while maintaining a distinction between prefixes, suffixes and infixes, or to build lexicons of roots and patterns, to which lists of pre- or suffixed elements were added. Continuous look up of elements that could belong to any of these classes is then supposed to yield an analysis of Arabic word-forms.

7.2.2.2 The Lexeme-based Approach

Soudi et al. (2001) propose adopting a lexeme-based morphology, and describe MORPHE, which is a morphological rule compiler for implementation. The lexeme is an abstract concept representing a lexical meaning. Word-forms that share the same lexical meaning are related to a lexeme as members. For example, WORK is a verbal lexeme that includes as members: *work*, *works*, *worked* and *working*. All four word-forms share the lexical meaning ('working'). The variations among them are grammatical, such as past tense versus non past, etc., but not lexical.

An interesting question is: where does the Arabic root fit in a lexeme-based theory? Can we regard the root as a lexeme? The root represents a broad semantic field. In a Lexeme-based model (Aronoff, 1994) all the word forms of a lexeme belong to the same word class whereas the words generated by a particular Arabic root belong to various word classes. Clearly, two different verbs like *kataba,* كتب 'to write' and *Ǎiktataba /'iktataba/,* اكتتب 'to enter one's name, to subscribe, to contribute, to invest in' respectively belong to two different lexemes although they are clearly related to the same root. This root also includes the verb *Ǎistaktaba /'istktaba/,* استكتب 'to get someone to write', 'to dictate to someone', which partly shares the same meaning as *kataba*, but has a different argument structure. This implies that roots should not be regarded as lexemes à la Aronoff, which raises the question of what is exactly a 'lexeme' in Arabic. One possible answer (Soudi et al., 2001) is that it can, as is the case in English, be defined as an abstract concept covering all the different grammatical forms of a given stem. Thus *katabnaA,* كتبنا 'we wrote' – *yaktubu,* يكتب 'he writes' – *sayaktubu,* سيكتب 'he will write', etc., respectively belong to one lexeme since they all share the same lexical meaning and they only vary in tense, which is a grammatical feature. This otherwise efficient lemmatization procedure nevertheless leaves unanswered the question of the grouping of lexemes sharing a same root in a 'morphological family', or the issue of the derivational role of patterns, as well as pattern-to-pattern derivational links, within a given root. Such a grouping of lexemes had already been outlined within a given root. Such a grouping of lexemes had already been outlined in Hassoun (1987) and further described in Dichy and Hassoun (1989).

7.2.2.3 The Machine Learning and Statistical Approach

Machine learning approaches to building NLP systems have become very popular in recent years. While rule-based NLP systems are usually time-consuming and require solid linguistic expertise, machine learning techniques are deemed to be fast, inexpensive, and requiring only large corpora. Surprisingly, machine learning techniques produce impressive results in a very short time and without the need for expensive linguistic knowledge (Forster et al., 2003) – although doubts could be raised in the case of languages for which heavy rule-based computational linguistic work has been conducted prior to the use of statistic-based methods. The fact is, one does not witness purely statistical systems, but rather mixed statistical and rule-based approaches (such as Dien et al., 2003). As has been mentioned in the final discussions of the IXth Machine Translation summit conference (New-Orleans, Sept. 2003), purely statistical methods may not at all yield the same results in less studied languages.

For languages like Arabic where solid computational linguistic knowledge and elaborate language resources (lexica, annotated corpora, tree-banks, etc.) are still rare (Nikkhou, 2004), statistical approaches nevertheless came to the rescue when national security needed to deal with millions of documents in Arabic, and little R&D funds, compared with 'big' languages such as English, Spanish, French or German. The underlying assumption here is that linguistic knowledge is present in linguistic data and that machine learning techniques can extract this knowledge going through cycles of training and retraining until it 'learns' the language. This assumption is immediately limited by the fact that researchers usually mention the existence of supervised learning modes, where the training data are annotated, thus facilitating the learning process (for instance, Schafer & Yarowsky (2003)). Effective annotation of corpora needs to be based on heavy previous linguistic development, traditional grammar being, for such a purpose, very far from being state-of-the-art, especially in Arabic, where traditional medieval grammar has not been sufficiently revisited in the light of modern linguistics. One can nevertheless mention unsupervised learning techniques, which are very important when annotated corpora of the language are unavailable.

The recent availability of parallel corpora for Arabic-English prompted many researchers to use machine learning techniques to salvage all kinds of linguistic information. For example, Diab and Resnik (2001) describe how they used a parallel corpus for word sense disambiguation under the assumption that different meanings of the same word in the source language will be translated into distinct words in the target language.

Rogati et al. (2003) followed the unsupervised learning approach for developing a prototype of a non-English Arabic stemmer. Their objective is to build a language-independent stemmer. The model they use is based on statistical machine translation using an English stemmer and a small parallel corpus for training purposes. Their main approach is to remove prefixes and suffixes from Arabic words. Although they did not remove infixes nor deal with morpho-phonological transformation, they report achieving an improvement of 22–38% on average over unstemmed text and 96% of the performance of a proprietary stemmer built using rules, affix lists and human annotation.

7.2.2.4 The Stem-based Approach

The above approaches aim at analysing and/or generating Arabic word-forms. The problem in any Arabic NLP system, such as tagging, document categorization, automated summarization, speech recognition, machine translation, etc., is that it is not enough just to recognize or generate forms. In NLP programs related to effective application results, important information needs to be associated with each morpheme and lexical entry. There is information coming from the morphological level, such as gender, number, person, mode and tense, definiteness, part of speech (POS), etc. There are also syntactic features, such as Count/Mass nouns, subcategorization frames, what type of a subject or an object a verb takes, etc. One will also have to add semantic information, such as categorizing a noun as referring to human/non human, animate/inanimate entities, or to place and/or time, etc.

The more elaborate the information associated with the lexical entry, the more sophisticated the grammar becomes, and the more powerful the NLP system as a whole turns out to be. In machine translation applications, for instance, such sophisticated linguistic information cannot be done without. It can, on the other hand, never be associated with an Arabic root or with a pattern, because neither Arabic root nor pattern belongs to word classes (the terms refer to linguistic abstraction, not to actual parts of speech). However, combined roots and patterns may form the nucleus of nouns, verbs and adjectives. The linguistic information under consideration, including indispensable grammar-lexis relations for Arabic NLP applications, can only be associated to stems, since a stem, by definition, belongs to a syntactic category, and never to roots or to a mere combination or root and pattern (Dichy & Farghaly, 2003). In the case of a homograph (a very frequent case in standard vowel-free Arabic writing), a given stem could belong to several syntactic categories.

The stem-based approach to Arabic morphology reduces the complexity of Arabic word structure, eliminates large numbers of lexical gaps, and makes it possible to associate relevant and specific morphological, syntactic and semantic features with each entry. Figure 7.1 shows a small subset of the morphological information associated with the lexical entry of an Arabic verb in the SYSTRAN monolingual Arabic dictionary, built as a component of the Arabic-English translator.

7.2.2.5 Stems, Based on Roots and Patterns

One of the most advanced treatments of Arabic morphology using both the root-and-pattern and stem approaches is the work done at Xerox Research Centre in

France by K. Beesley and his colleagues (Beesley, 2001). Beesley's implementation is based on the insights of Karttunen (1994) that morphotactics and variations in morphology can be expressed in regular expressions and can then be compiled into finite state automata which are very efficient, fast and bidirectional. It elaborates on Kimmo Koskenniemi's two-level morphology (Beesley & Karttunen, 2003; Karttunen & Beesley, 2005), on the basis of a partial revisiting of McCarthy's representation (Beesley, 1989), and integrates a first version of Tim Buckwalter's lexicon (Buckwalter, 2002).

The approach should therefore not be described as founded on mere root and pattern combination: in fact, it includes as an essential step the checking of candidate entries generated from root and pattern 'merging', and pre- or suffixes combination, against existing lexical entries, as attested by a reference dictionary such as Hans Wehr (1979), and fully takes into consideration the complexity of Arabic morphology. The Xerox Arabic Lexicon included, four years ago (Beesley, 2001): 4,930 roots, 400 patterns, and 90,000 stems based on roots and patterns. The latter correspond to 70,000 root-pattern intersections on the lower side of the two-level morphological representation, the differences depending on information associated to stems on the higher side (see Beesley & Karttunen (2003)), which clearly shows that, in this approach, morphological and word-form grammar-lexis information is associated to stems. The figure of 90,000 stems also shows that the blind combination of roots and patterns (4,930 roots x 400 patterns = 1,972,000 root-pattern virtual links) has been severely restricted.

```
[perfect=زَرَعَ],[imperfect=يَزْرَعُ],[imperative=إِزْرَعْ],
[passperf=زُرِعَ],[passimperf=يُزْرَعُ]
```

Fig. 7.1. A sample of SYSTRAN's monolingual dictionary entry of the Arabic verb *zaraça* زرع 'to plant'

7.2.2.6 Stem-based Lexical Resources, Including Root-Pattern and Grammar-lexis Information

Another advanced treatment of Arabic morphology was initiated in France in the early 1980s, in what was known as the SAMIA project[5] (Desclés et al., 1983; Dichy, 1984, 1987; Dichy & Hassoun, 1989; Hassoun, 1987), and has been going on since. It has led to the completion, in collaboration with a Tunisian research centre (IRSIT, now IT.COM), of the DIINAR.1 Arabic lexical resource. Morphological analyzers drawing on this resource have been completed on a parallel basis. The approach can be described as deliberately stem-based, including root and pattern information on the one hand, and a comprehensive coverage of word-form grammar-lexis relations on the other. This makes it closer to the requirements of

[5] SAMIA is the acronym for "Synthèse et Analyse Morphosyntaxiques Informatisées de l'Arabe".

machine translation, such as has been illustrated, in many couples of languages, by the SYSTRAN engines.

In this approach, representations of word-form structures and of word-level grammar-lexis relations are very explicit. This is due to the database structure of DIINAR.1, the subsequent declarative programming of the associated software (Abbès, 2004; Ghenima, 1998; Ouersighni, 2001; Zaafrani, 2002), and also, to the comprehensiveness of the coverage of Arabic morpho-lexical data.

The concepts and methods at stake in that representation of Arabic computational morphology, which is centred on grammar-lexis relations, are presented in some detail in the following section.

7.3 Mapping the Arabic Lexicon: Word-form Structure, Rules and Grammar-lexis Relations in Arabic

7.3.1 Structure of the Word-form in Arabic (Short Recall)

Word-form units feature in Arabic a complex, albeit very regular, structure. Standard word-forms comprise one lexical nucleus and one only.[6] On the right and left sides of that nucleus, specified sets of bound morphemes can be found, in either affixed or cliticized position (Cohen, 1961; Desclés, 1983; Dichy, 1990, 1997; Dichy & Hassoun, 1989).

The structure encompasses:[7]

– **proclitics (PCL),** which consist of mono-consonantal conjunctions (such as *wa-*, وَ- 'and' , *li-*, لِـ 'in order to'), prepositions (i.e. *bi-,* بِـ 'in', 'at' or 'by', *li-*, لِـ "for"), the pre-verb *sa-,* سَـ (indicating the future), the article *Al- /'al/,* الـ etc.;
– a **prefix (PRF)**. The category, after D. Cohen's representation of the word-form (1961), only includes the prefixes of the imperfective, such as *Âa- /'a/*, أ-, morpheme of the 1st person sing., etc.;
– a **stem.** Stems are divided in two general categories (Dichy, 1984):

  • **Type 1 stems**: this first subset consists of major lexical categories that are liable be represented in terms of a PATTERN and of a 3-consonant or 4-consonant ROOT. (Major lexical categories encompass nouns, adjectives,

[6] Poly-lexical entries are, in Arabic, either composed of more than one word-form (e.g. *Al-quruwn Al-wusTý, /'al-qurūn al-wusTā/* القرون الوسطى 'the Middle Ages') or reduced by the morphological system of the language to a mono-lexical unit, e.g. *qarwasaTiy~,* قروسطي 'medieval'. The meaning of the Arabic lexicographical term *naHt,* نحت 'coinage', which describes the phenomenon, refers to the above reduction, which brings the compound to comply with (a) the model of 3-consonant or 4-cons. roots, and (b) the structure of the mono-lexical word-form (Dichy, 2003).

[7] Hebrew word-forms feature similar complex structures (Sampson, 1985), pp. 90–91; for a psycholinguistic approach, see Frost, Deutsch & Forster (2000).

verbs and deverbals.)[8] By convention, the terms ‘root’ and ‘pattern’ will be henceforth presented in small capital letters, referring to the formal definition above (Dichy, 2003). A **ROOT** is an ordered triple of consonants (3-C) or, by extension of the system, a quadruple (4-C).[9] A **PATTERN** is, in short words, a template of syllables, the consonants of which are that of the 3-C or 4-C ROOT, with the addition of mono-consonantal affixes (belonging to mono-consonantal roots[10]), such as the *t* ‘echo-morpheme’ (Roman, 1990). Consider for instance the stem *takab~ar,* تكبَّر ‘to be haughty’. This stem can be analysed into the 3-C ROOT /k-b-r/, and the PATTERN /taR^{1}aR^{2}R^{2}aR^3/ (تَفَعَّل), which includes the mono-consonantal root /*t*/ and the 1st, 2nd and 3rd consonant of the 3-C ROOT (respectively: R^1 = *k*, R^2 = *b*, R^3 = *r*). It is crucial to remember that type 1 stems include *all* the verbs and deverbals of the language (Dichy, 1984).

- **Type 2 stems:** the second subset of stems contains only nouns that cannot be represented in terms of PATTERN and ROOT, such as: */’ismāçīl/,* إسماعيل ‘Ishmael’, *fiyziyaA',* فيزياء ‘Physics’. There are *no verbs* in this category of stems (a corollary of the fact, which has just been mentioned, that all verbs belong to type 1 stems);

– **suffixes (SUF)**, such as verbal inflexions, nominal cases, the nominal feminine ending *+aħ, /a(t)/* ـَة+, etc.;

– **enclitics (ECL)**. In Arabic, enclitics are complement pronouns. Some verbs can have a double ECL, for example: *çal~am+tum-uw-niy-haA, /çallam+tum-ū-nī-hā/,* علّمتمونيها “you [plur. masc.] have taught-me-it” (this /ū/ sequence

---

[8] The term ‘**deverbal’** refers to what could also be called ‘verbal-nominal forms’, i.e., nominal forms that include syntactic-semantic verbal features, such as transitivity, etc. These are, in Arabic, the infinitive form, *maSdar,* مصدر, the active participle, *Ǎism Al-faAçil, /ism al-fā‘il/,* اسم الفاعل, and the passive participle, *Ǎism Al-mafçuwl, /ism al-maf‘ūl/,* اسم المفعول. Note that other subcategories have been included in deverbals in the DIINAR.1 lexical resource (see § 7.3.3.2, Figure 7.4), following the categorisation of traditional Arabic grammar. This has proven not to be consistent beyond morphological analysis. Concerning the three subcategories above, research conducted in the DIINAR.1 project has shown that traditional Arabic grammatical terminology obscures the fact that the forms in consideration are liable to be *either* deverbals, *or* nouns. Consider for instance the sentence *ÂanaA saAkin fiy ruwmaA, /’anā sākin fī rūmā/* أنا ساكن في روما, ‘I’m living in Rome. The active participle *saAkin* (ساكن) admits suffixed plural forms, e.g., *naHnu saAkinuwn* (masc. ساكنون) or *saAkinaAt* (fem. ساكنات) *fiy ruwmaA*, but excludes the ‘broken plural’ form *suk~aAn,* سكّان which refers to the meaning of ‘inhabitant(s). The former is a deverbal, the latter (*saAkin,* plur. *suk~aAn,* ساكن, ج , سكّان) has undergone a nominalization process, i.e. has left the deverbal category to become a ‘purely’ nominal lexical entry (see § 7.3.5.1[b]). Such cases require two distinct entries, each associated with its own grammar-lexis specifiers.

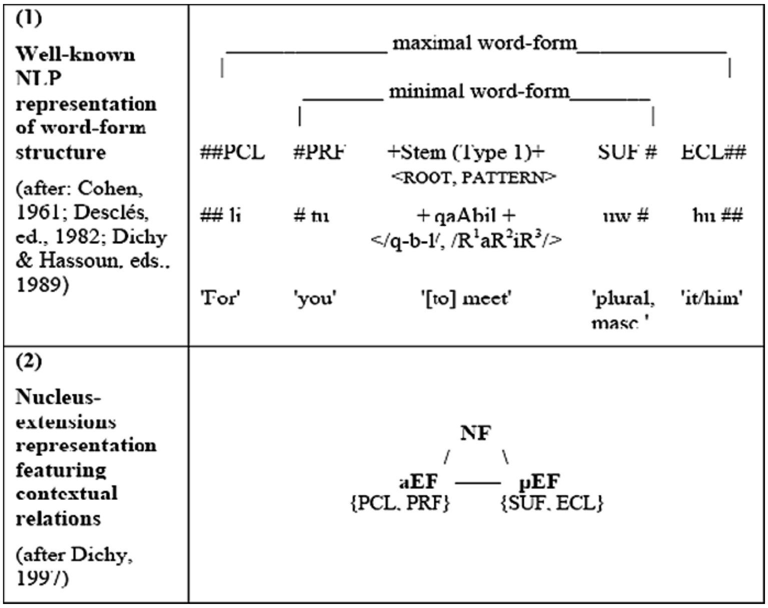
[9] 5-consonant so-called ‘roots’, included for instance in *Ǎiçranfaza, /’ içranfaza/,* اعرنفز ‘to almost die from cold’, which can only be found in ancient poetry or medieval dictionaries, have been neglected.

[10] Mono-consonantal roots in Arabic and Semitic languages have been disclosed by Roman (1990, 1999).

only appears with the plural masculine form of the subject pronoun when an ECL pronoun is attached).

Figure 7.2 (Dichy, 1997) illustrates this structure, in the case of Type 1 stems (conventions in the lower part are explained immediately after):

Conventions (not already encountered here): ## = ‘word boundary’; # = ‘clitic boundary’; + = ‘pre- or suffix boundary’. NF = ‘nucleus formative’ (referring to the lexical nucleus of the word-form); EF = ‘extension formative’ (referring to



(1) Well-known NLP representation of word-form structure (after: Cohen, 1961; Desclés, ed., 1982; Dichy & Hassoun, eds., 1989)	maximal word-form / minimal word-form
(2) Nucleus-extensions representation featuring contextual relations (after Dichy, 1997)	NF / aEF —— pEF / {PCL, PRF} {SUF, ECL}

##PCL	#PRF	+Stem (Type 1)+ <ROOT, PATTERN>	SUF #	ECL##
## li	# tu	+ qaAbil + </q-b-l/, /$R^1aR^2iR^3$/>	uw #	hu ##
'For'	'you'	'[to] meet'	'plural, masc.'	'it/him'

Fig. 7.2. Arabic word-form structure (with ROOT-and-PATTERN stems) – لتقابلوه

bound grammatical morphemes); aEF, pEF = ‘ante-positioned’ or ‘post-positioned’ EF. The set of aEF-s includes {PCL, PRF}, that of pEF-s comprises {SUF, ECL}.

7.3.2 Word Formatives, Word Specifiers and Word Formatives Grammar

The **Word Formatives Grammar (WFG)** accounts for the rules and relations that ensure correct combination of formatives within the boundaries of the word-form (Dichy, 1987). This grammar includes morpho-phonological transformation rules, and various contextual rules, which will be outlined below (Subsection 7.3.4). Phonological transformations were not accounted for in the morphological analysis of vowelled Arabic words initiated by Cohen (1961) or Hlal (1979, 1985a). They have on the other hand been included in the approach developed in the SAMIA project for the analysis or the generation of vowel-free word-forms, and the subsequent elaboration of the DIINAR.1 lexical database. One of the postulates of this approach is that linguistic formatives must be specified in terms of morpo-syntactic rules and relations according to the syntagmatic extension of the unit they are inserted in (Dichy, 1987, 1997). Owing to the structure of the Arabic word-form, one is brought to give special attention to rules and grammar-lexis relations, accounting, in short, for insertion rules operating within the scope of that syntagmatic unit. The following concepts and conventions have been subsequently adopted:

– *Word formatives*, i.e., morphemes considered in the frame of the word-form structure, are associated with grammar-lexis **word-specifiers (w-specifiers)**.
– *Sentence formatives* need to be associated with **s-specifiers**, and *text formatives*, with **t-specifiers**.

This can be considered as an overall framework. It could easily be shown that the three types of specifiers above involve different types of phenomena (Dichy, 2005). Specifiers involving word and sentence formatives can be described as **morphosyntactic specifiers**.

7.3.3 Grammar-Lexis Relations in the Processing of Written Arabic

7.3.3.1 Multiple Analyses at Word-form and Sentence Level

Let us recall that, in the morphological analysis of Arabic, the complex operation referred to as the ‘segmentation’ of the word-form into morphemes (or formatives) is rendered the more difficult because standard writing is ‘unvowelled’ or ‘vowel-free’, i.e. bare of *secondary diacritical signs* indicating short vowels (*HarakaAt,* حركات), consonant doubling (*šad~a,* شدّة), diacritical case-endings (*tanwiyn,* تنوين), etc. This has been presented in previous chapters. It has been shown in some details, quite a few years ago (Desclés, 1983; Dichy, 1984), that the resulting homographs entail a high number of potential *existing* analyses for a relevant percentage of word-forms (Abbès, 2004; Ghenima, 1998).[11] This is indeed the case because computational morphology, when it is not included in a syntactic analyzer (Ditters, 1992; Ouersighni, 2001, 2002), deals with word-forms context-free.

Ambiguity due to multiple analyses should subsequently *not* be considered a problem in itself: morphological and morpho-syntactic analyzers aim at assigning word-forms with *all* the analyses that comply with the rules and lexicon of the language, and them only. It is on the other hand necessary to restrict the combination of word-formatives to forms that are ‘legal’ according to the morpho-syntactic system (including the morphotactics of the writing system) and the lexicon of the language. This is also required to prevent the number of analyses per word-form to climb much higher than allowed by the language and its writing

[11] This could also be tested, in addition to the morphological analyzers drawing on the DIINAR.1 resource (Abbès, 2004; Ghenima, 1998; Zaafrani, 2002), with the morphological analyzer put on the Internet by the Xerox European Research Centre (Beesley, 2001).

system. Non existing analyses should therefore be ruled out: the vowel-free word-form *’çlnt,* أعلنت, for instance, should not be analysed as **Âaçluntu, */’açluntu/,* أعلُنْتُ or **Âaçlunat, */’açlunat/,* أعلُنَتْ (no meaning in both cases), but – among other forms – as *Âaçlantu, /’açlantu/,* أعلَنْتُ or *Âaçlanat*, */’açlanat/,* أعلَنَتْ ‘I’ or ‘she declared publicly’.

7.3.3.2 Restricting Generated Lexica

Ruling out what could be described as ‘morphological noises’ is an equally crucial issue when it comes to restricting the number of entries of a lexical resource.

Let us consider a few figures:

1) In the DIINAR.1 lexical resource, the number of combined proclitics and suffixes which are effectively in use in Modern Standard Arabic, and that of prefixes and enclitics is shown below (Abbès, Dichy & Hassoun, 2004):

Comments:

(a) Prefixes do not combine (see § 7.3.1 above).
(b) Enclitics may combine in doubly transitive verbs, which seldom occurs in present-day Arabic, where one of the complements is usually preceded by a preposition; e.g.: Ancient Arabic *manaH+tu-ka-hu,* منحتكه ‘I have given_you_it’ is currently realized as *manaH+tu-hu la-ka,* منحته لك ‘I have given_it to_you’.

Proclitics (combined)	64
Prefixes	8
Suffixes (combined)	67
Enclitics	13

Fig. 7.3. Number of EF-s in DIINAR.1

(c) In the above numbers, extension formatives (EF-s) combinations have been restricted to effective use. Ancient Arabic proclitic combinations, such as *Âa-fa-bi-ka-Al-, /’a-fa-bi-ka-’l/* أ/فـ/بـ/كـ/الـ ‘interrogative-then-by-such_as-the (generic article)’, or *bi-ka*, ‘by-such_as’, and a few others, have not been included.
(d) In Figure 7.3, suffixes that only include secondary diacritics, (traditionally called ‘vowel-signs’, *HarakaAt,* حركات), i.e. basic case-endings in nouns and a subset of mode markers in verbs, have not been included. This ensures a ‘lower-hypothesis’ interpretation of the calculations presented in the demonstration below.

2) The number of lemma-entries belonging to major lexical categories is the following:

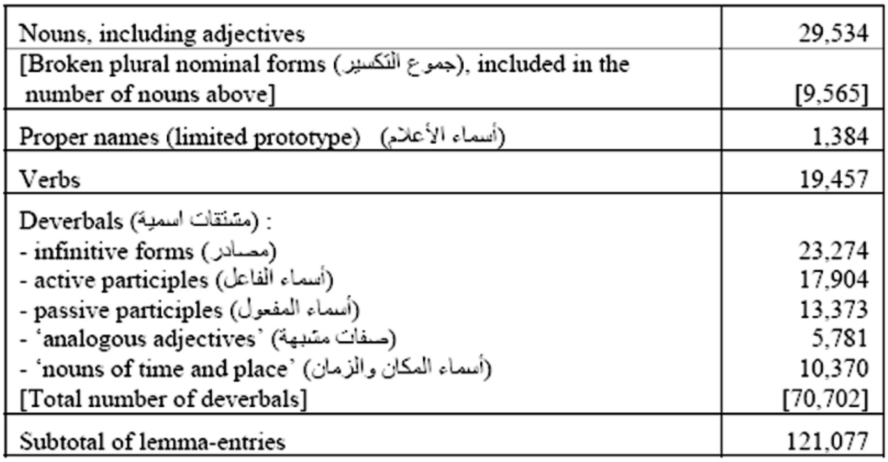
Comments:

In the DIINAR.1 lexical resource, following traditional Arabic grammar, adjectives have been included in nominal stems, and two morphological subcategories have been added to deverbals. These are, as shown in Figure 7.4: (a) ‘analogous adjectives’ (صفات مشبهة) and (b) ‘nouns of time and place’ (أسماء المكان والزمان). Both categorisations, which remained acceptable in the context of computational morphology, have proved inconsistent when extending grammar-lexis relations to syntactic features. Clearly, (a) are adjectives and (b) are nouns. This bias can be corrected in the related analyzers and generators, through modifying the (sub)category in specifiers associated with lexical entries.

Let us now undergo a bit of *ab absurdo* reasoning:

On the basis of Figure 7.3, blind combination of bound grammatical morphemes would give:

64 × 8 × 67 × 13 = 445,952 ‘virtual’ extension formatives (EF-s).



Nouns, including adjectives	29,534
[Broken plural nominal forms (جموع التكسير), included in the number of nouns above]	[9,565]
Proper names (limited prototype) (أسماء الأعلام)	1,384
Verbs	19,457
Deverbals (مشتقات اسمية) :	
- infinitive forms (مصادر)	23,274
- active participles (أسماء الفاعل)	17,904
- passive participles (أسماء المفعول)	13,373
- ‘analogous adjectives’ (صفات مشبهة)	5,781
- ‘nouns of time and place’ (أسماء المكان والزمان)	10,370
[Total number of deverbals]	[70,702]
Subtotal of lemma-entries	121,077

Fig. 7.4. Number of lemma-entries in the DIINAR.1 Lexical resource

Unconstrained combination with the total number of stems in Figure 7.4 leads to a generated lexicon of ‘virtual’ word-forms of:

445,952 EF-s × 129,258 stems = 57,642,863,616 ‘virtual’ word-forms.

Limiting the figures to inflected forms, the combination would still yield:

8 prefixes × 67 suffixes × 129,258 stems = 69,282,288 ‘virtual’ forms.

In the DIINAR.1 resource, the effective number of inflected word-forms is 7,774,938 (Abbès, Dichy and Hassoun, 2004, which includes a breakdown according to lexical categories), i.e. 11.22% of the ‘virtual’ figure above.

As for the lexical nuclei of word-forms, knowing that the amount of ROOTS in DIINAR.1 is 6,546, with an estimated number of 400 patterns, the figure would be:

6,546 ROOTS × 400 PATTERNS = 2,618,400 ROOT-PATTERN virtual links.

This comes against 119,693 lexical *existing* lemma-entries (and 129,258 existing stems including 'broken plurals', proper names being for obvious reasons left out). We have seen in § 7.2.2.5 that the Buckwalter-Xerox lexicon includes 90,000 entries, based on 4,930 ROOTS and 400 PATTERNS, the blind combination of which would have led to 1,972,000 ROOT-PATTERN virtual links.

Remarkably – knowing that sources and research contexts did in fact differ –, the ratios of overall entries per ROOT in the two lexical resources are next to identical:

– DIINAR.1 lexical database: 119,693 entries / 6,546 ROOTS = 18.28
– Buckwalter-Xerox lexicon: appr. 90,000 entries / 4,930 ROOTS = 18.25.

One is therefore brought to the conclusion that the 'virtual results' above are not only absurdly enormous, they are also blurred: for lack of explicit decision procedures, there would be no way in which a given analysed or generated 'form' could, or not, be deemed part of the language. Restricting generated lexica through rules involving grammar-lexis relations associated to actual lexical entries is therefore necessary both for computational generation and analysis, and in the building of efficient lexical resources. Let us now consider the general types of rules and relations that are valid within the boundaries of Arabic word-forms.

7.3.4 General Types of Rules and Relations in the Word-Formatives Grammar

7.3.4.1 The Three Types of Contextual Relations Involved in the WFG

Rules involving word-formatives (NF and EF-s) are based on three types of relations (Dichy, 1987): ⇒ **'entails', ≠> 'excludes', ** 'is compatible with'** or **'admits'**, the third of which is attached to the opposed pair of the first two as an 'elsewhere' relation of a special kind, directly connected to ambiguity in language analysis processes. In generation, all 'compatibility' (or 'admit') relations can in fact be rewritten in terms of either 'entail' or 'exclude' rules. 'Compatibility' relations are mostly useful in the formalization of recognition rules, when ambiguity is at stake. They express relations that only appear in analysis.[12]

It is essential to note that automatic analysis and generation of linguistic data are *not* to be considered as reverse processes (Desclés, 1983; Dichy, 1984, 1990, 1997).

[12] Developments on ambiguity in Arabic NLP have been presented in Dichy (1990, 2000). Statistics on ambiguity in 'unvowelled' written Arabic are given in Abbès (2004). For a general reference on ambiguity in Arabic, see Arar (2003), and A. Farghaly's contribution on "Lexical Ambiguity in Arabic Machine Translation Systems" in the same volume.

7.3.4.2 Rules Related to the Two General Fields of the Word-form Structure

Grammar-lexis relations are connected to the **word-Formatives Grammar (WFG)**, which accounts for the rules and relations involved in Arabic word-form structures (Dichy, 1987, 1990). In the well-known representation recalled in the upper half of Figure 7.2 (see § 7.3.1 above), two general types of word-formatives can be distinguished:

– Formatives pertaining to the bound grammatical morphemes of the language, and encompassed within the boundaries of the word-form, are called **EF-s (Extension Formatives)**.
– The lexical nucleus of the word form (except in word-forms that only include grammatical morphemes) is called a **NF (Nucleus Formative)**.

The WFG includes, accordingly, three types of rules and/or relations, which are directly attached to the triangle featured in the lower part of Figure 7.2. By convention, in ‘PCL $\rightarrow$ SUF’, the arrow ‘$\rightarrow$’ can be read either as ‘determine’ (either $\Rightarrow$ ‘entails’, or $\neq>$ ‘excludes’) or ‘are compatible with’ (**), with reference to the three types of rules mentioned in § 7.3.4.1. The two-headed arrow ‘$\leftrightarrow$’ is read in the same way, with the addition of ‘reciprocity’ (‘and vice-versa’).

Types of rules and relations involved are:

[a] **EF $\leftrightarrow$ EF contextual rules and relations**, such as PCL $\rightarrow$ SUF rules, e.g.:

PCL = Prep. {*bi#, li#*} $\Rightarrow$ SUF = {*+i, +in,+a, +n, +iyna, +iy, +ayni, +ay*}

which can be phrased as: ‘*if* the proclitic is a preposition (i.e., a member of the set between braces), *it follows* (or: *this entails*) that the suffix is one of the indirect (or genitive, *majruwr* مجرور) case suffixes’, the set of which is listed between braces. The selection of the correct case-ending in the list is performed through different types of morphological and syntactic rules.

[b] **NF $\leftrightarrow$ EF rules and relations**. A simple example involves the major lexical category to which the NF belongs, such as PCL $\rightarrow$ NF category. The above rule, for instance, needs to be completed by the following:

PCL = Prep. $\Rightarrow$ NF = Noun.

[c] **NF – NF relations** are morphological derivation links, which are, in a great number of cases, not rule-predictable. Consider, in nouns, singular – ‘broken plural’ links, for instance: sing. *kitaAb,* كِتاب – plur. *kutub,* كُتُب ‘book(s) vs sing. *sinaAn,* سِنان – plur. *Âasin~ah̄, /’asinna(t)/,* أسِنَّة ‘spearhead(s)’, sing. *HimaAr,* حِمار – plur. *Hamiyr,* حَمير *Humur,* حُمُر and *ÂaHmirah̄, /’aHmira(t)/,* أحْمِرة ‘donkey(s)’. In these nouns, the PATTERN of the singular is /R^{1}iR^{2}āR^3/, فِعال ; the PATTERNS of ‘broken plurals’ appear to be, sometimes /R^{1}uR^{2}uR^3/, فُعُل, sometimes /’aR^{1}R^{2}iR^{3}a(t)/, أفْعِلة, and sometimes another ‘broken plural’ form, including, in some cases, a suffixation plural form, e.g.: *qiTaAr,* قِطار ‘train’, shows two plural forms, *quTur,* قُطُر , which pertains to the

/R^{1}uR^{2}uR^3/, فُعُل PATTERN of 'broken plurals', and *qiTaAraAt,* قِطَارَات, which is constructed with the suffix *+aAt,* ـَات.

7.3.5 Rules of the WFG and Grammar-Lexis Specifiers

Rules and relations involved in the Word Formatives Grammar entail the need, for the entries of a lexical database, to be associated with grammar-lexis relations. The latter are essentially attached, in computational morphology, to types [b] and [c] above. As mentioned above, they are called in the SAMIA-DIINAR.1 approach, word-specifiers (w-specifiers).

7.3.5.1 The Two Types of NF ↔ EF Contextual Rules and Relations

A crucial point in the overall structure of the Arabic lexicon, which seems to have been widely overlooked, appeared in the elaboration of the DIINAR.1 resource. Grammar-lexis relations concerned with the nucleus are liable to involve, in addition to compositional combinations of nucleus and extension formatives, non-compositional ones, i.e. 'frozen' or 'lexicalized' combinations (Dichy, 1984, 1990, 1997). Let us consider these two types:

[a] NF ↔ EF compositional relations, and related w-specifiers

Compositional NF ↔ EF combinations are, on the whole, easy to grasp, although they include a few tricky aspects (see § 7.3.5.3). A simple example is that of: Stem → SUF rules, e.g.:

$$\text{Stem} = \text{diptote} \Rightarrow \text{SUF} = \{u, a, i\}$$

This rule can be rephrased as: 'a stem whose declension is diptote (*mamnuwç mina AlS~arf, /mamnūç mina S-Sarf/,* ممنوع من الصرف) entails case-endings belonging to the listed set'. An additional rule restricts the occurrence of SUF /i/ with diptote nouns and adjectives to construct-state syntactic structures (*ǍiDaAfaħ, /'iDāfa(t)/*, إضافة), e.g.: *min maçaAlimi Al-çaASimaħ, /min maçālimi Al-çāSima(t)/,* من معالمِ العاصمة 'from the monuments of the capital'. Other suffixes are accounted for in different rules.

Many other examples could be given: nouns with 'broken plurals' often exclude (≠>) suffixed plural forms; intransitive verbs exclude ECL complement pronouns; verbs that only admit non-human complements exclude a subset of the ECL-s, such as *-hum*, 3rd person plur. masc. or *-ki*, 2nd person sing. fem., which can only refer to human entities.

[b] **NF ↔ EF non-compositional lexicalized relations, and related w-specifiers**

In Arabic, as in other Semitic languages of the same family, in addition to ROOT and PATTERN derivation, one finds lexical derivation by means, essentially, of suffixation.[13] With Type 1 stems (§ 7.3.1) featuring ROOT and PATTERN combination, the two means of derivation are liable to add up. Letting aside, in the present contribution, *phrasal compound* expressions in which a given syntactic structure is frozen (as in *majmaς çilmiy~,* مجمع علمي 'Science Academy', *jamç Al-maçluwmaAt,* جمع المعلومات 'data gathering', or *ÂaHaATa fulaAnAã çilmAã bi-, /'aHāTa fulānan çilman bi-/,* أحاط فلانًا علمًا بـ 'to inform someone of'), one can distinguish between two types of lexical entries based on strict morphological means (Dichy, 1997):

[1] **'Simple' lexical entries** coincide with the stem (or nucleus), e.g.: *maktab,* مكتب "office", "bureau". The entry can, as expected, be inserted in a word-form, such as *wa-bi-maktab-i-naaA,* وبمكتبنا "and by our office" ("and-by-office-genitive case /i/-of us").

[2] **Morphological compound entries** (as opposed to *phrasal compounds*) feature a 'lexical freezing' of the combination of a given nucleus with a given extension formative. The morphological compound is coded in the database as a full entry of its own (a w-specifier is, in addition associated with the stem, in order to account for occurrences that have not undergone a 'lexical freezing' of the NF – EF relation). The lexical entry *SuHuf-iy,* صُحُفي 'journalist', for instance, is a compound entry:

(a) it does not coincide with the stem, or nucleus, it encompasses. The stem is, here: *SuHuf,* صُحُف (otherwise meaning 'sheets', 'papers'), which features a combination of ROOT /S-H-f/ and PATTERN /$R^1uR^2uR^3$/ (فُعُل);

(b) it includes on the other hand the extension formative *+iy~* (*yaA' Al-n~isbaħ, /yā' al-nisba(t)/,* ياء النسبة 'relative adjective or noun' morpheme). An analogous example, going as far back as the IInd cent. of Hijra/VIIIth cent. c.e., is *kutub+iy~,* كُتُبِيّ, 'librarian' (in the medieval meaning of the word).

– A given word can be either a frozen morphological compound, or result from composition: *jaAmiçaħ, /jāmiça(t)/,* جامعة can be analysed either as the morphological compound *jaAmiç+aħ, /jāmiç+a(t)/,* meaning 'mosque', which is linked in the lexicon with the 'broken plural' *jawaAmiç,* جوامع or as the feminine of the active participle *jaAmiç+aħ, /jāmiç+a(t)/,* meaning 'collecting', i.e. 'she who collects', or 'compiles' or 'brings together'. This is a very frequent phenomenon.

– Morphological compounds can, of course, also be inserted in a word-form, e.g. *wa-bi-SuHuf+iy~+i-naaA,* وبصُحُفيّنا 'and by our journalist' (= 'and_by_ journalist_genitive case /i/_of us').

[13] In this representation, affixed elements included in PATTERNS, such as /ma/ in *mawçid,* موعد 'promise', 'pledge' (or 'appointment', 'date') are *not* considered as prefixes or suffixes (following Cohen (1961) and Desclés (1983), as well as traditional Arabic grammar) – see § 7.3.1.

> Another example is found in proper names, such as *(Al-Kuwayt,* الكويت in which the lexicalized EF is the proclitic article *Al-*.

The above distinction is methodologically crucial. It gives additional interpretative evidence to the demonstration presented in § 7.3.3.2, according to which derivation of 'virtual' lexical entries from ROOT and PATTERN is no sufficient basis for the Arabic lexicon. But the issue goes much further: ROOT and PATTERN derivation is complemented by 'external' derivation, i.e. by the lexicalization of the NF – EF combination, which is only found in nouns (Dichy, 1984), and cannot be predicted by rules, because the process described above only occurs to answer the need, for the lexicon of a given language, to build a new entry, when a newly encountered entity requires nomination. This is correlated to the non-compositional nature of the NF – EF relation. It follows, in a computational perspective, that morphological compounds can only be recognized or generated with the help of a lexical resource.

7.3.5.2 Morphological NF – NF Derivation Links, and Related W-Specifiers

Another type of relation is NF – NF linking combinations, also called, in Semitic studies, 'internal' derivation, because these links feature a variation in PATTERN, the ROOT remaining constant. Such derivations have to be encoded as w-specifiers, whenever the morphological link is not strictly rule-predictable, which occurs in a majority of stems (Dichy, 1987, 1990; Hassoun, 1987).

This is the case, for instance, in a wide number of singular ↔ 'broken plural' links in nouns or adjectives (exemplified in § 7.3.4.2[c]), as well as in most 'perfective' ↔ 'imperfective' links (*maADĩ, /mādin/ ↔ muDaAriç, /muDāriç/,* ماض ↔ مضارع), in verbs of 'simple' PATTERNS (*Al-fiçl Al-mujar~ad,* الفعل المجرد). In the same subcategory of verbs, the verb ↔ infinitive form link (*fiçl ↔ maSdar,* فعل ↔ مصدر), and other similar ones (such as verb ↔ analogous adjective, *fiçl ↔ Sifaħ mušab~ahaħ, /Sifa(t) mušabbaha(t)/,* فعل ↔ صفة مشبّهة) also need to be encoded lexically.

Restrictions on conjugation paradigms, such as the passive or the imperative, which can be described in terms of semantic rules (based on features such as agentivity, and human/non human complements, etc., cf. Ammar & Dichy (1999), pp. 17, 19–20), also pertain to NF ↔ NF links. In a lexical resource, they need to be encoded as w-specifiers, because the entries are not actual linguistic signs (with 'signifiant' and 'signifié' features), but mere chains of characters associated with a set of linguistic specifiers (Dichy, 1997).

7.3.5.3 Morphological Derivation Links Including a Morphological Compound

In many cases, *morphological compound entries* also feature a *morphological derivation link.* For example: the morphological compound *madras+aħ, /madras+a(t)/,* مدرسة, 'school' is associated with the broken plural form *madaAris,* مدارس. By contrast, the analogous entry *maktab+aħ, /maktab+a(t)/,* مكتبة 'library', has a suffixed plural form *maktaba+At,* مكتبات. This is due to the fact that the expected broken plural *makaAtib,* مكاتب is already associated in the lexicon with *maktab,* مكتب "office", "bureau" (which is a 'simple' lexical entry – see § 7.3.5.1[b] above).

7.4 Conclusion: Exhaustive Coverage of Morphological Features, and the Question of what Lays Beyond

The third section of this contribution has presented the main types of grammar-lexis relations that can be observed at word-form level in Arabic, and produced evidence for the crucial need for a comprehensive mapping of rules and relations involving the lexical nucleus of words in the computational morphology of the language.

To make this presentation clearer, three questions remain to be answered: (1) Are the rules and relations in consideration finite in number? (2) What is the actual extent of their coverage of word-form structures, relations and rules? (3) And what lays ahead, beyond word-level analysis? These issues are crucial in a computational perspective, because they are concerned with, respectively, the *feasibility* of the task of associating a whole lexicon with w-specifiers, the *reliability* of the software drawing on the lexical resource in consideration, and the compatibility of the results achieved in computational morphology with further developments involving sentence and text analyses.

7.4.1 Finiteness in Number of W-Specifiers, and the Feasibility of their Association with the Entries of an Entire Lexical Database

Arabic EF $\leftrightarrow$ EF rules belong to finite sets for obvious reasons: EF-s are finite in number, and a finite set of relations are at stake, because EF-s belong to the grammatical morphemes of the language.

Altough lexical NF-s belong to open sets, the **finiteness of NF $\leftrightarrow$ EF w-specifiers**, i.e. of the number of w-specifiers to be associated with the entire set of lexical entries, can be demonstrated as follows (Dichy, 1997):

(a) Because EF-s belong to finite sets, it follows that a finite set of features constricting NF $\leftrightarrow$ EF relations can be established.
(b) This finite set of features, which corresponds to morpho-syntactic w-specifiers, can in turn be associated with every entry of a lexical database, i.e., to each NF.

In other word, both morphological grammar-lexis relations and the task of assigning every entry of a lexical resource with specifiers operating at word-level can be demonstrated as limited. Though lengthy, the task can subsequently be performed in a limited period of time.

7.4.2 Finiteness *and* Exhaustiveness in the Coverage of Data at Morphological Level, and the Reliability of Resulting Lexical Resources and Analyzers

Because grammar-lexis relations are liable to be embedded in finite sets of w-specifiers, it follows that the coverage of data at word (or morphological) level can be exhaustive. In other terms, *finite* sets of w-specifiers can produce *exhaustive* coverage of data within the boundaries of the word-form. This ensures a very high level of reliability in the software and analyzers drawing on a language resource such as DIINAR.1.

The above demonstration naturally goes beyond the mere case of the Arabic language. The criteria of finiteness *and* exhaustiveness in linguistic sets of features have been introduced by Mel'čuk (1982) in the general context of morphological description, and later taken up by him in lexicography. On the other hand, the demonstration also partly explains, in our view, why morphological and morphotactic rules and relations accounting for word-form generation and/or analysis, can be implemented very effectively using finite state transducers (Beesley & Karttunen, 2003; Karttunen, 1994).

7.4.3 Beyond Computational Morphology: Grammar-lexis Relations at Sentence and Text Levels

We now come to our conclusive remark, which deals with the usefulness of morphosyntactic specifiers included in rules and relations operating at word-form level in sentence and text analysis. Grammar-lexis relations at sentence level (s-specifiers), and furthermore at text level (t-specifiers) will, for obvious reasons, require different approaches. In the related resources, contextual relations need to be established between categories and sets of morphemes on the one hand, and sets of denotative and referential semantic categories and features on the other. The mapping of grammar-lexis relations already performed in the finite domain of computational morphology can nevertheless be considered a substantial progress towards lexical resources needed at sentence and text-level, owing the high degree of complexity of word-form structures in Arabic, compared to, say, French or English. W-specifiers already include a number of semantically related syntactic features needed at higher levels of analysis, such as (in)transitivity (including related prepositional structures) in verbs, type of plural according to the lexical subcategory a given nominal form belongs to and many other lexical categorisations and descriptions, which have been exemplified in various sections of this chapter.

References

Abbès, R. (2004). *La conception et la réalisation d'un concordancier électronique pour l'arabe*. Thèse de doctorat en sciences de l'information, ENSSIB/INSA, Lyon.

Abbès, R., Dichy, J. & Hassoun, M. (2004). The Architecture of a Standard Arabic lexical database: some figures, ratios and categories from the DIINAR.1 source program. In *Proceedings of the* COLING-04 *Workshop on Computational Approaches to Arabic Script-based Languages* (pp. 15–22), Geneva.

Ammar, S. & Dichy, J. (1999). *Les verbes arabe.* Paris: Hatier. Fully Arabic version, with specific introduction: *Al-'afālu l-ςarabiyya,* الأفعال العربية (same publisher and year).

Arar, M. (2003). *D̲āhiratu l-labsi fī l-ςarabiyya* [The phenomenon of ambiguity in Arabic,ظاهرة اللبس في العربية]. Amman: Dār Wā'il.

Aronoff, M. (1994). *Morphology by Itself: Stems and Inflectional Classes*. Cambridge, MA: MIT Press.

Beesley, K. (1989). Computer Analysis of Arabic Morphology: A two-level approach with detours. In Comrie, B. & Eid, M. (Eds.) (1991), *Perspectives on Arabic Linguistics III: Papers from the Third Annual Symposium on Arabic Linguistics* (pp. 155–172). Amsterdam: John Benjamins.

Beesley, K. (2001). Finite-state morphological analysis and generation of Arabic at Xerox research: Status and plans in 2001. In *Proceedings of the ACL-01 Workshop on Arabic Language Processing: Status and Prospects* (pp. 1–8), Toulouse, France.

Beesley, K. & Karttunen, L. (2003). *Finite State Morphology.* Stanford, CA: CSLI Publications.

Bentin, S. & Frost, R. (1995). Morphological factors in visual word identification in Hebrew. In Feldman L.B., (Ed.), Morphological aspects of language processing (pp. 271–292). Hillsdale, NJ: Erlbaum.

Buckwalter, T. (2002). *Buckwalter Arabic Morphological Analyzer Version 1.0.* Linguistic Data Consortium, Philadelphia. LDC catalog number LDC2002L49 and ISBN 1-58563-257-0.[14]

Cantineau, J. (1950a). La notion de 'schème' et son altération dans diverses langues sémitiques. In *Semitica, 3,* 73–83.

Cantineau, J. (1950b). Racines et schèmes. In *Mélanges offerts à William Marçais.* Paris : Maisonneuve.

Cassuto, P. (2000). Le classement dans les dictionnaires de l'hébreu. In Cassuto, P. & Larcher, P. (Eds.), *La sémitologie, aujourd'hui* (pp. 133–158).

Cassuto. P. & Larcher, P. (Eds.). (2000). *La sémitologie, aujourd'hui.* Travaux du Cercle linguistique d'Aix-en-Provence n°16, Publications de l'université de Provence:

Cohen, D. (1961). Essai d'une analyse automatique de l'arabe. *T.A. informations*. Reprod. in Cohen, D. *Études de linguistique sémitique et arabe* (pp. 49–78). The Hague/Paris: Mouton.

Desclés, J.-P., dir. (1983). (H. Abaab, J.-P. Desclés, J. Dichy, D.E. Kouloughli, M.S. Ziadah). *Conception d'un synthétiseur et d'un analyseur morphologiques de l'arabe, en vue d'une utilisation en Enseignement assisté par Ordinateur*. Rapport rédigé à la demande du Ministère des Affaires étrangères.

Diab, M. & Resnik, P. (2001). An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 255–262), Philadelphia, PA.

Dichy, J. (1984). Vers un modèle d'analyse automatique du mot graphique non-vocalisé en arabe. Presented at the Conference on "Communication entre langues européennes et

[14] Retrieved December 16, 2006, from http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L49

langues orientales", Montvillargenne, Oise. Revised version in Dichy, J. & Hassoun, M. (Eds.), (1989), pp. 92–158.

Dichy, J. (1987). The SAMIA Research Program, Year Four, Progress and Prospects. In *Processing Arabic Report 2* (pp. 1–26). T.C.M.O., Nijmegen University, Netherlands.

Dichy, J. (1990). *L'écriture dans la représentation de la langue : la lettre et le mot en arabe.* Doctorat d'État, Université Lumière Lyon 2, Lyon.

Dichy, J. (1993). Deux grands 'mythes scientifiques' relatifs au système d'écriture de l'arabe. In *Savoir, images, mirages,* Journées d'Études arabes, Special issue of *l'Arabisant* (pp. 32–33). Paris: Association Française des Arabisants.

Dichy, J. (1997). Pour une lexicomatique de l'arabe : l'unité lexicale simple et l'inventaire fini des spécificateurs du domaine du mot. *Meta 42,* 291–306. Presses de l'Université de Montréal.

Dichy, J. (2000). Morphosyntactic Specifiers to be associated to Arabic Lexical Entries - Methodological and Theoretical Aspects. In *Proceedings of ACIDA 2000* (Vol. 'Corpora and Natural Language Processing', pp. 55–60), Monastir, Tunisia.

Dichy, J. (2003). Sens des schèmes et sens des racines en arabe: le principe de figement lexical (PFL) et ses effets sur le lexique d'une langue sémitique. In Rémi-Giraud, S. & Panier, L., dir., *La polysémie ou l'empire des sens* (pp. 189–211). Lyon: Presses Universitaires de Lyon.

Dichy, J. (2005). Spécificateurs engendrés par les traits [±animé], [±humain], [±concret] et structures d'arguments en arabe et en français. In Béjoint, H. & Maniez, F. (Eds.), *De la mesure dans les termes*, Actes du colloque en hommage à Philippe Thoiron (pp. 151–181). Lyon: Presses Universitaires de Lyon.

Dichy, J. Braham, A., Ghazali, S. & Hassoun, M. (2002). La base de connaissances linguistiques DIINAR.1 (DIctionnaire INformatisé de l'ARabe, version 1). In Braham, A. (Ed.), *Proceedings of the International Symposium on the Processing of Arabic,* Université de la Manouba, Tunisia.

Dichy, J. & Farghaly, A. (2003). Roots and Patterns vs. Stems plus Grammar-Lexis Specifications: on what basis should a multilingual lexical database centred on Arabic be built? In Proceedings of the *IXth MT Summit Workshop on Machine Translation for Semitic Languages: Issues and Approaches* (pp. 1–8), New Orleans.

Dichy, J. & Hassoun, M. (Eds.) (1989). *Simulation de modèles linguistiques et Enseignement Assisté par Ordinateur de l'arabe – Travaux SAMIA I.* Paris: Conseil International de la Langue Française.

Dien, D., Kiem, H. & Hovy, E. (2003). BTL: a Hybrid Model for English-Vietnamese Machine Translation. In *Proceedings of the IXth MT Summit* (pp. 87–94), New Orleans.

Ditters, E. (1992). *A Formal Approach to Arabic Syntax: The Noun phrase and the Verb Phrase.* Ph.D. dissertation, Catholic University of Nijmegen, Netherlands.

Farghaly, A. (1987). *Three Level Morphology*. Paper presented at the Arabic Morphology Workshop, Linguistic Summer Institute, Stanford, CA.

Farghaly, A. (1994). Discontinuity in the Lexicon: A Case from Arabic Morphology. In *International Conference on Arabic Linguistics*, The American University in Cairo, Cairo, Egypt.

Fassi-Fehri, A. (1997). *Al-Maçjama wa-t-taxTīT – NaḎarāt jadīda fī qaDāyā l-luγa l-çarabiyya* [Lexicography and language planning. Arabic Language matters reconsidered, المعجمة والتخطيط – نظرات جديدة في قضايا اللغة العربية]. Casablanca, Morocco: Al-Markaz al-thaqāfiyy al-çarabiyy.

Forster, G., Grandrabur, S., Langlais, P., Plamondon, P., Russel, G. & Simard, M. (2003). Statistical Machine Translation: Rapid Development with limited Resources. In *Proceedings of the IXth MT Summit* (pp. 110–117), New Orleans.

Frost, R., Deutsch, A. & Forster, K.I. (2000). Decomposing morphologically complex words in a non linear morphology. *Journal of Experimental Psychology: Learning, Memory and Cognition, 26*, 751–65.

Frost, R., Forster, K.I. & Deutsch, A. (1997). What can we learn from the morphology of Hebrew? A masked priming investigation of morphological representation. *Journal of Experimental Psychology: Learning, Memory and Cognition, 23*, 829–856.

Geith, M. & El-Saadany, T. (1987). Arabic morphological analyzer on a personal computer. Presented at the Arabic Morphology Workshop, Linguistic Summer Institute, Stanford, CA.

Ghenima, M. (1998). *Analyse morpho-syntaxique en vue de la voyellation assistée par ordinateur des textes écrits en arabe*. Ph.D. dissertation, ENSSIB/Université Lyon 2.

Grainger, J., Dichy, J., El-Halfaoui, M. & Bamhamed, M. (2003). Approche expérimentale de la reconnaissance du mot écrit en arabe. In Jaffré, J.-P. (Ed.), *Dynamiques de l'écriture: approches pluridisciplinaires*. *Faits de langue*, 22, 77–86.

Hans Wehr. (1979) A dictionary of modern written Arabic. 4th edition, edited. by J. Milton Cowan. Wiesbaden, Harrassowitz.

Hassoun, M. (1987). *Conception d'un dictionnaire pour le traitement automatique de l'arabe dans différents contextes d'application.,* Ph.D. (thèse d'État), Université Lyon 1.

Hlal, Y. (1979). *Méthode d'apprentissage pour l'analyse morphosyntaxique (expérimentée dans le cas de l'arabe et du français).* Ph.D. dissertation, Université Paris-Sud, Centre d'Orsay.

Hlal, Y. (1985a). Morphology and syntax of the Arabic language. *Arab School of Sciences and Technology: Informatics* 4C, 1–8.

Hlal, Y. (1985b). Morphological analysis of Arabic speech. In *Workshop Papers Kuwait/Proceedings of Kuwait Conference on Computer Processing of the Arabic Language* (Section 13, pp. 273–294).

Karttunnen, L. (1994). Constructing Lexical Transducers. In *Proceedings of COLING-94*, (pp. 206–411), Tokyo, Japan.

Karttunnen, L. & Beesley, K.R. (2005). Twenty-five years of finite-state morphology. In Arppe, A., Carlson, L., Lindén, K., Piitulainen, J., Suominen, M., Vainio, M., Westerlund, H. & Yli-Jyrä, A. (Eds.), *Inquiries into Words, Constraints and Contexts. Festschrift for Kimmo Koskenniemi on his 60th Birthday (2005).* CSLI Studies in Computational Linguistics ONLINE, pp. 71–83. Copestake, A. (Series Ed.). Stanford, CA: CSLI Publications.

McCarthy, J. (1981). A Prosodic Theory of Nonconcatenative Morphology. *Linguistic Inquiry*, *12*, 373–418.

McCarthy, John J. & Prince, Alan S. (1996). Prosodic morphology. Technical report 32, Rutgers University Center for Cognitive science.

Melčuk, I. A. (1982). *Towards a Language of Linguistics: A System of Formal Notions for Theoretical Morphology.* München: Wilhem Fink Verlag.

Nikkhou, M. (Ed.) (2004). *NEMLAR International Conference on Arabic Language Resources and Tools,* Cairo. Paris: ELDA.

Ouersighni, R. (2001). A major offshoot of the DIINAR-MBC project: *AraParse*, a morpho-syntactic analyzer of unvowelled Arabic texts. In *ACL-01 Workshop on Arabic Language Processing: Status and Prospects* (pp. 66–72), Toulouse, France.

Ouersighni, R. (2002). *La conception et la réalisation d'un système d'analyse morpho-syntaxique robuste pour l'arabe: utilisation pour la détection et le diagnostic des fautes d'accord.* Ph.D. dissertation, ENSSIB/Université Lyon 2.

Rogati, M., McCarley, S. & Yang, Y. (2003). Unsupervised Learning of Arabic Stemming Using a Parallel Corpus. In *41st Annual Meeting of the Association of Computational Linguistics* (pp. 391–398), Sapporo, Japan.

Roman, A. (1990). *Grammaire de l'arabe.* Paris: P.U.F., coll. "Que sais-je?".

Roman, A. (1999). *La création lexicale en arabe, ressources et limites de la nomination dans une langue humaine naturelle.* Presses Universitaires de Lyon.

Rousseau, J. (1987). La découverte de la racine en sémitique par l'idéologue Volney. *Historiographia Linguistica, 14*(3), 341–365.

Sampson, G. (1985). *Writing systems.* Stanford University Press.

Schafer, C. & Yarowsky, D. (2003). A Two-Level Syntax-Based Approach to Arabic-English Statistical Machine Translation. In *Proceedings of the IXth MT Summit Workshop on Machine Translation for Semitic Languages: Issues and Approaches* (pp. 45–52), New Orleans.

Soudi, A., Cavalli-Sforza, V. & Jamari, A. (2001). A Computational Lexeme-Based Treatment of Arabic Morphology. In *ACL-01 Workshop on Arabic Language Processing: Status and Prospects* (pp. 155–162), Toulouse, France.

Troupeau, G. (1984). La notion de 'racine' chez les grammairiens arabes anciens. In Auroux, S., Glatiny, M., Joly, A., Nicolas, A. & Rosier, I. (Eds.), *Matériaux pour une histoire des théories linguistiques,* pp. 239–245. Presses Universitaires de Lille.

Zaafrani, R. (2002). *Développement d'un environnement interactif d'apprentissage avec ordinateur de l'arabe langue étrangère*. Ph.D. dissertation, ENSSIB/Université Lyon 2.

Zwiep, I. E. (1996). The Hebrew linguistic tradition of the Middle Ages. *Histoire Épistémologie Langage, 18*(1), 41–61.

PART III

Empirical Methods

8

Learning to Identify Semitic Roots

Ezra Daya[1], Dan Roth[2] and Shuly Wintner[3]

[1] *Department of Computer Science, University of Haifa and ClearForest Ltd.*
Ezra.Daya@ClearForest.com

[2] *Department of Computer Science, University of Illinois at Urbana-Champaign*
danr@cs.uiuc.edu

[3] *Department of Computer Science, University of Haifa*
shuly@cs.haifa.ac.il

Abstract: The morphology of Semitic languages is unique in the sense that the major word-formation mechanism is an inherently non-concatenative process of *interdigitation*, whereby two morphemes, a *root* and a *pattern*, are interwoven. Identifying the root of a given word in a Semitic language is an important task, in some cases a crucial part of morphological analysis. It is also a non-trivial task, which many humans find challenging. We present a machine learning approach to the problem of extracting roots of Semitic words. Given the large number of potential roots (thousands), we address the problem as one of combining several classifiers, each predicting the value of one of the root's consonants. We show that when these predictors are combined by enforcing some fairly simple linguistics constraints, high accuracy, which compares favorably with human performance on this task, can be achieved

8.1 Introduction

The standard account of word-formation processes in Semitic languages describes words as combinations of two morphemes: a *root* and a *pattern*.[1] The root consists of consonants only, by default three (although longer roots are known), called *radicals*. The pattern is a combination of vowels and, possibly, consonants too, with 'slots' into which the root consonants can be inserted. Words are created by *interdigitating* roots into patterns: the first radical is inserted into the first consonantal slot of the pattern, the second radical fills the second slot and the third fills the last slot. See Shimron (2003) for a survey of linguistic and psycho-linguistic issues in Semitic root-based morphology.

Identifying the root of a given word is an important task. Although existing morphological analyzers for Hebrew only provide a lexeme (which is a combination

[1] An additional morpheme, *vocalization*, is used to abstract the pattern further; for the present purposes, this distinction is irrelevant.

of a root and a pattern), for other Semitic languages, notably Arabic, the root is an essential part of any morphological analysis simply because traditional dictionaries are organized by root, rather than by lexeme. Furthermore, roots are known to carry some meaning, albeit vague. We believe that this information can be useful for computational applications.

We present a machine learning approach, augmented by limited linguistic knowledge, to the problem of identifying the roots of Semitic words. To the best of our knowledge, this is the first application of machine learning to this problem (but see also Marsi, van den Bosch, and Soudi, 2005; Habash and Rambow, 2005). While there exist programs that can extract the roots of words in Arabic (Beesley, 1998a; Beesley, 1998b) and Hebrew (Choueka, 1990), they are all dependent on labor-intensive construction of large-scale lexicons which are components of full-scale morphological analyzers. Note that Buckwalter (2002)'s Arabic morphological analyzer only uses "word stems – rather than root and pattern morphemes – to identify lexical items. (The information on root and pattern morphemes could be added to each stem entry if this were desired.)" The challenge of our work is to automate this process, avoiding the bottleneck of having to laboriously list the root and pattern of each lexeme in the language, and thereby gain insights that can be used for more detailed morphological analysis of Semitic languages.

As we show in Section 8.2, identifying roots is a non-trivial problem even for humans, due to the complex nature of Semitic derivational and inflectional morphology and the peculiarities of the orthography. From a machine learning perspective, this is an interesting test case of interactions among different yet interdependent classifiers. We focus on Hebrew in the first part of this chapter; after presenting the linguistic data in Section 8.3, we discuss a simple baseline learning approach (Section 8.4) and then propose two methods for combining the results of interdependent classifiers (Section 8.5), one which is purely statistical and one which incorporates linguistic constraints, demonstrating the improvement of the hybrid approach. Then, the same technique is applied to Arabic in Section 8.6 and we demonstrate comparable improvements. We conclude with suggestions for future research. An early version of this work was published as Daya, Roth, and Wintner (2004).

8.2 Linguistic Background

In this section we refer to Hebrew only, although much of the description is valid for other Semitic languages as well. As an example of root-and-pattern morphology, consider the Hebrew roots *g.d.l*, *k.t.b* and *r.$.m* and the patterns *haCCaCa, hitCaCCut* and *miCCaC*, where the 'C's indicate the slots. When the roots combine with these patterns the resulting lexemes are *hagdala, hitgadlut, migdal, haktaba, hitkatbut, miktab, har$ama, hitra$mut, mir$am*, respectively. After the root combines with the pattern, some morpho-phonological alternations take place, which may be non-trivial: for example, the *hitCaCCut* pattern triggers assimilation when the first consonant of the root is *t* or *d*: thus, *d.r.$+hitCaCCut* yields *hiddar$ut*. The same pattern triggers metathesis when the first radical is *s* or *$*: *s.d.r+hitCaCCut* yields *histadrut* rather than the expected **hitsadrut*. Semi-vowels such as *w* or *y* in the root are frequently combined with the vowels of the pattern, so that *q.w.m+haCCaCa* yields *haqama*, etc. Frequently, root consonants such as *w* or *y* are altogether missing from the resulting form.

These matters are complicated further due to two sources: first, the standard Hebrew orthography leaves most of the vowels unspecified. It does not explicate *a* and *e* vowels, does not distinguish between *o* and *u* vowels and leaves many of the *i* vowels unspecified. Furthermore, the single letter *w* is used both for the vowels *o* and *u* and for the consonant *v*, whereas *i* is similarly used both for the vowel *i* and for the consonant *y*. On top of that, the script dictates that many particles, including four of the most frequent prepositions, the definite article, the coordinating conjunction and some subordinating conjunctions all attach to the words which immediately follow them. Thus, a form such as *mhgr* can be read as a lexeme ("immigrant"), as *m-hgr* "from Hagar"or even as *m-h-gr* "from the foreigner". Note that there is no deterministic way to tell whether the first *m* of the form is part of the pattern, the root or a prefixing particle (the preposition *m* "from").

The Hebrew script has 22 letters, all of which can be considered consonants. The number of tri-consonantal roots is thus theoretically bounded by 22^3, although several phonological constraints limit this number to a much smaller value. For example, while roots whose second and third radicals are identical abound in Semitic languages, roots whose first and second radicals are identical are extremely rare (see McCarthy (1981) for a theoretical explanation). To estimate the number of roots in Hebrew we compiled a list of roots from two sources: a dictionary (Even-Shoshan, 1993) and the verb paradigm tables of Zdaqa (1974). The union of these yields a list of 2152 roots.[2]

While most Hebrew roots are regular, many belong to *weak paradigms*, which means that root consonants undergo changes in some patterns. Examples include *i* or *n* as the first root consonant, *w* or *i* as the second, *i* as the third and roots whose second and third consonants are identical. For example, consider the pattern *hCCCh*. Regular roots such as *p.s.q* yield forms such as *hpsqh*. However, the irregular roots *n.p.l, i.c.g, q.w.m* and *g.n.n* in this pattern yield the seemingly similar forms *hplh, hcgh, hqmh* and *hgnh*, respectively. Note that in the first and second examples, the first radical (*n* or *i*) is missing, in the third the second radical (*w*) is omitted and in the last example one of the two identical radicals is omitted. Consequently, a form such as hC_1C_2h can have any of the roots $n.C_1.C_2$, $C_1.w.C_2$, $C_1.i.C_2$, $C_1.C_2.C_2$ and even, in some cases, $i.C_1.C_2$.

While the Hebrew script is highly ambiguous, ambiguity is somewhat reduced for the task we consider here, as many of the possible lexemes of a given form share

[2] Only tri-consonantal roots are counted. Ornan (2003) mentions 3407 roots, whereas the number of roots in Arabic is estimated to be 10,000 (Darwish, 2002).

the same root. Still, in order to correctly identify the root of a given word, context must be taken into consideration. For example, the form *$mnh* has more than a dozen readings, including the adjective "fat" (feminine singular), which has the root *$.m.n*, and the verb "count", whose root is *m.n.i*, preceded by a subordinating conjunction. In the experiments we describe below we ignore context completely, so our results are handicapped by design.

8.3 Data and Methodology

We take a machine learning approach to the problem of determining the root of a given word. For training and testing, a Hebrew linguist manually tagged a corpus of 15,000 words (a set of newspaper articles). Of these, only 9752 were annotated with root information; the reason for the gap is that some Hebrew words, mainly borrowed but also some frequent words such as prepositions, do not have roots; we further eliminated 168 roots with more than three consonants and were left with 5242 annotated word types, exhibiting 1043 different roots. Table 8.1 shows the distribution of word types according to root ambiguity.

Table 8.2 provides the distribution of the roots of the 5242 word types in our corpus according to root type, where C_i is the i-th radical (note that some roots may belong to more than one group, so the total is greater than 100%).

As assurance for statistical reliability, in all the experiments discussed in the remainder of this chapter (unless otherwise mentioned) we performed 10-fold cross validation runs for every classification task during evaluation. We also divided the test corpus into two sets: a *development set* of 4800 words and a *held-out set* of 442 words. Only the development set was used for parameter tuning. A given *example* is a word type with all its (manually tagged) possible roots. In the experiments we describe below, our system produces one or more root *candidates* for each example. For each example, we define *tp* as the number of candidates correctly produced by the system; *fp* as the number of candidates which are not correct roots; and *fn* as the number of correct roots the system did not produce. As usual, we define *recall* as $\frac{tp}{tp+fp}$ and *precision* as $\frac{tp}{tp+fn}$; we then compute F-measure for each example (with $\beta = 1$) and (macro-) average to obtain the system's overall F-measure.

To estimate the difficulty of this task, we asked six human subjects to perform it. Subjects were asked to identify all the possible roots of all the words in a list of 200 words (without context), randomly chosen from the test corpus. All subjects were computer science graduates, native Hebrew speakers with no linguistic background. The average precision of humans on this task is 83.52%, and with recall at 80.27%,

Table 8.1. Root ambiguity in the corpus

Number of roots	1	2	3	4
Number of words	4886	335	18	3

Table 8.2. Distribution of root paradigms

Paradigm	Number	Percentage (%)
$C_1 = i$	414	7.90
$C_1 = w$	28	0.53
$C_1 = n$	419	7.99
$C_2 = i$	297	5.66
$C_2 = w$	517	9.86
$C_3 = h$	18	0.19
$C_3 = i$	677	12.92
$C_2 = C_3$	445	8.49
Regular	3061	58.41

F-measure is 81.86%. Two main reasons for the low performance of humans are the lack of context and the ambiguity of some of the weak paradigms.

8.4 A Machine Learning Approach

To establish a baseline, we first performed two experiments with simple baseline classifiers. In all the experiments described in this chapter we use SNoW (Roth, 1998) as the learning environment, with *winnow* as the update rule (using *perceptron* yielded comparable results). SNoW is a multi-class classifier that is specifically tailored for learning in domains in which the potential number of information sources (features) taking part in decisions is very large. It works by learning a sparse network of linear functions over a pre-defined or incrementally learned feature space. SNoW has already been used successfully as the learning vehicle in a large collection of natural language related tasks, including POS tagging, shallow parsing, information extraction tasks, etc., and compared favorably with other classifiers (Florian, 2002; Punyakanok and Roth, 2001; Roth, 1998). Typically, SNoW is used as a classifier, and predicts using a winner-take-all mechanism over the activation values of the target classes. However, in addition to the prediction, it provides a reliable confidence level in the prediction, which enables its use in an inference algorithm that combines predictors to produce a coherent inference.

8.4.1 Feature Types

All the experiments we describe in this work share the same features and differ only in the target classifiers. The features that are used to characterize a word are both grammatical and statistical:

- Location of letters (e.g., the third letter of the word is *b*). We limit word length to 20, thus obtaining 440 features of this type (recall the the size of the alphabet is 22).

- Bigrams of letters, independently of their location (e.g., the substring *gd* occurs in the word). This yields 484 features.
- Prefixes (e.g., the word is prefixed by *k$h* “when the”). We have 292 features of this type, corresponding to 17 prefixes and sequences thereof.
- Suffixes (e.g., the word ends with *im*, a plural suffix). There are 26 such features.

8.4.2 Direct Prediction

In the first of the two experiments, referred to as Experiment A, we trained a classifier to learn roots as a single unit. The two obvious drawbacks of this approach are the large set of targets and the sparseness of the training data. Of course, defining a multi-class classification task with 2152 targets, when only half of them are manifested in the training corpus, does not leave much hope for ever learning to identify the missing targets.

In Experiment A, the macro-average precision of ten-fold cross validation runs of this classification problem is 45.72%; recall is 44.37%, yielding an *F*-score of 45.03%. In order to demonstrate the inadequacy of this method, we repeated the same experiment with a different organization of the training data. We chose 30 roots and collected all their occurrences in the corpus into a test file. We then trained the classifier on the remainder of the corpus and tested on the test file. As expected, the accuracy was close to 0%.

8.4.3 Decoupling the Problem

In the second experiment, referred to as Experiment B, we separated the problem into three different tasks. We trained three classifiers to learn each of the root consonants in isolation and then combined the results straightforwardly, conjoining the decisions of the three classifiers. This is still a multi-class classification but the number of targets in every classification task is only 22 (the number of letters in the Hebrew alphabet) and data sparseness is no longer a problem. As we show below, each classifier achieves much better generalization, but the clear limitation of this method is that it completely ignores interdependencies between different targets: the decision on the first radical is completely independent of the decision on the second and the third.

We observed a difference between recognizing the first and third radicals and recognizing the second one, as can be seen in Table 8.3. These results correspond well to our linguistic intuitions: the most difficult cases for humans are those in which the second radical is *w* or *i*, and those where the second and the third consonants are identical. Combining the three classifiers using logical conjunction yields an *F*-measure of 52.84%. Here, repeating the same experiment with the organization of the corpus such that testing is done on unseen roots yielded 18.1% accuracy.

To demonstrate the difficulty of the problem, we conducted yet another experiment. Here, we trained the system as above but we tested it on different words whose roots were known to be in the training set. The results of experiment A here were

Table 8.3. Accuracy of SNoW identifying the correct radical

	C_1	C_2	C_3	root
Precision:	82.25	72.29	81.85	53.60
Recall:	80.13	70.00	80.51	52.09
F-measure:	81.17	71.13	81.18	52.84

46.35%, whereas experiment B was accurate in 57.66% of the cases. Evidently, even when testing only on previously seen roots, both naïve methods are unsuccessful.

8.5 Combining Interdependent Classifiers

Evidently, simply combining the results of the three classifiers leaves much room for improvement. Therefore we explore other ways for combining these results. We can rely on the fact that SNoW provides insight into the decisions of the classifiers – it lists not only the selected target, but rather all candidates, with an associated confidence measure. Apparently, the correct radical is chosen among SNoW's top-n candidates with reasonable accuracy, as the data in Table 8.3 reveal.

This observation calls for a different way of combining the results of the classifiers which takes into account not only the first candidate but also others, along with their confidence scores.

8.5.1 HMM Combination

We considered several ways, e.g., via HMMs, of appealing to the sequential nature of the task (C_1 followed by C_2, followed by C_3). Not surprisingly, direct applications of HMMs are too weak to provide satisfactory results, as suggested by the following discussion. The approach we eventually opted for combines the predictive power of a classifier to estimate more accurate state probabilities.

Given the sequential nature of the data and the fact that our classifier returns a distribution over the possible outcomes for each radical, a natural approach is to combine SNoW's outcomes via a Markovian approach. Variations of this approach are used in the context of several NLP problems, including POS tagging (Schütze and Singer, 1994), shallow parsing (Punyakanok and Roth, 2001) and named entity recognition (Tjong Kim Sang and De Meulder, 2003).

Formally, we assume that the confidence supplied by the classifier is the probability of a state (radical, c) given the observation o (the word), $P(c|o)$. This information can be used in the HMM framework by applying Bayes rule to compute

$$P(o|c) = \frac{P(c|o)P(o)}{P(c)},$$

where $P(o)$ and $P(c)$ are the probabilities of observing o and being at c, respectively. That is, instead of estimating the observation probability $P(o|c)$ directly from training data, we compute it from the classifiers' output. Omitting details (see Punyakanok and Roth, 2001), we can now combine the predictions of the classifiers by finding the most likely root for a given observation, as

$$r = argmaxP(c_1c_2c_3|o, \theta)$$

where θ is a Markov model that, in this case, can be easily learned from the supervised data. Clearly, given the short root and the relatively small number of values of c_i that are supported by the outcomes of SNoW, there is no need to use dynamic programming here and a direct computation is possible.

However, perhaps not surprisingly given the difficulty of the problem, this model turns out to be too simplistic. In fact, performance deteriorated. We conjecture that the static probabilities (the model) are too biased and cause the system to abandon good choices obtained from SNoW in favor of worse candidates whose global behavior is better.

For example, the root *&.b.d* was correctly generated by SNoW as the best candidate for the word *&obdim*, but since $P(C_3 = b|C_2 = b)$, which is 0.1, is higher than $P(C_3 = d|C_2 = b)$, which is 0.04, the root *&.b.b* was produced instead. Note that in the above example the root *&.b.b* cannot possibly be the correct root of *&obdim* since no pattern in Hebrew contains the letter *d*, which must therefore be part of the root. It is this kind of observations that motivates the addition of linguistic knowledge as a vehicle for combining the results of the classifiers. An alternative approach, which we intend to investigate in the future, is the introduction of higher-level classifiers which take into account interactions between the radicals (Punyakanok and Roth, 2001).

8.5.2 Adding Linguistic Constraints

The experiments discussed in Section 8.4 are completely devoid of linguistic knowledge. In particular, experiment B inherently assumes that any sequence of three consonants can be the root of a given word. This is obviously not the case: with very few exceptions, all radicals must be present in any inflected form (in fact, only *w*, *i*, *n* and in an exceptional case *l* can be deleted when roots combine with patterns). We therefore trained the classifiers to consider as targets only letters that occurred in the observed word, plus *w, i, n* and *l*, rather than any of the alphabet letters. The average number of targets is now 7.2 for the first radical, 5.7 for the second and 5.2 for the third (compared to 22 each in the previous setup).

In this model, known as the *sequential model* (Even-Zohar and Roth, 2001), the performance of SNoW improved slightly, as can be seen in Table 8.4 (compare to Table 8.3). Combining the results in the straightforward way yields an F-measure of 58.89%, a small improvement over the 52.84% performance of the basic method. This new result should be considered baseline. In what follows we always employ the sequential model for training and testing the classifiers, using the same constraints. However, we employ more linguistic knowledge for a more sophisticated combination of the classifiers.

Table 8.4. Accuracy of SNoW's identifying the correct radical, sequential model

	C_1	C_2	C_3	root
Precision:	83.06	72.52	83.88	59.83
Recall:	80.88	70.20	82.50	57.98
F-measure:	81.96	71.34	83.18	58.89

8.5.3 Combining Classifiers Using Linguistic Knowledge

SNoW provides a ranking on all possible roots. We now describe the use of linguistic constraints to re-rank this list. We implemented a function which uses knowledge pertaining to word-formation processes in Hebrew in order to estimate the likelihood of a given candidate being the root of a given word. The function practically classifies the candidate roots into one of three classes: good candidates, which are likely to be the root of the word; bad candidates, which are highly unlikely; and average cases.

The decision of the function is based on the observation that when a root is regular it either occurs in a word consecutively or with a single *w* or *i* between any two of its radicals. The scoring function checks, given a root and a word, whether this is the case. Furthermore, the suffix of the word, after matching the root, must be a valid Hebrew suffix (there is only a small number of such suffixes in Hebrew). If both conditions hold, the scoring function returns a high value. Then, the function checks if the root is an unlikely candidate for the given word. For example, if the root is regular its consonants must occur in the word in the same order they occur in the root. If this is not the case, the function returns a low value. We also make use in this function of our pre-compiled list of roots. A root candidate which does not occur in the list is assigned the low score. In all other cases, a middle value is returned.

The actual values that the function returns were chosen empirically by counting the number of occurrences of each class in the training data. For example, "good" candidates make up 74.26% of the data, hence the value the function returns for "good" roots is set to 0.7426. Similarly, the middle value is set to 0.2416 and the low value to 0.0155.

As an example, consider *hipltm*, whose root is *n.p.l* (note that the first *n* is missing in this form). Here, the correct candidate will be assigned the middle score while *p.l.t* and *l.t.m* will score high.

In addition to the scoring function we implemented a simple edit distance function which returns, for a given root and a given word, the inverse of the edit distance between the two. For example, for *hipltm*, the (correct) root *n.p.l* scores $1/4$ whereas *p.l.t* scores $1/3$. When the edit distance is 0, an empirically chosen high value is used.

We then run SNoW on the test data and rank the results of the three classifiers *globally*, where the order is determined by the product of the three different classifiers. This induces an order on *roots*, which are combinations of the decisions of three independent classifiers. Each candidate root is assigned three scores: the product of the confidence measures of the three classifiers; the result of the scoring function;

Table 8.5. Performance of the system when producing top-i candidates

$i =$	1	2	3	4
Precision	**82.02**	46.17	32.81	25.19
Recall	**79.10**	87.83	92.93	94.91
F-measure	**80.53**	60.52	48.50	39.81

and the inverse edit distance between the candidate and the observed word. We rank the candidates according to the product of the three scores (i.e., we give each score an equal weight in the final ranking).

In order to determine which of the candidates to produce for each example, we experimented with two methods. First, the system produced the top-i candidates for a fixed value of i. The results on the development set are given in Table 8.5.

Obviously, since most words have only one root, precision drops dramatically when the system produces more than one candidate. This calls for a better threshold, facilitating a non-fixed number of outputs for each example. We observed that in the "difficult" examples, the top ranking candidates are assigned close scores, whereas in the easier cases, the top candidate is usually scored much higher than the next one. We therefore decided to produce all those candidates whose scores are not much lower than the score of the top ranking candidate. The drop in the score, δ, was determined empirically on the development set. The results are listed in Table 8.6, where δ varies from 0.1 to 0.8 (δ is actually computed on the log of the actual score, to avoid underflow).

These results show that choosing $\delta = 0.4$ produces the highest F-measure. With this value for δ, results for the held-out data are presented in Table 8.7. The results clearly demonstrate the added benefit of the linguistic knowledge. In fact, our results are slightly better than average human performance (repeated here for convenience). Interestingly, even when testing the system on a set of roots which do *not* occur in the training corpus (see Section 8.4), we obtain an F-score of 65.60%. This result demonstrates the robustness of our method.

It must be noted that the scoring function alone is *not* a function for extracting roots from Hebrew words. First, it only scores a given root candidate against a given word, rather than yield a root given a word. While we could have used it exhaustively on all possible roots in this case, in a general setting of a number of classifiers the number of classes might be too high for this solution to be practical. Second, the function only produces three different values; when given a number of candidate

Table 8.6. Performance of the system, producing candidates scoring no more than δ below the top score

$\delta =$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Precision	81.81	80.97	79.93	**78.86**	77.31	75.48	73.71	71.80
Recall	81.06	82.74	84.03	**85.52**	86.49	87.61	88.72	89.70
F-measure	81.43	81.85	81.93	**82.06**	81.64	81.10	80.52	79.76

Table 8.7. Results: performance of the system on held-out data

	Held-out data	Humans
Precision:	**80.90**	83.52
Recall:	**88.16**	80.27
F-measure:	**84.38**	81.86

roots it may return more than one root with the highest score. In the extreme case, when called with all 22^3 potential roots, it returns on the average more than 11 candidates which score highest (and hence are ranked equally).

Similarly, the additional linguistic knowledge is not merely *eliminating* illegitimate roots from the ranking produced by SNoW. Using the linguistic constraints encoded in the scoring function only to eliminate roots, while maintaining the ranking proposed by SNoW, yields much lower accuracy. Clearly, our linguistically motivated scoring does more than elimination, and actually *re-ranks* the roots. It is only the *combination* of the classifiers with the linguistically motivated scoring function which boosts the performance on this task.

8.5.4 Error Analysis

Looking at the questionnaires filled in by our subjects (Section 8.3), it is obvious that humans have problems identifying the correct roots in two general cases: when the root paradigm is weak (i.e., when the root is irregular) and when the word can be read in more than one way and the subject chooses only one (presumably, the most prominent one). Our system suffers from similar problems: first, its performance on the regular paradigms is far superior to its overall performance; second, it sometimes cannot distinguish between several roots which are in principle possible, but only one of which happens to be the correct one.

To demonstrate the first point, we evaluated the performance of the system on a different organization of the data. We tested separately words whose roots are all regular, vs. words all of whose roots are irregular. We also tested words which have at least one regular root (mixed). The results are presented in Table 8.8, and clearly demonstrate the difficulty of the system on the weak paradigms, compared to almost 95% on the easier, regular roots.

Table 8.8. Error analysis: performance of the system on different cases

	Regular	Irregular	Mixed
Number of words	2598	2019	2781
Precision:	92.79	60.02	92.54
Recall:	96.92	73.45	94.28
F-measure:	94.81	66.06	93.40

Table 8.9. Error analysis: the weak paradigms

Paradigm	*F*-measure
$C_1 = i$	70.57
$C_1 = n$	71.97
$C_2 = i/w$	76.33
$C_3 = i$	58.00
$C_2 = C_3$	47.42

A more refined analysis reveals differences between the various weak paradigms. Table 8.9 lists *F*-measure for words of which the roots are irregular, classified by paradigm. As can be seen, the system has great difficulty in the cases of $C_2 = C_3$ and $C_3 = i$.

Finally, we took a closer look at some of the errors, and in particular at cases where the system produces several roots where fewer (usually only one) are correct. Such cases include, for example, the word *hkwtrt* ("the title"), of which the root is the regular *k.t.r*; but the system produces, in addition, also *w.t.r*, mistaking the *k* to be a prefix. Errors of this kind are most difficult to fix.

However, in many cases the system's errors are relatively easy to overcome. Consider, for example, the word *hmtndbim* ("the volunteers") whose root is the irregular *n.d.b*. Our system produces as many as five possible roots for this word: *n.d.b, i.t.d, d.w.b, i.h.d, i.d.d*. Clearly some of these could be eliminated. For example, *i.t.d* should not be produced, because if this were the root, nothing could explain the presence of the *b* in the word; *i.h.d* should be excluded because of the location of the *h*. Similar phenomena abound in the errors the system makes; they indicate that a more careful design of the scoring function can yield still better results, and this is the direction we intend to pursue in the future.

8.6 Extension to Arabic

Although Arabic and Hebrew have a very similar morphological system, being both semitic languages, the task of learning roots in Arabic is more difficult than in Hebrew, for the following reasons:

- There are 28 letters in Arabic which are represented using approximately 40 characters in Buckwalter (2002)'s transliteration of Modern Standard Arabic orthography. Thus, the learning problem is more complicated due to the increased number of targets (potential root radicals) as well as the number of characters available in a word.
- The number of roots in Arabic is significantly higher. We pre-compiled a list of 3822 three-letter roots from Buckwalter's list of roots, 2517 of which occur in our corpus. According to our lists, Arabic has almost twice as many roots as Hebrew.

Table 8.10. Arabic root ambiguity in the corpus

Number of roots	1	2	3	4	5	6
Number of words	28741	2258	664	277	48	3

- Not only is the number of roots high, the number of patterns in Arabic is also much higher than in Hebrew.
- While in Hebrew the only possible letters which can intervene between root radicals in a word are *i* and *w*, in Arabic there are more possibilities. The possible intervening letter sequences between C_1 and C_2 are *y, w, A, t* and *wA*, and between C_2 and C_3 *y, w, A* and *Aŷ*.

We applied the same methods discussed above to the problem of learning (Modern Standard) Arabic roots. For training and testing, we produced a corpus of 31991 word types (we used Buckwalter (2002)'s morphological analyzer to analyze a corpus of 152666 word tokens from which our annotated corpus was produced). Table 8.10 shows the distribution of word types according to root ambiguity.

We then trained naïve classifiers to identify each radical of the root in isolation, using features of the same categories as for Hebrew. Despite the rather pessimistic starting point, each classifier provides satisfying results, as shown in Table 8.11, probably owing to the significantly larger training corpus. The first three columns present the results of each of the three classifiers, and the fourth column is a straightforward combination of the the three classifiers.

The classifiers were combined using linguistic knowledge pertaining to word-formation processes in Arabic, by implementing a function that estimates the likelihood of a given candidate to be the root of a given word. The function actually checks the following cases:

- If a root candidate is indeed the root of a given word, then we expect it to occur in the word consecutively or with one of {*y, w, A, t, wA*} intervening between C_1 and C_2, or with one of { *y, w, A, Aŷ* } between C_2 and C_3 (or both).
- If a root candidate does not occur in our pre-compiled list of roots, it cannot be a root of any word in the corpus.

Of course, this is an over-simplistic account of the linguistic facts, but it serves our purpose of using very limited and very shallow linguistic constraints on the combination of specialized "expert" classifiers. Table 8.12 shows the final results.

Table 8.11. Accuracy of SNoW's identifying the correct radical in Arabic

	C_1	C_2	C_3	root
Precision:	86.02	70.71	82.95	54.08
Recall:	89.84	80.29	88.99	68.10
F-measure:	87.89	75.20	85.86	60.29

Table 8.12. Results: Arabic root identification

Precision:	78.21%
Recall:	82.80%
F-measure:	80.44%

The Arabic results are slightly worse than the Hebrew ones. One reason is that in Hebrew the number of roots is smaller than in Arabic (2152 vs. 3822), which leaves much room for incorrect root selection. Another reason can be the fact that in Arabic word formation is a more complicated process, for example by allowing more characters to occur in the word between the root letters as previously mentioned. This may have caused the scoring function to wrongly tag some root candidates as possible roots.

8.7 Conclusions

We have shown that combining machine learning with limited linguistic knowledge can produce state-of-the-art results on a difficult morphological task, the identification of roots of Semitic words. Our best result, over 80% precision for both Hebrew and Arabic, was obtained using simple classifiers for each of the root's consonants, and then combining the outputs of the classifiers using a linguistically motivated, yet extremely coarse and simplistic, scoring function. This result is comparable to average human performance on this task.

This work can be improved in a variety of ways. We intend to spend more effort on feature engineering. As is well-known from other learning tasks, fine-tuning of the feature set can produce additional accuracy; we expect this to be the case in this task, too. In particular, introducing features that capture contextual information is likely to improve the results. Similarly, our scoring function is simplistic and we believe that it can be improved. We also intend to improve the edit-distance function such that the cost of replacing characters reflect phonological and orthographic constraints (Kruskal, 1999).

In another track, there are various other ways in which different inter-related classifiers can be combined. Here we only used a simple multiplication of the three classifiers' confidence measures, which is then combined with the linguistically motivated functions. We intend to investigate more sophisticated methods for this combination, including higher-order machine learning techniques.

Finally, we plan to extend these results to more complex cases of learning tasks with a large number of targets, in particular such tasks in which the targets are structured. We are currently working on morphological disambiguation in languages with non-trivial morphology, which can be viewed as a POS tagging problem with a large number of tags on which structure can be imposed using the various morphological and morpho-syntactic features that morphological analyzers produce. We intend to investigate this problem for Hebrew and Arabic in the near future.

Acknowledgments

This work was supported by The Caesarea Edmond Benjamin de Rothschild Foundation Institute for Interdisciplinary Applications of Computer Science. Dan Roth is supported by NSF grants CAREER IIS-9984168, ITR IIS-0085836, and ITR-IIS 00-85980. We thank Meira Hess and Liron Ashkenazi for annotating the corpus and Alon Lavie and Ido Dagan for useful comments.

References

Beesley, Kenneth R. 1998a. Arabic morphological analysis on the internet. In *Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing*, Cambridge, April.

Beesley, Kenneth R. 1998b. Arabic morphology using only finite-state operations. In Michael Rosner, editor, *Proceedings of the Workshop on Computational Approaches to Semitic languages*, pages 50–57, Montreal, Quebec, August. COLING-ACL'98.

Buckwalter, Tim. 2002. Buckwalter Arabic morphological analyzer. Linguistic Data Consortium (LDC) catalog number LDC2002L49 and ISBN 1-58563-257-0.

Choueka, Yaacov. 1990. MLIM - a system for full, exact, on-line grammatical analysis of Modern Hebrew. In Yehuda Eizenberg, editor, *Proceedings of the Annual Conference on Computers in Education*, p. 63, Tel Aviv, April. In Hebrew.

Darwish, Kareem. 2002. Building a shallow Arabic morphological analyzer in one day. In Mike Rosner and Shuly Wintner, editors, *Computational Approaches to Semitic Languages, an ACL'02 Workshop*, pp. 47–54, Philadelphia, PA, July.

Daya, Ezra, Dan Roth, and Shuly Wintner. 2004. Learning Hebrew roots: Machine learning with linguistic constraints. In *Proceedings of EMNLP'04*, pp. 357–364, Barcelona, Spain, July.

Even-Shoshan, Abraham. 1993. *HaMillon HaXadash (The New Dictionary)*. Kiryat Sefer, Jerusalem. In Hebrew.

Even-Zohar, Y. and Dan Roth. 2001. A sequential model for multi class classification. In *EMNLP-2001, the SIGDAT Conference on Empirical Methods in Natural Language Processing*, pp. 10–19.

Florian, Radu. 2002. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pp. 175–178. Taiwan.

Habash, Nizar and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 573–580, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Kruskal, Joseph. 1999. An overview of sequence comparison. In David Sankoff and Joseph Kruskal, editors, *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. CSLI Publications, Stanford, CA, pp. 1–44. Reprint, with a foreword by John Nerbonne.

Marsi, Erwin, Antal van den Bosch, and Abdelhadi Soudi. 2005. Memory-based morphological analysis generation and part-of-speech tagging of Arabic. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pp. 1–8, Ann Arbor, Michigan, June. Association for Computational Linguistics.

McCarthy, John J. 1981. A prosodic theory of nonconcatenative morphology. *Linguistic Inquiry*, 12(3):373–418.

Ornan, Uzzi. 2003. *The Final Word.* Haifa, Israel: University of Haifa Press. In Hebrew.

Punyakanok, Vasin and Dan Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems 13*, pp. 995–1001. MIT Press.

Roth, Dan. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of AAAI-98 and IAAI-98*, pp. 806–813, Madison, Wisconsin.

Schütze, H. and Y. Singer. 1994. Part-of-speech tagging using a variable memory Markov model. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 181–187.

Shimron, Joseph, editor. 2003. *Language Processing and Acquisition in Languages of Semitic, Root-Based, Morphology.* Number 28 in Language Acquisition and Language Disorders. John Benjamins.

Tjong Kim Sang, Erik F. and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pp. 142–147. Edmonton, Canada.

Zdaqa, Yizxaq. 1974. *Luxot HaPoal (The Verb Tables).* Jerusalem: Kiryath Sepher. In Hebrew.

9

Automatic Processing of Modern Standard Arabic Text

Mona Diab[1], Kadri Hacioglu[2] and Daniel Jurafsky[3]

[1] *Center for Computational Learning Systems, Columbia University, mdiab@cs.columbia.edu*

[2] *Center for Spoken Language Research, University of Colorado, Boulder, hacioglu@colorado.edu*

[3] *Linguistics Department, Stanford University, jurafsky@stanford.edu*

Abstract: To date, there are no fully automated systems addressing the community's need for fundamental language processing tools for Arabic text. In this chapter, we present a Support Vector Machine (SVM) based approach to automatically tokenize (segmenting off clitics), part-of- speech (POS) tag and annotate Base Phrase Chunks (BPC) in Modern Standard Arabic (MSA) text. We adapt highly accurate tools that have been developed for English text and apply them to Arabic text. Using standard evaluation metrics, we report that the (SVM-TOK) tokenizer achieves an $F_{\beta=1}$ score of 99.1, the (SVM-POS) tagger achieves an accuracy of 96.6%, and the (SVM-BPC) chunker yields an $F_{\beta=1}$ score of 91.6

9.1 Introduction

The Arabic language is receiving growing attention in the NLP community, due both to its socio-political importance and to the NLP challenges presented by its dialect differences, diglossia, complex morphology, and non-transparent orthography. But like most languages, Arabic is lacking in annotated resources and tools. Fully automated fundamental NLP tools such as tokenizers, part of speech taggers, parsers, and semantic role labelers are still not available for Arabic.

In this chapter, we propose solutions for Modern Standard Arabic (MSA) on a crucial subset of these NLP tasks: clitic tokenization and basic lemmatization, part of speech tagging and base phrase chunking. Our algorithms draw heavily on the Arabic Tree Bank (**ATB**) (Maamouri et al., 2004).

We use the word **tokenization,** or **clitic tokenization** to refer to the process of segmenting **clitics** from stems. In Arabic, prepositions, conjunctions, and some pronouns are cliticized (orthographically and phonologically fused) onto stems. Separating conjunctions from the following noun, for example, is a key first step in parsing. There are many ways to define the tokenization problem, depending on exactly which morphemes are segmented off (Habash & Sadat, 2006). In this chapter we define the input to tokenization as space-delimited surface forms of words, and we define the output of the tokenizer as a sequence of clitics and **derivational lemma** forms.

A **lemma** is a citation form that exists as a dictionary entry. True lemmas require full morphological parsing, because a lemma would need to have inflectional morphology (such as plural markers) segmented off. Instead of producing lemmas, we produce **derivational lemmas**, which may have inflections, but which look more like lemmas than like surface wordform stems in other ways. When lemmas appear in context, some of the letters at the morpheme boundaries change due to **morphophonological rules**. Our derivational lemmas reverse these morphophonological rules, resulting in forms that look more like dictionary entries, although they have inflectional endings.

We use the term **Part of Speech** (POS) tagging to refer to the process of annotating these segmented words with parts of speech drawn from the ATB Arabic reduced part of speech tagset (Maamouri et al., 2004).

Finally, **Base Phrase Chunking** (BPC) is the process of annotating the input sequence of tokens with non-recursive base phrases such as noun phrases, adjectival phrases, verb phrases, preposition phrases, and so on.

For each of these tasks, we adopt a unified supervised machine learning perspective using Support Vector Machines (SVMs) trained on the ATB. We leverage off of already existing algorithms for English. The results are comparable to state-of-the-art results on English text when trained on similar sized data.

The remainder of this chapter is organized as follows: Section 9.2 sketches relevant aspects of the Arabic language and the ATB. Section 9.3 briefly discusses previous approaches to our tasks; Section 9.4 describes our approach; Sections 9.5 through 9.8 describe our four classifiers (tokenizer, lemmatizer, part-of-speech tagger, and base-phrase chunker) and their performance; we then conclude in Section 9.9 with some observations about the tasks and future directions.

9.2 The Arabic Language

Arabic is a Semitic language with rich templatic morphology. Templatic morphology refers to a type of morphology that is based on having roots and patterns (templates) to identify derivational lemmas in the language. An Arabic word may be composed of a stem (consisting of a consonantal root and a template/pattern that encodes mainly the vocalic distribution of a word in addition to other systematic information), plus affixes and clitics. The affixes include inflectional markers for tense, gender, and/or number. The clitics include some (but not all) prepositions, conjunctions, determiners, possessive and object pronouns. Some clitics are proclitics (attaching to the beginning of a stem) and some enclitics (attaching to the end of a stem). The following is an example of the different morphological segments in the word وبحسناتهم, read in transliteration as ***wbHsnAthm***,[1] which means *and by their virtues*. Arabic is read from right to left hence the directional switch in the English gloss as follows in Table 9.1.

The set of possible proclitics comprises the prepositions (*b, l, k*), meaning (*by/with, to, as*), respectively; the conjunctions (*w, f*), meaning (*and, then*), respectively; and the definite article or determiner (*Al*), meaning (*the*). Arabic words may have a conjunction and a preposition and a determiner cliticizing to the beginning of a word. The set of possible enclitics comprises the object and possessive pronouns. The set contains some overlaps as illustrated in Table 9.2.

The **tokens** of a word as defined in this chapter are the (optional) proclitics, the stem (together with its inflectional affixes), the (optional) enclitic, and the (optional) punctuation mark.

As suggested earlier, the tokenization process in this chapter does not correspond to complete morphological segmentation, because inflectional affixes are not segmented from the stem. Arabic has an intricate inflectional system. Verbs are marked for mood, tense, aspect, number, gender and voice. Nouns and adjectives are marked for number, gender, case and definiteness. Adjectives are marked

Table 9.1. Some morphemes in an MSA word

	Enclitic	Affix	Stem	Proclitic	Proclitic
MSA	هم	ات	حسن	ب	و
Transliteration	**Hm**	**At**	**Hsn**	**b**	**W**
Gloss	**their**	**S**	**virtue**	**by**	**And**

Table 9.2. List of possible MSA pronoun enclitics

MSA	Translit.	Number	Gender	Object	Possessive
ي	y	sing.	masc./fem.		my/mine
ني	ny	sing.	masc./fem.	me	
نا	nA	plur.	masc./fem.	ours	our
ك	k	sing.	masc./fem.	you	yours
كما	kmA	dual	masc./fem.	you	yours
كن	kn	plur.	fem.	you	yours
كم	km	plur.	masc./fem.	you	yours
ه	h	sing.	masc.	him/it	his/its
ها	hA	sing.	fem.	her/it	hers/its
هما	hmA	dual	masc./fem	their	theirs
هم	hm	plur.	masc.	their	theirs
هن	hn	plur.	fem.	their	theirs

[1] All transliteration in this chapter is rendered in the transliteration scheme described in Chapter 2 of this book.

for number and gender. Some of these features, like case (nominative, accusative, genitive, etc.) are determined contextually from the syntactic configuration in which a word appears. Since we are not addressing syntactic parsing in this chapter, we will not address the full processing of inflectional morphology. However, in the POS tagging task, nouns are marked with number information and verbs are marked with some tense and voice information as encoded in the tag set.

9.3 Related Work

To our knowledge, there exist no unified systems that deal with raw Arabic text processing from tokenization to base phrase chunking. However, various systems perform subsets of these tasks.

The current standard approach to Arabic tokenization and POS tagging — adopted in the ATB — relies on manually choosing the appropriate analysis from among the multiple analyses rendered by the Buckwalter Arabic Morphological Analyzer (BAMA).[2] BAMA is a sophisticated finite state rule based morphological analyzer. Morphological analysis may be characterized as the process of segmenting a surface word form into its component derivational and inflectional morphemes. In a language such as Arabic, which exhibits both inflectional and derivational morphology, the morphological tags tend to be fine grained amounting to a large number of tags — BAMA has 139 distinct morphological labels — in contrast to POS tags which are typically coarser grained. Using BAMA, the choice of an appropriate morphological analysis entails clitic tokenization as well assignment of a POS tag. It is worth noting that BAMA is not a disambiguator, it only renders analyses for each word in the text.

For tokenization, the two most recent specialized systems are those of Lee et al. (Lee et al., 2003) and Darwish (Darwish, 2002). Darwish proposes a rule-based system for tokenizing Arabic text into a sequence of prefix stem/root suffix. His system presupposes a single prefix. It reduces the stem to the root form based on a manually created list of word root pairs. The approach exploits look-up tables for a list of prefixes and suffixes. His approach yields an accuracy of 92.7%.

On the other hand, Lee et al. (Lee et al., 2003), propose a partially supervised n-gram language model approach for Arabic segmentation. Their segmentation aims at maximizing the correspondence between English and Arabic tokens for the purpose of statistical machine translation. Their approach relies on look-up tables for stems, prefixes and suffixes. Their notion of a prefix includes both clitics such as conjunctions and prepositions, and affixal inflectional morphemes such as the first person ***A*** in ***Akrr*** *(I repeat)*. Also for suffixes, they consider the plural marker ***At*** on feminine nouns and adjectives, as well as ***wn*** and ***yn*** on masculine nouns and adjectives as suffixes that are segmented out. In their approach, they start off with a supervised segmenter trained on tables derived from the manually

[2] http://www.ldc.upenn.edu/myl/morph/buckwalter.html

created ATB1 (101k tokens). Then they use this basic supervised segmenter to acquire more data in an unsupervised bootstrapping method from a large unannotated corpus. Their approach yields a segmentation accuracy of 97%. The approaches of Lee and of Darwish are not consistent with the ATB style tokenization since they segment off inflectional morphemes.

Most recently, for morphological disambiguation, Habash & Rambow (2005) propose a system MADA that relies on the output of BAMA to render the appropriate full morphological features for all words in MSA text. MADA learns the different 10 features, namely: basic POS tag (15 tags), presence of a conjunction, presence of a particle, presence of a pronoun, presence of a determiner, gender, number, person, voice and aspect. The features are learned independently using SVM-based learning. Then, MADA disambiguates choosing the most fitting set of feature values from the space of possible features using a decision tree algorithm. MADA achieves a morphological disambiguation accuracy of 95.6%.

Khoja (2001) reports preliminary results on a hybrid, statistical and rule based, POS tagger/morphological disambiguator, APT. APT yields 90% accuracy on a tag set of 131 tags. The tag set is more akin to the morphological tags used in BAMA. APT is a two-step hybrid system with rules and a Viterbi algorithm for statistically determining the appropriate tag. The first step is a dictionary look-up which assigns a word all possible tags listed in the dictionary. The second step is a stochastic decision process where an appropriate tag is chosen from the possible tags assigned.

Though they perform the same task, MADA and APT are not directly comparable since they use different tag sets. However they are similar in that they both render detailed tag sets that account for both inflectional and derivational morphology.

9.4 Our SVM-based Approach

Various machine learning approaches have been applied to part-of-speech tagging and base-phrase chunking, by casting them as classification tasks. Given a set of features extracted from the linguistic context, a classifier predicts the part-of-speech or base-phrase-chunk class of a token. Support Vector Machines (SVMs) (Vapnik, 1995) are one such supervised machine learning algorithm, with the advantages of discriminative training, robustness and a capability to handle a large number of (overlapping) features with good generalization performance. Consequently, SVMs have been applied in many NLP tasks with great success: text categorization (Joachims, 1998); syntactic chunking and shallow parsing (Kudo & Matsumoto, 2000); semantic role labeling (Hacioglu & Ward, 2003). We take this perspective a step further from previous studies in that we also cast the tokenization problem as a classification task. We adopt a unified tagging perspective, using the same SVM experimental setup which comprises a standard SVM as a multi-class classifier (Allwein et al., 2000). The difference for the tasks lies in the input, context and features. The features utilized in our approach are not explicitly language dependent except for the lemmatization features in the tokenization step. The following subsections illustrate the different tasks and their corresponding features and tag sets. We use a sequence model on the SVMs to take advantage of the context of the items being compared in a vertical manner in addition to the encoded features in the horizontal input of the vectors. Accordingly, in our different tasks, we define the notion of context to be a window of fixed size around the segment in focus for training and disambiguation.

9.4.1 Data

Our SVM classifiers are supervised, hence the need for annotated training data. We use three corpora from the ATB: ATB1 version 3, ATB2 version 2, and ATB3 version 2. They comprise articles from different newswires: Agence France Presse newswire; Al-Hayat newspaper distributed by Ummah; and An Nahar news agency. The articles mainly consist of political, economic and sports news texts. ATB1 comprises 734 news articles (140k words). ATB2 comprises 501 stories from the Ummah Arabic newswire (140k words). ATB3 consists of 600 stories (340k words). Therefore, the total number of words for ATB1, ATB2, ATB3 is 750k words after clitic segmentation, corresponding to 1900 news articles.

We performed various corpus cleanups including removing spurious characters and consistency-checking of the syntactic trees using David Chiang's *cat-tree.* (personal communication)

We formatted the corpus such that each line corresponds to a single tree. We also created a standard split for the corpus which could be useful for the community at large for other NLP or linguistic tasks. Each of the three ATB corpora is split into ~10% development data, ~80% training data and ~10% test data, maintaining document/article boundaries.[3] The development and training data are randomized on the document level. The respective splits for the different ATB corpora are concatenated; thus the development set for ATB1 is concatenated to the development set for ATB2 and ATB3, etc. Table 9.3 shows the size of the resulting splits.

The ATB data is distributed in a Latin-based ASCII representation using the Buckwalter transliteration scheme.[4] We use the unvocalized version of the ATB for all the experiments. All the data is derived from the syntactic trees in the ATB.

Table 9.3. Size of the data splits used in our experiments

Data	Tokens	Sentences
Development	70188	2304
Training	594683	18970
Test	69665	2337

[3] Data may be obtained from first author upon request.

[4] http://www.ldc.upenn.edu/myl/morph/buckwalter.html

9.4.2 SVM Setup

We use a standard SVM with a polynomial kernel, of degree 2 and C=1.[5] These parameters were the best yielding setting as determined on the development data.

9.4.3 Evaluation Metric

Standard metrics of accuracy (Acc), precision (Prec), recall (Rec), and F-score ($F_{ß=1}$), are reported on the test data for all our evaluations. We use the CoNLL standard shared task evaluation tools for all tasks.[6]

9.5 Clitic Tokenization

We approach clitic tokenization as a one-of-nine classification task, in which each letter in a word is tagged with a label indicating its morphological identity. For the purposes of this chapter, we do not tokenize the proclitic determiner ***Al*** *(the)* since it is not tokenized separately in the ATB. Therefore, a word may have *p* ($0 \leq p \leq 2$) proclitics and *e* ($0 \leq e \leq 1$) enclitics from the lists in Section 9.2. We model the task as a chunking problem, where each word is divided into a maximum of 4 chunks: a possible proclitic prefix1, followed by a possible proclitic prefix2, then the stem followed by a possible enclitic suffix.[7] We adopt an inside-outside-beginning (IOB) of a chunk approach (Ramshaw & Marcus, 1995). We break the surface words into, potentially, four regions. The first letter in each delimited chunk is labeled with a begin chunk marker B. The following letters in a designated chunk are labeled with an I marker indicating that they are inside a chunk. The O marker indicates spaces between surface form words. Table 9.4 illustrates an example of the input word described in Table 9.1 in the IOB representation.

Input: A sequence of transliterated Arabic letters processed from left-to-right with break markers for word boundaries - space delimiters.

Context: A fixed-size window of –5/+5 characters centered at the character in focus.

Features: All characters and previous tag decisions within the context.

Tag Set: The tag set is {B-PRE1, I-PRE1, B-PRE2, I-PRE2, B-WORD, I-WORD, B-SUFF, I-SUFF, O} where I denotes the inside of a segment and B denotes the beginning of a segment. In principle, the tag set allows for

[5] http://cl.aist-nara.ac.jp/~taku-ku/software/yamcha

[6] http://cnts.uia.ac.be/conll2003/ner/bin/conlleval

[7] We refer to the cliticized portion of the surface form as a **stem** rather than lemma since, as we will see, the process of clitic tokenization does not necessarily result in lemmas.

Table 9.4. IOB example

MSA	*Transliteration*	IOB Tag
و	*w*	B-PRE1
ب	*b*	B-PRE2
ح	*H*	B-WORD
س	*s*	I-WORD
ن	*n*	I-WORD
ا	*A*	I-WORD
ت	*t*	I-WORD
ه	*h*	B-SUFF
م	*m*	I-SUFF

internal structure for PRE1 and PRE2. In practice PRE1 and PRE2 are mostly proclitic tags which, in unvocalized text, are usually only one character long.[8] SUFF is an enclitic, and WORD is the derivational stem plus any inflectional affixes, and/or the determiner Al. Both latter categories could have internal structure.

Example: Figure 9.1 illustrates a training example with the IOB tags and the feature sets. All the training and test data is represented using the Buckwalter transliteration scheme.

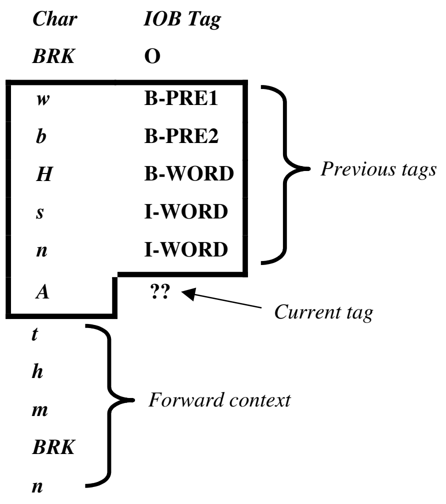


Fig. 9.1. Sequence SVM training for TOK. Current position is *A*; context features are the previous 5 characters and their tags, and the following 5 characters

[8] There are very few cases that occur with I-PRE1 or I-PRE2 such as لابد ***lAbd*** *(it is necessary)* where the ATB sometimes separates the ***lA*** from ***bd***.

In Figure 9.1, the character in focus is ***A***, and the context features are the five preceding character segments: *w* B-PRE1, ***b*** B-PRE2, ***H*** B-WORD, ***s*** I-WORD and ***n*** I-WORD and the five following character segments: ***t, h, m, BRK,*** and ***n***. Note that ***n*** in this case happens to be the character segment beginning the following word. The classifier is run left-to-right, and so the labels of the 5 following segments are not used as features to the classifier. The class label of the focus character is presented to the classifier during supervised training, but not during testing. Once the models are learned over the training data, the test data is disambiguated where each word is split into the respective chunks of proclitic/prefix, stem and enclitic/suffix.

While the SVM classifier is generally successful at tokenization, any purely segmentation-based approach does not deal well with **allomorphy** and **morphophonology**. Allomorphs are different realizations of the same morpheme that occur in different surface contexts. Morphophonology is the general term for changes in the phonemes (and letters) in a morpheme that occur when morphemes are combined. Undoing these morphophonological changes is part of **lemmatization**, the task of recovering the lemma that underlies the surface word form. True lemmatization also involves parsing off inflectional morphology, so our goal in this chapter is only a subset of lemmatization: producing word forms that are as close as possible to lemmas barring the further processing of inflectional morphology.

Our SVM segmenter's inability to model these morphophonological changes led to three classes of errors, requiring further processing. Two of these are described here; one is reserved for the next section.

- **Allomorphs of the definite article in the context of the proclitic preposition *l* '*to*'**

  The definite article ***Al*** *the* has a distinct allomorph after the preposition ***l*** *to*; it appears simply as ***l***. For instance, للعراق ***llςrAq*** (TO THE IRAQ, *to Iraq*), after the preposition clitic ***l*** is segmented off by our tokenizer, produces the form ***lςrAq***. The resulting form is now ambiguous between *to Iraq* and *the Iraq* because the initial ***l*** word is the allomorph of the definite marker ***A***. In this context (after ***l*** '*to*') the phrase can only mean *the Iraq*. The full definite article lemma thus needs to be restored in such contexts; ***lςrAq*** becomes ***AlςrAq***.

- **Allomorphs of Alif maqsura, *ý*, in closed-class words before enclitics**

  In closed class words such as the prepositions على ***ςlý*** (*on*) or لدى ***ldý*** (*at*), the word final ***ý*** appears as a surface ***y*** in the context of enclitics. Therefore, as a result of clitic tokenization, ***ldyhm*** would result in ***ldy +hm***. It is worth noting that ***ldy*** on its own (not in the context of an enclitic) corresponds to *with me* or *at my place*, exactly *chez moi*, in French. Thus to produce the correct lemma for the prepositions in the context of an enclitic, the final ***y*** needs to be restored to a ***ý.***

The required lemmatization for these two allomorphy cases is completely predictable from the context, therefore both cases are handled deterministically in our system with hand-written rules. In the case of the determiner, we only change an initial ***l*** to ***Al*** if it was preceded by an ***l*** before clitic tokenization. In case of the word final ***ý***, we change a closed class word final y to a ***ý*** in the context of an enclitic pronoun. Some examples of the closed class words are ***ςlý*** (*on*), ***Alý*** (*to*), ***ldý*** (*at*), ***mdý*** (*extent*).

9.5.1 Tokenization Results

The ATB text comes in pre-lemmatized form. In order to increase the robustness of our system, we chose to un-lemmatize the ATB text, to create more natural input. We un-lemmatized the ATB via hand-written rules. For example, the preposition ***Alý*** (*to*) followed by the enclitic ***hm*** (*them*) is written as the lemma ***Alý*** in the ATB. However, in naturally occurring text it would be the stem ***Aly*** with the enclitic suffix ***hm***. Therefore, we wrote a rule to transform closed class words that have a ***ý*** word finally in the context of an enclitic suffix into a ***y***. We acquired the token boundaries for training from the ATB.

Table 9.5 presents the results obtained using SVM-TOK, compared against two rule-based baseline approaches, RULE and RULE+DICT.

RULE marks a prefix if a word starts with one of five proclitic letters described in Section 9.4.1. A suffix is marked if a word ends with any of the object or possessive pronouns, enclitics, mentioned above in Section 9.4.1. A small set of 17 function words that start with the proclitic letters is explicitly excluded.

RULE+DICT applies the tokenization rules in RULE if the token does not occur in a dictionary. The dictionary used comprises the 47,261 unique unvocalized word entries obtained from the first column of Buckwalter's dictStem, freely available with the BAMA distribution. In some cases, dictionary entries retain inflectional morphology and clitics.

9.5.2 Tokenization Discussion

Performance of SVM-TOK is very high with an $F_{ß=1}$=99.1. The task, however, is quite easy, and SVM-TOK is only about 5 F-score points better (absolute) than the better baseline, RULE+DICT. While RULE+DICT could certainly be improved with larger dictionaries, the largest dictionary will still have coverage problems if it is not aligned well with the corpus at hand. Hence, there will remain a role for a data-driven approach such as SVM-TOK. Table 9.6 illustrates the performance of SVM-TOK per chunk.

Table 9.5. Results (%) of SVM-TOK compared against RULE and RULE+DICT

System	Acc	Prec	Rec	$F_{ß=1}$
SVM-TOK	**99.8**	**99.1**	**99**	**99.1**
RULE	96.8	86.3	91.1	88.6
RULE+DICT	98.3	93.7	93.7	93.7

Table 9.6. Results (%) for SVM-TOK per chunk category

Chunk	Prec	Rec	$F_{\beta=1}$
PRE1	98.5	98.2	98.3
PRE2	97.4	85.2	92
WORD	99.3	99.3	99.3
SUFF	96.7	96.4	96.5

The results overall for all the chunks are in the high 90s except for PRE2. Table 9.7 further illustrates a confusion matrix for the different classes in SVM-TOK.

The empty cells in Table 9.7 indicate no confusion between these classes. The first column contains the gold tag, *i.e.* the manually assigned tag in the treebank. The table reads as the percentage of time a class from column 1 is confused with one of the other classes. Therefore, B-PRE1 is incorrectly tagged as a B-WORD 1.8% of the time. The largest number we see in this matrix is the percentage of time B-PRE2 is confused with the beginning of a word B-WORD. This is expected since the B-PRE2 has two ambiguous boundaries. An example of the errors is the word ***wbAlm***. The correct segmentation is ***w#bAlm*** (*and Palm*), SVM-TOK hypothesized an extra B-PRE2 chunk therefore the resulting segmentation is as follows: ***w#b#Alm*** (*and with pain*). This is a result of the fact that ***bAlm*** (*Palm*) is a proper name which was not seen in the training data while the word ***Alm*** (*pain*) occurs frequently as a noun in the training data. Most of the problems observed are problems with proper names. Proper names that begin with characters that could be proclitics will always pose a problem for this task unless we incorporate some form of Named Entity Tagging in the process.

The results obtained in this study are not significantly different from those obtained using data from ATB1v2 alone (Diab et al., 2004). The results in our previous study with one fifth of the data were the same, $F_{\beta=1}$=99.1. We attribute this finding to a hypothesized ceiling effect on performance given the relatively impoverished set of features being used. Unless we include richer morphological features, we do not anticipate a significant improvement on performance.

Our results are not directly comparable to Darwish's work (Darwish, 2002) nor to the segmentation task of Lee (Lee et al., 2003) since they include inflectional morpheme segmentation.

Table 9.7. SVM-TOK classes confusion matrix (%)

Class	B-PRE2	B-WORD	I-WORD	B-SUFF	I-SUFF
B-PRE1		1.8			
B-PRE2		11.4	3.4		
B-WORD	0.18		0.24		
I-WORD		0.06		0.05	0.03
B-SUFF		0.17	3.4		
I-SUFF			2.8		

9.6 Singular Noun Feminine Marker Restoration

We saw above that some allomorphy problems with our segmenter could be dealt with by hand-written rules. We turn now to a more difficult allomorphy issue. The ending for singular feminine noun and adjective lemmas is a word final *tā' marbūTa*, ***ħ.*** When a singular feminine noun is followed by an enclitic pronoun, the word final *tā' marbūTa* surfaces as a regular ***t***.[9] This noun-final ***t*** needs to be converted back to the feminine marker as it serves as an important clue for the POS tagging step in processing.

This nominal feminine marker is difficult to disambiguate since a ***t*** can be followed by an enclitic in multiple cases; the singular feminine noun followed by a possessive pronoun, and a verb followed by an object pronoun. Only in the nominal case does the final ***t*** need to be lemmatized to a *tā' marbūTa*, ***ħ***. At this stage of our processing, however, we do not have access to POS information. Indeed, we need to restore the *tā' marbūTa*, ***ħ***, onto feminine nouns in order to aid the POS tagging step.

In our running example, the stem ***Hsn*** if viewed as a lemma corresponds to the adjective meaning *good* but in our example it is a noun meaning *virtue* as it is pluralized and followed by a possessive enclitic pronoun. The lemma form of the noun is ***Hsnħ***. The translation of *and by their virtue* would be ***wbHsnthm***. After running our tokenization step, the word will be rendered ***Hsnt +hm.*** The word ***Hsnt*** without vocalization is a verb meaning *to be good* or *to improve* or *she has improved* and in the context of the enclitic, this construction could be translated as *she/it improved them* where the enclitic would be the object pronoun. However, the context of the proclitics preceding the word make the only possible POS associated with ***Hsnt*** be a noun interpretation. But ***Hsnt*** does not exist as a noun in MSA. Therefore, ***Hsnt*** should be converted to its lemma form ***Hsnħ***.[10]

We deal with the lemmatization of the feminine marker as another classification problem. Feminine marker restoration (or feminine lemmatization) is hence a one-of-three classification task. Each tokenized segment resulting from SVM-TOK is labeled with a class indicating whether it has a ***t*** word finally, and if so, should it be restored to an ***ħ.*** Therefore, the sequence model has the following form:

Input: A sequence of tokens processed from left-to-right.

Context: A window of -2/+2 tokens centered at the focus token.

Features: Where the features for our tokenization classifier were position sensitive, the features for the feminine lemmatization classifier are instead bag-of-n-grams features. Every character n-gram, for all n <= 4 that occurs in the focus

[9] Adjectives do not cliticize for enclitics.

[10] The plural form of feminine nouns do not undergo the *tā' marbūTa* restoration as plural lemmas typically end with ***At.*** We do not address inflectional morphology, the word ***HsnAt***, in our running example, is left as is, i.e. we do not reduce it further to ***Hsnħ*** and the plural marker ***At***.

token, are used as features. Additional features include the 5 tokens themselves, and feminine lemma tag decisions for the previous tokens within the context.

Tag Set: Three classes KTT (Keep the word final T), NTT (Not a word final t), CTP (Convert word final T to a ***ħ***, which is a ***p*** in Buckwalter encoding).

Example: Table 9.8 illustrates an example of the data representation for training.

Table 9.8 shows five words and their corresponding ngram based features. The verb ***AfAdt*** *(declared/reported/announced)* has a word final ***t*** that should be kept as is, hence, the class label KTT. ***HSylt*** *(result)*, on the other hand, is a feminine singular noun that should have a word final ***ħ***, since it is followed by an enclitic. Therefore, it is labeled with a class CTP. In the ATB, the lemma form is rendered in the parsed trees. However, we deterministically convert the ***ħ*** noun finally to a regular ***t*** in the context of an enclitic possessive pronoun for training purposes. As illustrated in the example, the rest of the words ***hA***, ***nhAyħ, rsmyħ*** (*it/she, ending, official*, respectively) do not end in a ***t*** in the first place, therefore they are labeled NTT. ***nhAyħ*** is a noun but it is not in the context of an enclitic therefore it is left intact. We refer to this module as SVM-LEM.

9.6.1 Lemmatization Results

Again, for training we need to un-lemmatize the ATB: feminine nouns ending with an ***ħ*** in the ATB in the context of a suffix enclitic possessive pronoun are converted to their stem forms. For example, the noun and its ensuing possessive pronoun, ***Hsnth*** (*his virtue*) is listed in the ATB as ***Hsnħ*** plus the suffix enclitic 3rd person masculine singular ***h***. For training purposes, we automatically convert the lemma form ***Hsnħ***, as produced by the treebank annotators, to the stem form ***Hsnt***.

Results shown here assume gold clitic tokenization. Table 9.9 shows the results obtained with SVM-LEM against a simple random baseline, Rand-LEM. Rand-LEM randomly converts word final ***t*** for nouns to ***ħ***. Rand-LEM has access to basic category POS information such as noun, verb, adjective, but does not have access to gender or number information. Hence in the case of Rand-LEM, the system would convert a ***t*** final noun randomly to a ***ħ***. POS information is not one of the features used by SVM-LEM.

Table 9.8. Training example for SVM-based feminine marker restoration module

Translit.	Features								Class
AfAdt	***A***	***Af***	***AfA***	***AfAd***	***t***	***dt***	***Adt***	***fAdt***	KTT
HSylt	***H***	***HS***	***HSy***	***HSyl***	***t***	***lt***	***ylt***	***Sylt***	CTP
hA	***h***	***hA***	–	–	***A***	***hA***	–	–	NTT
nhAyħ	***n***	***nh***	***nhA***	***nhAy***	***ħ***	***yħ***	***Ayħ***	***hAyħ***	NTT
rsmyħ	***r***	***rs***	***rsm***	***rsmy***	***ħ***	***yħ***	***myħ***	***smyħ***	NTT

Table 9.9. SVM-LEM results (%) compared against a random baseline Rand-LEM

System	Acc
Rand-LEM	92.16
SVM-LEM	99.84

9.6.2 Lemmatization Discussion

SVM-LEM significantly outperforms Rand-LEM resulting in a close to perfect performance. This performance is an indication of the ease of the task. All the remaining errors are due to confusion about proper nouns and verbal forms with a feminine ending.

SVM-LEM correctly converts the word final ***t*** to ***ħ*** 99.2% of the time. However, it over generalizes and converts some verb final ***t*** as well. SVM-LEM confuses KTT with CTP 3.5% of the time. The confusion typically occurs in the context of an enclitic object pronoun with a verb that has a word final ***t*** such as ***AqAmt*** (*held*) and ***sqTt*** (*slipped*); as well as with proper names such as ***Ayfyrt***.

9.7 Part of Speech Tagging

Part-of-Speech Tagging (POS) is the problem of assigning each token its correct part-of-speech class such as verb, noun, adjective, etc. In this chapter, we used the ATB Reduced Tag Set (RTS). The RTS is distributed with the ATB documentation from the LDC and is illustrated in Table 9.10. This tag set reflects number for nouns and some tense information for verbs. In the RTS, gender, definiteness, and case information are lost for nouns and verbs, number information is lost for adjectives, and mood, future tense and aspect information are lost for verbs. RTS comprises 24 POS tags, reduced from the 139 morphosyntactic tags produced by the Buckwalter morphological analyzer, BAMA. RTS was created for two reasons: to render parsing tractable; and to enhance the compatability between the English and Arabic Treebanks.

We model this task as a 1-of-24 classification task, where the class labels are drawn from RTS. This is a sequence model similar to the SVM-TOK and SVM-LEM. The model can be described as follows:

Input: A sequence of tokens processed from left-to-right.

Context: A window of –2/+2 tokens centered at the focus token.

Features: Every character n-gram, n$\leq$ 4 that occurs in the focus token, the 5 tokens themselves, their ‘type’ from the set {alpha, numeric} indicating if a number is included in the token of interest, and POS tag decisions for previous tokens within context. The feature set is thus similar to the feature set adopted for the SVM-LEM process, modulo the type feature.

Tag Set: The 24 tags are listed in Table 9.10.

Table 9.10. The ATB Reduced Tag Set (RTS) and their meanings

Tag	Meaning	Tag	Meaning
CC	Conjunction	*PRP$*	Possessive Pronoun
CD	Number	*RB*	Adverb
RP	Particle	*UH*	Interjection
DT	Determiner	*VB*	Imperative Verb
FW	Foreign word	*VBD*	Perfective Verb
IN	Preposition	*VBN*	Passive Verb
JJ	Adjective	*VBP*	Imperfective Verb
NN	Singular Noun	*WP*	Interrogative particle
NNS	Plural Noun	*WRB*	Interrogative adverb
NNP	Singular Proper Noun	*NNPS*	Plural Proper Noun
PRP	Pronoun	*NO_FUNC*	Unknown

Example

Table 9.11 illustrates an input training example for POS tagging task

9.7.1 Part-of-speech Tagging Results

Results reported here assume gold tokenization and lemmatization. Table 9.12 shows the results obtained with the SVM-POS compared against those obtained with a simple baseline, Baseline-POS. Baseline-POS assigns a test token the most frequent POS tag associated with it as observed in the training data. If the token does not occur in the training data, the token is assigned the NN tag as a default tag.

Table 9.11. Training example illustrating the alpha vs. numeric type feature

Translit	Features									Class
AfAdt	***A***	***Af***	***AfA***	***AfAd***	***t***	***dt***	***Adt***	***fAdt***	Alp	VBD
HSylt	***H***	***HS***	***HSy***	***HSyl***	***t***	***lt***	***ylt***	***Sylt***	Alp	NN
hA	***H***	***hA***	–	–	***A***	***hA***	–	–	Alp	PRP$
nhAyħ	***n***	***nh***	***nhA***	***nhAy***	***ħ***	***yħ***	***Ayħ***	***hAyħ***	Alp	NN
rsmyħ	***r***	***rs***	***rsm***	***rsmy***	***ħ***	***yħ***	***myħ***	***smyħ***	Alp	JJ
l	***l***	–	–	–	***l***	–	–	–	Alp	IN
100	***1***	***10***	***100***	–	***0***	***00***	***100***	–	Num	NN

9.7.2 Part-of-speech Tagging Discussion

The performance of SVM-POS is significantly better than the baseline. Table 9.13 further illustrates the accuracy per POS tag together with their respective most confusable POS tag.

Closely observing Table 9.13, we note that the worst results rendered are those for NO_FUNC followed by VB and VBN. We experimented with changing NO_FUNC to NNP and the results went up to 97.6%. There are only 3 imperative verbs, VB, in the test corpus, compared to 12 in the training corpus. However, the VB category is highly confusable with NN due to similar contextual clues. VBN, passive verb, is confused with the imperfective verb, VBP, 21% of the time. With the lack of diacritics marking passivization, VBN is a very hard category to predict.

Table 9.12. POS tagging results comparing Baseline-POS against SVM-POS

System	Acc%
Baseline-POS	92.2
SVM-POS	96.6

Table 9.13. Results per POS category and their respective confusable POS tag

Tag	Acc %	Highest Confusion Tag	Confusion %
CC	99.9	NN	0.01
CD	97.1	NN	1.6
DT	100		
IN	99.6	RP	0.18
JJ	93.6	NN	4.3
NNPS	100		
NNP	91.6	NN	6.5
NNS	97.3	NN	1.3
NN	97	JJ	1.3
NO_FUNC	9.5	NN	29.5
PRP$	98.2	PRP	1.8
PRP	97	PRP$	2.7
PUNC	100		
RB	92.9	IN	4
RP	90.1	IN	5.9
UH	76	WP	8
VBD	92.5	NN	4.7
VBN	54.1	VBP	21
VBP	96	VBD	1.9
VB	37.5	NN	50
WP	98.3	IN	1.2
WRB	74.1	IN	17.7

In passive forms, the initial character is marked with a *Damma* diacritic – the consonant/long vowel is followed by a short vowel ***u*** – indicating passivization. Hence, in newspapers for instance, even though all diacritics and short vowels are missing, one of the exceptions would be the addition of the short ***u*** vowel onto verbs to mark passives. The confusion arises from the fact that MSA may be pro-drop which permits the absence of an overt subject hence the lack of a subject is not necessarily a signal for a passive construction.

RB and WRB are consistently confused with IN. Upon closer inspection of the data, we note that the closed word class in these two categories are the ones consistently annotated as IN. The RB ***Htý*** *(even)* and the WRB ***ndmA*** *(at the time)*, ***bςdmA*** *(after the time),* ***TAlmA*** *(as long as)* are all marked as IN.

Another two confusable categories are the NN and JJ POS tags. These two are a canonical confusable POS in many languages. The problem is more pronounced in MSA since any JJ maybe an NN depending on context. In classical Arabic, there is no real notion of an adjective as a stand-alone category. This is evidenced by the fact that any adjective (a word describing or modifying a noun) in Arabic could be used as a noun and a proper noun. This is reflected in the manual annotations in the Treebank where the annotators are not being consistent due to this fundamental confusion between the two categories hence we see, for instance, the word for *United* in *United States of America* or *United Nations* is randomly tagged as a noun, or an adjective in the training data. This inconsistency in annotation renders the two categories highly confusable.

Comparing the results to those obtained in our previous work, (Diab et al., 2004), we note a 1% overall improvement (from 95.5% to 96.6% accuracy) though we are using 4 times the data in these experiments. This indicates a ceiling effect on the performance of such a language independent approach that does not exploit any of the language specific rich morphological features available in the data. Our results are not comparable to the work in (Khoja, 2001) or that in (Habash & Rambow, 2005) since both studies use different evaluation metrics and different tag sets.

9.8 Base Phrase Chunking

In this task, we use a setup similar to that of Kudo and Matsumoto (2000), where 9 types of chunked phrases are recognized using a phrase IOB tagging scheme; Inside (I) a phrase, Outside (O) a phrase, and Beginning (B) of a phrase. The 11 chunk phrases are: ADJP (Adjectival Phrase), ADVP (Adverbial Phrase), CONJP (Conjunctive Phrase), INTJ (Interjective Phrase) such as *oh*, LST (enumerated list) such as *first second* etc., PP (Prepositional Phrase), PRT (Partitive Phrase), NP (Noun Phrase), S (Sentential Unit), SBAR (Subjunctive Phrase), and VP (Verb Phrase). Thus, in principle, the task is a one of 23 classification task (since there are I and B tags for each chunk phrase type, and a single O tag). However, in practice we only have 19 classes, since four of the I phrases are not instantiated. Three of the four cases constitute singleton words, INTJ, LST, and PRT. The fourth chunk is the S category marking only the beginning of an S chunk. The 19 classes are as follows: B-ADJP, I-ADJP, B-ADVP, I-ADVP, B-CONJP, I-CONJP, B-INTJ, B-LST, B-PP, I-PP, B-PRT, B-NP, I-NP, B-S, B-SBAR, I-SBAR, B-VP, I-VP, O. The training data is derived from the ATB using the ChunkLink software (Buchholz et al., 1999).[11] ChunkLink flattens the tree to a sequence of base (non-recursive) phrase chunks. For example, a token occurring at the beginning of a noun phrase is labeled as B-NP. Table 9.14 illustrates the tagging scheme.

Input: A sequence of (word, POS tag) pairs.

Context: A window of –2/+2 tokens centered at the focus token.

Features: Word and POS tags that fall in the context along with previous IOB tags within the context.

Tag Set: The tag set comprises 19 tags: B-ADJP, I-ADJP, B-ADVP, I-ADVP, B-CONJP, I-CONJP, B-INTJ, B-LST, B-PP, I-PP, B-PRT, B-NP, I-NP, B-S, B-SBAR, I-SBAR, B-VP, I-VP, O

9.8.1 Base Phrase Chunking Results

Reported results assume gold POS tags and gold tokenization.

9.8.2 Base Phrase Discussion

The overall performance of SVM-BPC is $F_{ß=1}$ = 91.6 and an accuracy of 93.4%. These results are interesting in light of state-of-the-art for English BPC performance which is at an $F_{ß=1}$ score of 93.48 on five of the chunk types listed here, this is compared against a baseline of 77.7 in CoNLL 2000 shared task (Tjong Kim Sang & Buchholz, 2000). Per-class F-scores are displayed in Table 9.15. The best results obtained are for VP and PP, yielding $F_{ß=1}$ scores of 97.8 and 98.5, respectively. There is room for improvement in the other categories however. We used the ChunkLink software as is, without special modification for the ATB. We believe that adding features such as definiteness, and gender should aid with the performance especially with ADJP and NN categories. If we exclude the PRT, LST, INTJ and S categories from the set of chunks to be discovered, the results go up to an $F_{ß=1}$ =92.

Table 9.14. Base phrase chunking tagging scheme

Tags	*O*	*B-VP*	*B-NP*	*I-NP*
MSA	و	قالت	روث	شوارتز
Transliteration	***w***	***qAlt***	***Rw***	***šwArtz***
Gloss	And	Said	Ruth	Schwartz

[11]http://ilk.uvt.nl/~sabine/chunklink

Table 9.15. Results (%) for SVM-BPC per chunk

BPC	Prec	Rec	$F_{ß=1}$
ADJP	72.6	56.1	63.3
ADVP	77.3	69	73.9
CONJP	81.8	85.7	83.7
INTJ	37.5	33.3	35.3
LST	0	0	0
NP	89.1	90.1	89.6
PP	98.4	98.7	98.5
PRT	92.8	93.2	93
S	50	26.1	34.3
SBAR	89.3	90	89.7
VP	98.8	96.8	97.8
Total	91.8	91.5	91.6

To further analyze the performance of SVM-BPC, we created a confusion matrix for the 19 different classes. There are only two cases of LST in the test data and they are always confused with NP. There are 9 cases of INTJ in the test data and 3 of them are confused with an O class while another 3 are confused with an NP class. Finally, the S class is at a low $F_{ß=1}$ of 34.3. An S tag is always confused (17 out of 23 times) with an SBAR. This reflects some inconsistency in the ATB annotations of S and SBAR categories. Moreover, the S tag is by default a recursive tag which requires more context than the window of +/–2 utilized in our current set-up.

9.9 Conclusions & Future Directions

We have presented a unified machine-learning approach using SVMs to solve the problem of automatically annotating Arabic text with tags at different levels: Clitic tokenization at the morphological level (including the restoration of the word final feminine marker for singular nouns in the context of possessive enclitics); POS tagging at the lexical level, and BP chunking at syntactic level. The adopted framework is language independent and yields highly accurate results for each task. The task of clitic tokenization is performed at an $F_{ß=1}$ score of 99.1. Feminine marker restoration is performed with 99.8% accuracy. Part-of-speech tagging achieves 96.6% accuracy. The POS tagging results are not very much below the best English POS tagging performance of 97.5% (Toutanova et al., 2003). Base-phrase chunking achieves an $F_{ß=1}$ score of 91.6. To the best of our knowledge, these are the best results reported for these tasks in Arabic natural language processing. However, there is ample room for improvement using language specific features for the different processing levels. Compared to our previous study with a quarter the data size used in these current experiments, we do not observe any large improvements in any of the modules. This is an indication of a ceiling effect when no language specific information is included as features in our models. One of the goals for this study was to create a baseline/lower bound that is language independent to investigate how well can these approaches perform with little knowledge of the language.

We are currently exploring means of incorporating the rich morphological features in Arabic to gain in performance. We are exploring means of incorporating more features for the clitic tokenization module and toying with the idea of automatically discovering inflectional boundaries. We are varying the POS tag set to include definiteness, gender and number information for the different tags. We are investigating means of giving more depth to the BPC module, i.e. we are interested in more recursive structure and what that entails for syntactic parsing in a discriminative framework.

Acknowledgments

We would like to thank Antal van den Bosch and Nizar Habash for valuable comments on earlier versions of this manuscript. This work was partially supported by ARDA AQUAINT and by the National Science Foundation via a KDD Supplement to NSF CISE/IRI/Interactive Systems Award IIS-9978025. And also for the first and third authors by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023.

References

Allwein, E. L., Schapire, R. E. & Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, *1*, 113–141.

Buchholz, S., Veenstra, J. & Daelemans, W. (1999). Cascaded grammatical relation assignment. In *Proceedings of EMNLP/VLC* (pp. 239–246).

Darwish, K. (2002). Building a shallow Arabic morphological analyser in one day. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages* (pp. 47–54), Philadelpia, PA.

Diab, M., Hacioglu, K. & Jurafsky, D. (2004). Automatic tagging of Arabic text: From raw text to base phrase chunks. In *Proceedings of North American Association for Computational Linguistics* (NAACL, pp. 149–152).

Habash, N. & Rambow, O. (2005). Arabic Tokenization, Part-of-Speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the Association for Computational Linguistics* (ACL, pp. 573–580).

Habash, N. & Sadat, F. (2006). Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the North American chapter of the Association for Computational Linguistics* (NAACL, pp. 49–52).

Hacioglu, K. & Ward, W. (2003). Target word detection and semantic role chunking using support vector machines. In *Proceedings of Human Language Technology and North American Association for Computational Linguistics* (HLT-NAACL, pp. 25–27).

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning* (EMCL, pp. 137–142).

Khoja, S. (2001). APT: Arabic part-of-speech tagger. In *Proceedings of the North American Association for Computational Linguistics Student Workshop* (pp. 20–25).

Kudo, T. & Matsumato, Y. (2000). Use of support vector learning for chunk identification. In *Proceedings of the 4th Conference on Computational Natural Language Learning* (CoNLL, pp. 142–144).

Lee, Y.-S., Papineni, K., Roukos, S., Emam, O. & Hassan, H. (2003). Language model based Arabic word segmentation. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics* (pp. 399–406).

Maamouri, M., Bies, A. & Buckwalter, T. (2004). The Penn Arabic treebank: Building a largescale annotated Arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Ramshaw, L. A. &. Marcus, M. P. (1995). Text chunking using transformation-based learning. In *Proceedings of the Association for Computational Linguistics Workshop on Very Large Corpora* (pp. 82–94).

Tjong Kim Sang, E. & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 4th Conference on Computational Natural Language Learning* (CoNLL, pp. 127–132).

Toutanova, K., Klein, D., Manning, C. & Singer, Y. (2003). Feature-Rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of Human Language Technology and North American Association for Computational Linguistics* (HLT-NAACL, pp. 252–259).

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York: Springer Verlag.

10

Supervised and Unsupervised Learning of Arabic Morphology

Alexander Clark

Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, United Kingdom

alexc@cs.rhul.ac.uk

Abstract: The broken plural in Arabic is a canonical example of nonconcatenative morphology. We discuss the supervised and unsupervised learning of this type of transduction using different techniques, based on the use of stochastic transducers, trained with the Expectation-Maximisation algorithm. A basic method for supervised learning using the transducers is discussed and then a more advanced technique using a memory-based learning technique with a distance derived from the Fisher kernel of the model. We then discuss how these algorithms can be employed for unsupervised learning, modelling the alignment between the strings as a hidden variable

10.1 Introduction

In this chapter we discuss the problem of learning morphology, with particular reference to the problem of non-concatenative morphology best exemplified by the Arabic broken plural. We shall look at two formulations of the problem; one where we have to learn the mapping or transduction from a base form to an inflected form (learning from a set of pairs) and the second where we must learn from two sets of words one of which consists of base forms and one of which is composed of inflected forms.

Existing approaches to these problems have focussed on the acquisition of concatenative morphology – that is to say languages where the vast majority of the morphological processes are based on the operation of the concatenation of strings. For example, in English the present participle of verbs is formed by concatenating the stem with the suffix “-ing”; “walk” becomes “walking”. Under the assumption that a language uses exclusively concatenative morphology, the problem of learning the morphology of the language reduces to finding a suitable segmentation of the strings of the language. This approach has successfully been pursued by several researchers, most notably Goldsmith (2001). However, many languages use non-concatenative processes to a greater or lesser extent: even English has a residual process of vowel change or ablaut for some irregular past tenses – ring/rang for example. If one is interested in studying this phenomenon, or rather the learnability of this phenomenon, then the obvious example to seek out is the Arabic broken plural, which we shall discuss below. This is in many respects the canonical example of non-concatenative morphology. Its great complexity means that an algorithm that can learn this mapping is likely to be able to learn similar mappings in other languages that have a less exaggerated form of the same process.

The motivation for studying these problems is to examine the sorts of algorithms that can learn in an unsupervised way from naturally occurring data in the hope that this will cast light on first language acquisition. Thus our interest in Arabic arises from the particular challenges that it presents for learning algorithms. We have chosen to explore the potential of algorithms that do not exploit any putative innate domain-specific knowledge of language. Thus we shall restrict ourselves to general-purpose learning algorithms, that are given no further information about the problem, and where the only learning biases are those that are implicitly defined by the learning algorithms themselves. Methodologically it is appropriate to exhaust the possibilities of such algorithms before proceeding to less parsimonious hypotheses.

As has been known for some time, finite-state methods are in large part adequate to model morphological processes (Kaplan & Kay, 1994). A standard methodology is that of two-level morphology (Koskenniemi, 1983), which is capable of handling the complexity of Finnish, though it needs substantial extensions to handle non-concatenative languages such as Arabic (Kiraz, 1994). These models are primarily concerned with the mapping from *deep* lexical strings, that is to say from strings that include abstract morphemes, to surface strings. Here we discuss algorithms that learn transductions between pairs of uninflected and inflected words, that is to say between pairs of surface forms. This characterisation of a morphological process as a transduction from the lemma form to the inflected form is overly simplistic for a number of reasons. Firstly, in Arabic as in many other languages the inflected form is not purely phonologically specified, but is at least in part lexically specified. Secondly, the same phonological principles operate in many different areas and modelling each transduction separately will inevitably fail to will inevitably fail to capture important generalisations.

The base of our technique is a simple finite state model, but one that is stochastic. We use a simple (generative) maximum likelihood technique to train the models, starting from a randomly initialised model. This technique, while able to model the individual transductions, turns out not to work very well on the task of selecting which transduction should be applied, particularly with the mixture of regular and irregular forms that is so characteristic of morphological processes in general. It thus turns out to be necessary to use some more sophisticated learning technique to achieve good results, at least with the comparatively small training sets that we use here. Though there are a number of possible techniques that could be applied in this case, we have chosen to apply one that has been used before in morphology. Memory-based learning techniques, based on principles of non-parametric density estimation, are a powerful form of machine learning well-suited to natural language tasks. A particular strength is their ability to model both general rules and specific exceptions in a single framework (Van den Bosch & Daelemans, 1999).

They have generally only been used in supervised learning techniques where a class label or tag has been associated to each feature vector. Given these manual or semi-automatic class labels, a set of features and a predefined distance function new instances are classified according to the class label of the closest instance. However these approaches are not a complete solution to the problem of learning morphology, since they do not directly produce the transduction. The problem must first be converted into an appropriate feature-based representation and classified in some way. The techniques presented here operate directly on sequences of atomic symbols, using a much less articulated representation, and much less input information.

While the work presented here is far from a complete analysis of the problem of learning Arabic morphology, it does establish that it is possible to learn even the most complex transductions without needing to add any domain-specific biases, but rather using biases implicit in general-purpose learning algorithms.

We shall start by presenting the basic device we use to model these transductions. Section 10.2 describes the sort of stochastic transducer we shall use, as well as the learning or training algorithms we use, and the inferencing process. We then discuss in Section 10.3 the use of Fisher kernels and information geometry. In Section 10.4 we describe the data we use, and present some experiments. We then move on to discussing the problem of semi-supervised learning (Section 10.5).

10.2 Stochastic Transducers

The modelling device we use in this chapter is the stochastic finite state transducer. We will describe the basic model and then discuss the particular models that we use in more detail. Formally the transducers we use are a tuple $(A, B, S, p, q, s_0, s_f)$ where A and B are finite non-empty sets that are the input and output alphabets, S is a finite set of states, s_0 and s_f are the initial and final states respectively. p and q are the transition and output functions. p is a function from $S \times S$ to $[0, 1]$ such that for every state s

$$\sum_{s' \in S} p(s'|s) = 1 \qquad (1)$$

This represents the probability that the transducer will make transition from state s to state s'. q is an output function from $S \times (A \cup \{\epsilon\}) \times (B \cup \{\epsilon\})$ to [0,1] such that for every state s

$$\sum_{a \in (A \cup \{\epsilon_A\})} \sum_{b \in (B \cup \{\epsilon_B\})} q(a, b|s) = 1 \qquad (2)$$

where ϵ_A and ϵ_B are empty strings from A^* and B^*. This represents the probability that given that the model is in state s it will output the symbol a on the input stream, and the symbol b on the output stream. These normalisation requirements will ensure that this defines a probability distribution over $A^* \times B^*$.[1]

There are a number of ways in which this differs from the finite state transducer as normally defined. First, these transducers are stochastic. They define a joint distribution over the input and output distributions, not a conditional distribution. Secondly, they are non-deterministic. We have in no way restricted these so that the underlying sequence of states is determined by the input or the output or even both. Thus in general there may be exponentially many possible sequences of state transitions given fixed input and output strings, or indeed an infinite number, though we later will disallow this possibility. Thirdly, there is a symmetry in the way we have defined this. Though the transductions are clearly not symmetric since we might have disjoint input and output alphabets, for every transducer T from A^* to B^* it is trivial to define a transducer T from B^* to A^* that defines the same distributions, but with input and output swapped. Thus instead of using the terms input and output for the two streams of symbols generated by the model, we shall sometimes use the terms left and right streams. Fourthly, since we have added the possibility of ϵ inputs and outputs, it is possible for the input and output strings to have different lengths. Finally, note that we have attached the output function to states, rather than to transitions. This brings the formalism much closer to that of a Hidden Markov Model. Indeed, these models are sometimes called Pair Hidden Markov Models (Durbin, Eddy, Krogh, & Mitchison, 1998).

The first modification we will make is that we consider the input and output alphabets to be identical. In other applications of these learning techniques, for example text-to-speech, one might have an input alphabet of letters, and an output alphabet of phonemes, but in this domain we are interested in phoneme-to-phoneme transductions. As a consequence of this we will also reduce the parameter space. We will only allow output functions of the following three types:

- q_{11} functions which output the *same* character on both streams.
- q_{10} functions which output a character on the left stream, and no character on the right stream.
- q_{01} functions which output a character on the right stream, and no character on the left stream.

Thus we have removed two possibilities – producing an epsilon on both strings, and producing different characters simultaneously on both streams. Of course, this latter effect can be achieved by having a q_{10} output followed by a q_{01} output, but this will require two distinct states, rather than one. The effect of these changes is to bias the process towards the identity transduction. This means that randomly generated transducers will tend to assign a high probability to pairs of strings that are similar,

[1] An additional assumption is strictly speaking necessary, namely that every state reachable from the initial state can reach the final state with nonzero probability.

in the sense that they have small Levenshtein edit distance(Levenshtein, 1966) from each other.

The normalisation requirement for the output functions then becomes that for every state s

$$\sum_{c \in A} q_{11}(c|s) + q_{10}(c|s) + q_{01}(c|s) = 1$$

Before we discuss the algorithm for learning the models, it is worth illustrating how a model of this class could be used to represent a particular transduction. We shall consider an abstraction of one of the forms of the Arabic broken plural, namely the mapping from words with a skeleton of $C_1aC_2C_3$ to $C_1uC_2uwC_3$, where C_i represents a consonant; for example, from *bank* to *bunuwk*. The long vowels are represented as single symbols internally; so "uw" is represented by "U". This can be represented by a transducer with 6 emitting states. The transducer proceeds through the states in order – the transitions in this case do not depend on the data at all. The first state is a q_{11} output which will output the same consonant on the left and right. Next there is a q_{10} state which outputs an a on the left followed by a q_{01} state which outputs a u on the right. The fourth state is again a q_{11} output that copies the second consonant, followed by a q_{01} that produces the long U, and finally the sixth state copies the final consonant.

If we label the states $q_1, \dots q_6$, together with an initial state q_0 and a final state q_f, we can write $p(i, j)$ for the probability that the model will make a transition from state p_i to state p_j. Thus the transition probabilities of our simple model are $p(0, 1) = \dots p(i, i+1) = p(6, f) = 1$ and are zero otherwise, and if there are N consonants we can define the output functions to be for a consonant c $q_{11}(c|1) = q_{11}(c|4) = q_{11}(c|4) = 1/N$ and $q_{10}(a|2) = 1$ and $q_{01}(u|3) = q_{01}(U|5) = 1$ and zero otherwise. It is easy to see that this model defines a distribution over pairs of strings that is nonzero only for the N^3 pairs of the form $(C_1aC_2C_3, C_1uC_2uwC_3)$ where C_i is a consonant.

Though these models may be capable of representing these transductions, the important issue is whether there are algorithms capable of producing adequate models given data. The similarity of these models to Hidden Markov Models naturally suggests learning algorithms. While the general problem of learning HMMs is computationally hard (Abe & Warmuth, 1992), we can use a variant of the expectation-maximisation algorithm (Baum & Petrie, 1966) to train them. This requires an extension of the usual dynamic programming algorithms to work with all possible alignments of input and output strings (Casacuberta, 1995; Clark, 2001; Ristad & Yianilos, 1997).

We define the *forward* and *backward* probabilities as follows. Given two strings $u_1, \dots u_l$ and $v_1, \dots v_m$ we define the forward probabilities $\alpha_s(i, j)$ as the probability that it will start from s_0 and output $u_1, \dots, u_i$ on the left stream, and $v_1, \dots, v_j$ on the right stream and be in state s, and the backward probabilities $\beta_s(i, j)$ as the probability that starting from state s it will output $u_{i+1}, \dots, u_l$, on the right and $v_{j+1}, \dots, v_m$ on the left and then terminate, i.e. end in state s_1.

We can calculate these using the following recurrence relations:

$$\begin{aligned}\alpha_s(i,j) = &\sum_{s'} \alpha_{s'}(i, j-1)p(s|s')q_{01}(v_j|s)\\ &+ \sum_{s'} \alpha_{s'}(i-1, j)p(s|s')q_{10}(u_i|s)\\ &+ \sum_{s'} \alpha_{s'}(i-1, j-1)p(s|s')q_{11}(u_i, v_j|s)\end{aligned}$$

$$\begin{aligned}\beta_s(i,j) = &\sum_{s'} \beta_{s'}(i, j+1)p(s'|s)q_{01}(v_{j+1}|s')\\ &+ \sum_{s'} \beta_{s'}(i+1, j)p(s'|s)q_{10}(u_{i+1}|s')\\ &+ \sum_{s'} \beta_{s'}(i+1, j+1)p(s'|s)q_{11}(u_{i+1}, v_{j+1}|s')\end{aligned}$$

where, in these models, $q_{11}(u_i, v_j)$ is zero unless u_i is equal to v_j. Instead of the normal two-dimensional trellis discussed in standard works on HMMs, which has one dimension corresponding to the current state and one corresponding to the position, we have a three-dimensional trellis, with a dimension for the position in each string. With these modifications, we can use all of the standard HMM algorithms. In particular, we can use this as the basis of a parameter estimation algorithm using the expectation-maximization theorem. We use the forward and backward probabilities to calculate the expected number of times each transition is taken; at each iteration we set the new values of the parameters to be the appropriately normalized sums of these expectations.

This maximum likelihood training approach can be used to iteratively improve a model. This can form the basis of a learning algorithm by starting with a randomly initialised model which we then train. This approach to learning has a number of serious flaws, most importantly the fact that the likelihood surface is not convex and as a result there are numerous local maxima some of which may be very poor models. Thus with some simple models, for example probabilistic context free grammars, it has been known for some time that this combination is quite ineffective.

In our particular case we have further problems: since we are maximising the joint probability, it is not necessarily even the case that the global maximum does in fact model the transduction correctly. Still less is it necessarily the case that the local maximum we in fact find, which may or may not coincide with the global maximum will also model these transductions properly.

However we have found that in fact this approach works extremely well for simple transductions. Experimenting on the transduction ($C_1aC_2C_3 \rightarrow C_1uC_2UC_3$) with a 10 state model, we find that over 90% of the time the model converges to a model that defines the correct transduction. In a few pathological cases the model converges

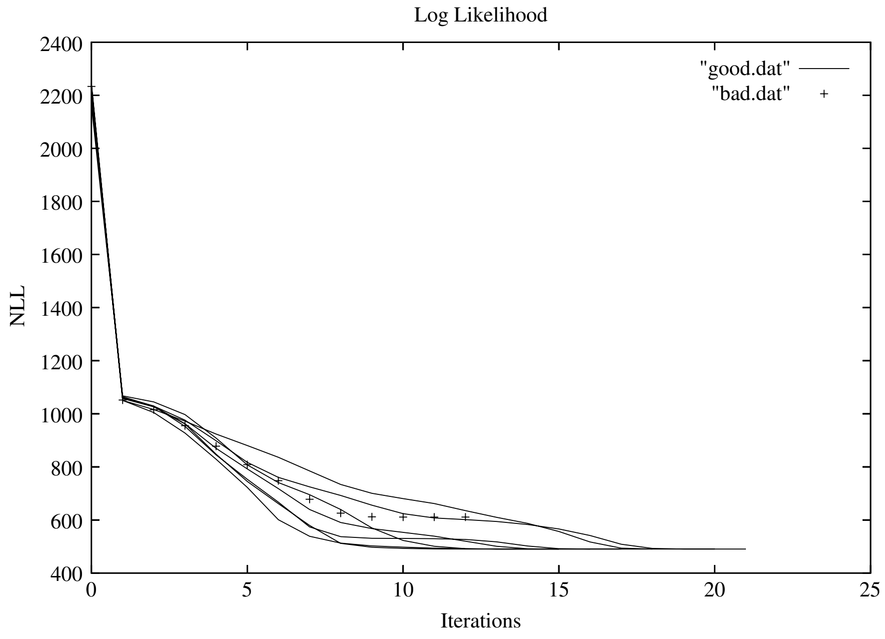


Fig. 10.1. Incorrect transduction (marked with crosses) ends up with significantly worse log likelihood than the correct transductions (solid lines). The figure shows negative log likelihood plotted against the number of iterations. (Many correct transductions have been removed to improve legibility.)

to a local maximum with a substantially lower log likelihood. On a data set of sixty examples, manually extracted from one of our data sets (PN) described below, the log likelihood of the model converges as shown in Figure 10.1.

The other component of a solution is to be able to do inference with the models: in this context this means that given a FST, and a string u, we need to find the string v that maximizes $p(u, v)$. This is equivalent to the task of finding the most likely string generated by a HMM, which is NP-hard (Casacuberta & de la Higuera, 2000), but it is possible to sample from the conditional distribution $p(v|u)$, which allows an efficient stochastic computation. If we consider only what is output on the left stream, the FST is equivalent to a HMM with null transitions corresponding to the q_{01} transitions of the FST. We can remove these using standard techniques and then use this to calculate the *left backward* probabilities for a particular string u: $\beta^L_s(i)$ defined as the probability that starting from state s the FST generates $u_{i+1}, \ldots, u_l$ on the left and terminates. Then if one samples from the FST, but weights each transition by the appropriate left backward probability, it will be equivalent to sampling from the conditional distribution of $P(v|u)$. We can then find the string v that is most likely given u, by generating randomly from $p(v|u)$. After we have generated a number of strings, we can sum $p(v|u)$ for all the observed strings; if the difference between this sum and 1 is less than the maximum value of $p(v|u)$ we know we have found the most likely v. In practice, the distributions we are interested in often have a v with $p(v|u) > 0.5$; in this case we immediately know that we have found the maximum.

The ability to model transductions is not enough though. A complete model must be able to decide which transduction to use. Having a very large model turns out not to perform well. This is partly because the training maximizes the joint likelihood, rather than the conditional likelihood. It is also extremly inefficient, since the number of parameters is quadratic in the number of states. One way to improve the efficiency is to use a mixture of models as discussed in (Clark, 2001), each corresponding to a morphological paradigm, or particular transduction. The productivity of each paradigm can be directly modelled, and the class of each lexical item can be parametrized. While this works after a fashion, there are a number of criticisms that could be made of this approach. First, many of the models produced merely memorize a single pair of strings, which is extremely inefficient. Secondly, although the model correctly models the productivity of some morphological classes, it models this directly. A more satisfactory approach would be to have this arise naturally as an emergent property of other aspects of the model. Thirdly, these models may not be able to account for some psycho-linguistic evidence that appears to require some form of *proximity* or similarity. Finally, getting adequate performance even on very simple models, requires a moderately complex smoothing algorithm. This could be alleviated by having a representation for the symbols that is based on phonological features, but in line with our knowledge-light approach, we would like to avoid this, particularly if the features are not purely acoustic.

In the next section we shall present a technique that addresses these problems.

10.3 Fisher Kernels and Information Geometry

The method used is a simple application of the information geometry approach introduced by Jaakkola & Haussler (1999) in the field of bioinformatics. The central idea is to use a generative model to define a finite-dimensional representation of a symbol sequence; in particular a representation that is sensitive to the properties of the data being modelled. Given a generative model for a string, one can use the sufficient statistics of those generative models as features. The vector of sufficient statistics can be thought of as a finite-dimensional representation of the sequence in terms of the model. This transformation from an unbounded sequence of atomic symbols to a finite-dimensional real vector is very powerful and allows the use of Support Vector Machine techniques for classification. (Jaakkola & Haussler, 1999) recommend that instead of using the sufficient statistics, that the Fisher scores are used, together with an inner product derived from the Fisher information matrix of the model. This has the satisfying consequence that the results are then invariant when the parametrization is changed. The Fisher scores are defined for a data point x and a particular model as

$$U_x^i = \frac{\partial \log p(x;\theta)}{\partial \theta_i} \tag{3}$$

The partial derivative of the log likelihood is easy to calculate as a byproduct of the E-step of the EM algorithm, and has the value for HMMs (Jaakkola, Diekhans, & Haussler, 2000) of

$$U_x^i = \frac{E[z_i|x]}{\theta_i} - E[s_j|x] \tag{4}$$

where z_i is the indicator variable for the parameter i, and s_j is the indicator value for the state j where z_i leaves state j; the last term reflects the constraint that the sum of the parameters must be one.

The kernel function is defined as

$$K(x,y) = U_x I_\theta^{-1} U_y \tag{5}$$

where I_θ is the Fisher information matrix.

This kernel function thus defines a distance between elements,

$$d(x,y) = (K(x,x) - 2K(x,y) + K(y,y))^{1/2} \tag{6}$$

This distance in the feature space then defines a pseudo-distance in the example space.

The name information geometry which is sometimes used to describe this approach derives from a geometrical interpretation of this kernel. For a parametric model with k free parameters, the set of all these models will form a smooth k-dimensional manifold in the space of all distributions. The curvature of this manifold can be described by a Riemannian tensor – this tensor is just the expected Fisher information for that model.

In spite of this compelling geometric explanation, there are difficulties with using this approach directly. First, the Fisher information matrix cannot be calculated directly, and secondly in natural language applications, unlike in bio-informatic applications we have the perennial problem of data sparsity, which means that unlikely events occur frequently. This causes the scaling in the Fisher scores to give extremely high weights to these rare events, which can skew the results. Accordingly this work uses the unscaled sufficient statistics.

The Fisher kernel can be compared to other string based kernels. An advantage of the Fisher kernel is that it can be sensitive to global properties of the strings, whereas the string kernels can only be sensitive to substrings.

10.3.1 Details

Given a transducer that models the transduction from uninflected to inflected words, we can extract the sufficient statistics from the model in two ways. We can consider the statistics of the joint model $p(u,v|\Theta)$ or the statistics of the conditional model $p(v|u,\Theta)$. Here we have used the conditional model, since we are interested primarily in the change of the stem, and not the parts of the stem that remain unchanged. It is thus possible to use either the features of the joint model or of the conditional model, and it is also possible to either scale the features or not, by dividing by the parameter value as in Equation 4. The second term in Equation 4 corresponding to the normalization can be neglected.

Based on the experiments reported previously (Clark, 2002) we have chosen the unscaled conditional sufficient statistics for the rest of the experiments presented here, which are calculated thus:

$$C_i((u, v)) = E[z_i|(u, v)] - E[z_i|u] \tag{7}$$

Given an input string u we want to find the string v such that the pair u, v is very close to some element of the training data. We can do this in a number of different ways. Clearly if u is already in the training set then the distance will be minimized by choosing v to be one of the outputs that is stored for input v; the distance in this case will be zero. Otherwise we sample repeatedly (here we have taken 100 samples) from the conditional distribution of each of the submodels. This in practice seems to give good results, though there are more principled criteria that could be applied.

We are using a k-nearest-neighbor rule with $k = 1$, since there are irregular words that have completely idionsyncratic inflected forms. It would be possible to use a larger value of k, which might help with robustness, particularly if the token frequency was also used, since irregular words tend to be more common.

In summary the algorithm proceeds as follows:

- We train a small FST on the pairs of strings using the EM algorithm.
- We derive from this model a distance function between two pairs of strings that is sensitive to the properties of this transduction.
- We store all of the observed pairs of strings.
- Given a new word, we sample repeatedly from the conditional distribution to get a set of possible outputs.
- We select the output such that the input/output pair is closest to one of the observed pairs.

10.4 Experiments

The data sets used in the experiments are summarized in Table 10.1. We have included a standard data set for a problem from English morphology to provide a comparison. This data (Ling, 1994) is drawn from the English past tense, a problem

Table 10.1. Summary of the data sets. LING is drawn from Ling (1994), PN from Plunkett & Nakisa (1997) and MP from McCarthy & Prince (1990)

Label	Language	Total Size	Train	Test
LING	English Past tense	1394	1251	140
PN	Arabic plural	859	773	86
MP	Arabic broken plural	3261	2633	293

that has exerted a fascination out of all proportion to its interest. It consists of pairs of strings of the base form and past form in UNIBET phonetic transcription. We use two Arabic data sets. The first is a data set prepared for Plunkett & Nakisa (1997). It consists of pairs of singular and plural nouns, in Modern Standard Arabic, randomly selected from the standard Wehr dictionary in a fully vocalized ASCII transcription. It has a mixture of broken and sound plurals, and has been simplified by removing forms of the broken plural that occurred below a certain frequency threshold. The second, prepared by McCarthy & Prince (1990) consists solely of broken plurals, with all variant plurals added.

10.4.1 Evaluation

We used 10-fold cross validation on all of these data sets. We compared the performance of the models evaluated using them directly to model the transduction using the conditional likelihood (CL) and using the MBL approach with the unscaled conditional features. Based on these results, we used the unscaled conditional features; subsequent experiments confirmed that these performed best.

The results are summarized in Table 10.2. Run-times for these experiments were from about 1 hour to 1 week on a standard workstation.[2] There are a few results to which these can be directly compared; on the LING data set, Mooney & Califf (1995) report figures of approximately 90% using a logic program that learns decision lists for suffixes. For the Arabic data sets, Plunkett & Nakisa (1997) do not present results on modelling the transduction on words not in the training set; however they report scores on the easier task of merely selecting the class of output of 63.8% (0.64%) using a neural network classifier. The data is classified according to the type of the plural, and is mapped onto a syllabic skeleton, with each phoneme represented as a bundle of phonological features. We have observed in further experiments that the MBL approach significantly outperforms the conditional likelihood method over a wide range of experiments. The performance on the training data is a further

Table 10.2. Results. CV is the degree of cross-validation, Models determines how many components there are in the mixture, CL gives the percentage correct using the conditional likelihood evaluation and MBLSS, using the Memory-based learning with sufficient statistics, with the standard deviation in brackets

Data Set	CV	Models	States	Iterations	CL	MBLSS
LING	10	1	10	10	61.3 (4.0)	85.8 (2.4)
	10	2	10	10	72.1 (2.0)	79.3 (3.3)
PN	10	1	10	10	0.6 (0.8)	15.4 (3.8)
	10	5	10	10	9.2 (2.9)	31.0 (6.1)
	10	5	10	50	11.3 (3.3)	35.0 (5.3)
MP	10	5	10	10	1.6 (0.6)	16.7 (1.8)

[2] 1 GHz Pentium processor.

difference, the MBL approach scoring close to 100%, whereas the CL approach scores only a little better than it does on the test data. It is certainly possible to make the conditional likelihood method work rather better than it does in this chapter by paying careful attention to convergence criteria of the models to avoid overfitting, and by smoothing the models carefully. In addition some sort of model size selection must be used. A major advantage of the MBL approach is that it works well without requiring extensive tuning of the parameters.

In terms of the absolute quality of the results, this depends to a great extent on how phonologically predictable the process is. When it is completely predictable, the performance approaches 100%; similarly a large majority of the less frequent words in English are completely regular, and accordingly the performance on EPT is very good. However in other cases, where the morphology is very irregular the performance will be poor. In particular with the Arabic data sets, the PN data set is very small compared to the complexity of the process being learned, and the MP data set is rather noisy, with a large number of erroneous transcriptions.

10.5 Semi-supervised Learning

We now move on from the task of learning from pairs of words, to a slightly less idealised task that approaches the problem of completely unsupervised learning. A number of different approaches to the unsupervised learning of morphology have been presented in the past few years (Goldsmith, 2001; Schone & Jurafsky, 2000). Though they achieve impressive results, they share a common failing: an *a priori* limitation on the form of the morphological transductions that can be modelled, restricted to simple concatenation, and often only suffixation. This is clearly undesirable, since many non-Indo-European languages, and some Indo-European ones as well, notably German, use other inflectional processes. A related limitation is that these approaches can only learn regular morphology. Though these systems perform well within their limitations, a general language learning system cannot make these sorts of assumptions. Supervised learning algorithms on the other hand, are capable of learning irregularities and non-concatenative morphology (Clark, 2001; Mooney & Califf, 1995; Rumelhart & McClelland, 1986). What is desirable is to have a means for turning a supervised acquisition model into an unsupervised acquisition model. In this section, we discuss a general algorithm for doing this, and show how this can be applied using the stochastic transducers we have been using. In this case we will not be using the Fisher kernel method.

This work is closely related to that of Yarowsky & Wicentowski (2000). They are concerned with integrating diverse sources of information; here we are concerned with the correct application of a single source of information, namely the surface forms of each word. As they say (Yarowsky & Wicentowski, 2000, p. 207):

> But for many languages, and to a quite practical degree, inflectional morphological analysis and generation can be viewed primarily as an alignment task on a broad coverage word list

Accordingly our approach is to use the stochastic model of the joint probability of the pair of strings, that we have used up to now, and to treat the alignment between the strings as a further hidden variable which can be modelled again with the EM algorithm. A clean and well-motivated treatment of this will allow integration of this into more complex and broader language acquisition systems. We do not present a solution to the general problem of unsupervised learning of morphology here: rather we consider a slightly easier task, that we call *partially supervised* learning: this is where the learner is presented with two sets of strings, and must work out what the relationship is between them.

10.5.1 Perfect Situation

We will start with an artificially simple situation. Let us suppose we have two sets of strings U and V of the same size. We wish to align them, i.e. find a bijection between them, and simultaneously train a model on the aligned data. We can model this as a stochastic process in two stages: first we generate n pairs of strings, and then we generate a permutation of the second set that shuffles them. We can model the permutation as a hidden variable X, that takes one of the $n!$ permutations as its value. We can consider this as an $n \times n$ permutation matrix such that $X_{ij} = 1$ if the ith element of U is aligned with the jth element of V and is zero otherwise.

$$p(U, V) = \sum_X p(X)p(U, V|X) \tag{8}$$

Since we have no reasons to prefer one permutation rather than another we set $p(X) = \frac{1}{n!}$. The probability given the alignment is just the product of the probabilities of the matching pairs,

$$p(U, V|X) = \prod_i p(u_i, v_{X(j)}) \tag{9}$$

If we consider the matrix P which has in i, j the element $p(u_i, v_j)$, the sum of the $n!$ permutations is called in linear algebra the *permanent* of the matrix (Bhatia, 1996).

$$Per(P) \overset{def}{=} \sum_\sigma \prod_i P_{i,\sigma(i)} \tag{10}$$

where σ ranges over all the permutations of n elements.

It is similar to the more familiar determinant but without the alternating signs. One problem is that it is not possible to calculate this efficiently (Barvinok, 1999). There is a great deal of active research in this area though as it is possible to encode various combinatorial problems into an appropriate matrix.[3]

[3] For example, finding the number of maximum matchings in a bipartite graph.

To train the model with the EM algorithm we need to be able to calculate the posterior expectation of X_{ij} given the data and the model. Since X_{ij} is one or zero, the expectation equals the probability that i is aligned with j.

$$p(X_{ij}|U,V) = \frac{\sum_{X:X_{ij}=1} p(U,V|X)}{\sum_X p(U,V|X)} \tag{11}$$

The denominator of this fraction is the permanent of the matrix P, i.e. the sum over all $n!$ permutations. The numerator is the sum of the $(n-1)!$ of those permutations that have $X_{ij} = 1$. This is the product of P_{ij} with the permanent of the ij-minor [4] of P.

Thus we can write

$$E[X_{ij}|U,V] = \frac{P_{ij} Perm(P^{ij})}{Perm(P)} \tag{12}$$

We can consider these posterior probabilities as a matrix, that will be doubly stochastic. This map from the matrix of probabilities to the matrix of posteriors is sometimes called the Bregman map (Bregman, 1967). Though intractable to compute exactly, we can approximate it under certain circumstances using the technique of Sinkhorn balancing (Sinkhorn, 1964), as advocated by Beichl & Sullivan (1999). This converges rapidly (Soules, 1991) giving a overall complexity of $O(n^3)$, which is tractable for matrices with dimensions ≈ 1000, such as we use here. The method of Sinkhorn balancing is intuitively quite straightforward; we want to scale a positive matrix so it is doubly stochastic. If we normalise the row sums, we will have a matrix that is row-stochastic, i.e. has its row sums equal to unity, but not necessarily column-stochastic. If we then normalise the column sums, we will have a matrix that is column-stochastic but probably not row-stochastic. If we continue in this way, alternating the normalisation of the rows and of the columns, we converge to a doubly stochastic matrix which under certain circumstances is an approximation to the Bregman map. We now have the basis for an algorithm.

1. Choose a random model.
2. Calculate the matrix of $p(u_i, v_j)$
3. Estimate the matrix of the posteriors, using Sinkhorn balancing.
4. Train the model on every pair (u_i, v_j) weighting by the value of the posterior probability.
5. Repeat from step 2, until the posterior probability matrix is (very close to) a permutation matrix.

Theoretically we know this will converge by the EM algorithm and empirically we observe that the matrix of posteriors rapidly converges to a permutation matrix.

[4] The ij-minor of a matrix is the matrix formed by removing the row and column containing the ij element, to form an $n-1$ by $n-1$ matrix.

10.5.2 Imperfection

This is obviously a highly artificial situation. More interesting cases are where we have two sets that are not the same size, with two subsets that we want to align. More formally we have two sets of strings U and V of size m and n respectively with subsets $U' \subset U$ and $V' \subset V$ both of size k. We then have three models. A model for U and a model for V and a model for a joint distribution over $U' \times V'$. We then have a hidden variable which corresponds to the selection of the sets and the bijection between U' and V'. Given a value of X we can write the total likelihood function as:

$$p(U, V|X) = \prod_{u \in U-U'} p_U(u) \prod_{v \in V-V'} p_V(v) \left(\prod_{u \in U', v \in V'} p_M(u, v) \right)^{\alpha} \tag{13}$$

We now have a certain degree of flexibility in how we define $p(X)$; we can make it depend on the size of the sets that are selected. We can use this to make some of the calculations more tractable.

This algorithm allows one to trade off the gain from aligning them against not aligning them, by including models for the information in the individual sequences (Allison, Powell, & Dix, 1999). We can tweak it if need be by raising the joint model probability to a power $\alpha \in [1, 2]$. This will have the effect of making it less likely to align words; if α is 1, then it is likely to align words even when they have little relation since $p_M(u, v)$ can in general be at least $p_U(u)p_V(v)$ for related u and v. Conversely, values of α close to 2, will mean that the model will only align the words if there is a very strong link between them. Thus α is a tunable parameter that allows us to adjust the recall/precision trade-off.

Suppose U has m elements, and V has n elements, then we have an $m \times n$ matrix of the joint probabilities, which we can call P_M We can also define a $m \times m$ diagonal matrix, corresponding to the probabilities according to the model of U, which we can call P_U, and a $n \times n$ diagonal matrix for the probabilities of V, P_V. If we also create a matrix of size $n \times m$ with every element 1, P_1 then we can form an $(m + n)$ square matrix thus:

$$M = \begin{pmatrix} P_U & P_1 \\ P_M & P_V \end{pmatrix} \tag{14}$$

Then every permutation of this matrix corresponds to a particular choice of the alignment, and we can use exactly the same techniques for estimating the posterior probabilities on this matrix, as we did before. There is one substantive difference which is that because of the block of ones in the top right, alignments that align k of U and V together will have a "bonus" factor of $k!$, corresponding to the $k!$ paths through the $k \times k$ submatrix of P_1. This is generally good, since we want to encourage the algorithm to align as much as possible. We can accommodate this formally by making $p(X)$ in our generative model be proportional to $k!$. We then

Table 10.3. Matrix of probabilities for U = {cat, dog, fox} and V = {cats, dogs}

$$\begin{pmatrix}
p_U(\text{cat}) & 0 & 0 & 1 & 1 \\
0 & p_U(\text{dog}) & 0 & 1 & 1 \\
0 & 0 & p_U(\text{fox}) & 1 & 1 \\
p_M(\text{cat,cats}) & p_M(\text{dog,cats}) & p_M(\text{fox,cats}) & p_V(\text{cats}) & 0 \\
p_M(\text{cat,dogs}) & p_M(\text{dog,dogs}) & p_M(\text{fox,dogs}) & 0 & p_V(\text{dogs})
\end{pmatrix}$$

train all three models, weighting the probabilities by the appropriate values from the posterior matrix.

A simple example from English will clarify: suppose $U = \{\text{cat}, \text{dog}, \text{fox}\}$ and $V = \{\text{cats}, \text{dogs}\}$. Table 10.3 shows the resulting composite matrix.

Now each permutation of this matrix will correspond to a particular alignment, and the probability given that alignment will be the product of the appropriate elements of the matrix. The identity matrix will correspond to none being aligned, and will have probability equal to the product of the elements along the diagonal of the matrix in Table 10.3, i.e.

$$\begin{aligned} p(U, V|X) = {} & p_U(\text{cat})p_U(\text{dog})p_U(\text{fox}) \\ & p_V(\text{cats})p_V(\text{dogs}) \end{aligned} \tag{15}$$

If *cat* and *dog* are aligned correctly then (k=2) there are 2!, matrices which are shown here

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{16}$$

and also

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{17}$$

each of which will have probability

$$p(U, V|X) = p_M(\text{cat,cats})p_M(\text{dog,dogs})p_U(\text{fox}) \tag{18}$$

10.5.3 Experiments with Arabic

We prepared two data sets from Plunkett & Nakisa (1997); first, we prepared a data set (PN1) merely by collecting all of the singulars into one set and all of the plurals into another. This is clearly unrealistically simple, and so we also prepared a data set by randomly removing half of the singulars and half of the plurals, to see whether the algorithm could correctly match up in the absence of the missing data. This produced a data set (PN2) of size 443/430, with 215 possible pairs to be aligned. The results of these three tests are summarised in Table 10.4. The experiments of the second data set were run with various values of the exponent α to see the effect of the exponent on precision and recall. As expected, the algorithm performed well on the initial data set, aligning with high accuracy and precision. On the imperfect data set, PN2, the effect of the exponent is quite marked. With the exponent at 1.0, the precision is very poor, but as the exponent is increased the precision increases rapidly. Given a larger data set, it would be possible to repeatedly apply the algorithm to the same data, removing at each time the pairs already aligned, thus potentially combining a number of high precision models into a high recall one.

The small number of states employed are for efficiency purposes, they are clearly too small to allow correct modelling of these processes.

There is very little work that is directly comparable: there has been no prior work on the unsupervised learning of Arabic. However we can compare the results in English to (Yarowsky & Wicentowski, 2000). They use a number of different sources of information, and have results ranging from 31.3% using only Levenshtein distance after 1 iteration to 99.2 % for the final model combining all sources of information including frequency and semantic information. De Roeck & Al-Fares (2000) present an algorithm for identifying Arabic roots, that uses a language specific distance function; we note also the work of Rogati, McCarley, & Yang (2003) who use a small aligned bilingual corpus to learn a stemmer. Similarly Yarowsky & Wicentowski (2000) use a weighted edit distance as one component of their model, also performing a Viterbi approximation to the EM reestimation; for particular languages it will always be able to perform well with a simpler algorithm using prior knowledge about the language in question. This is clearly not an option in cognitive modelling, since it must work with all languages without language-specific information.

The algorithms presented here are comparatively slow in their naive form since we have to compute all the elements of the matrix, so the complexity is $O(|U||V|)$. A simple optimisation could be used to avoid having to compute pairs that are obviously not related. Of course, *went* is radically different from *go*, and yet *went*

Table 10.4. Semi-supervised learning results. The second set of results on the PN2 data set show the effect the exponent has on the precision and recall

Data	States	α	U	V	Pairs	Correct	Incorrect	Precision	Recall
PN1	10	1.0	859	859	859	820	27	96.8	95.5
PN2	10	1.0	443	430	215	157	405	38.8	73.8
PN2	10	1.5	443	430	215	140	25	84.8	65.1
PN2	10	1.75	443	430	215	78	2	97.5	36.2
PN2	10	2.0	443	430	215	2	0	100.0	0.9

is the correct past tense, so this approach will introduce errors. Handling this sort of complete stem suppletion seems likely to require other sorts of information: frequency and semantic information are the obvious candidates. Since suppletion tends to occur infrequently and with very frequent words, these should suffice. This may allow an understanding of the prevalence of phonological transparency in natural languages.

10.6 Conclusion

We have discussed the learning of Arabic morphology in a knowledge-light framework, motivated primarily by the study of first language acquisition. The interesting non-concatenative broken plural is an excellent test bed for theories that aim to account for the acquisition of morphology. English is simply too trivial to be a suitable test bed, since almost any algorithm can work. Arabic requires more sophisticated techniques. We do not put this forward directly as a cognitive model, but only indirectly as an exemplar of a class of algorithms that may be capable of learning complex morphological transductions, and thus are at least potentially of explanatory value.

We have studied the learnability under two different assumptions. The first is where the learner is presented with a set of pairs of base and inflected form. This is implausibly easy, as the most that we can hope is for an early learner to be able to identify word classes (Clark, 2003). Accordingly we study also what we call partially or semi-supervised learning where the learner is presented with a pair of sets of words and must also align them. We show how stochastic transducers can be used to learn under both of these two situations, and how a memory based learning technique together with an information geometry based approach can enhance performance. The underlying unpredictability of the morphological processes that we study does mean that the final accuracy of the system is inevitably rather low, since the extent to which the choice of plural is phonologically specified is an upper bound on the performance.

Acknowledgements

I am grateful to Alexander Barvinok for pointing out to me the work of Beichl and Sullivan. I am also grateful to John McCarthy and Ramin Nakisa for allowing me to use their data. Thanks also to Bill Keller, Eric Gaussier and others for helpful comments. This work was originally done as part of the EU TMR network *Learning Computational Grammars*.

References

Abe, N., & Warmuth, M. K. (1992). On the Computational Complexity of Approximating Distributions by Probabilistic Automata *Machine Learning*, *9*, 205–260.

Allison, L., Powell, D., & Dix, T. I. (1999). Compression and Approximate Matching *The Computer Journal*, *42*(1), 1–10.

Barvinok, A. I. (1999). Polynomial time algorithms to approximate permanents and mixed discriminants within a simple exponential factor *Random Structures and Algorithms*, *14*, 29–61.

Baum, L. E., & Petrie, T. (1966). Statistical Inference for probabilistic functions of finite state Markov chains *Annals of Mathematical Statistics*, *37*, 1559–1663.

Beichl, I., & Sullivan, F. (1999). Approximating the Permanent via Importance Sampling with application to the dimer covering problem *Journal of Computational Physics*, *149*(1), 128–147.

Bhatia, R. (1996). *Matrix Analysis*. Berlin: Springer Verlag.

Bregman, L. M. (1967). Proof of Convergence of Sheleikhovskii's method for a problem with transportation constraints *Zh. vychsl. Mat. mat. Fiz.*, *147*(7).

Casacuberta, F. (1995). Probabilistic Estimation of Stochastic Regular Syntax-directed Translation Schemes In *Proceedings of the VIth Spanish Symposium on Pattern Recognition and Image Analysis*, pp. 201–207.

Casacuberta, F., & de la Higuera, C. (2000). Computational Complexity of Problems on Probabilistic Grammars and Transducers In Oliveira, A. L.(Ed.), *Grammatical Inference: Algorithms and Applications*, pp. 15–24. Berlin: Springer Verlag.

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

Clark, A. (2002). Memory-Based Learning of Morphology with Stochastic Transducers In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 513–520.

Clark, A. (2003). Combining Distributional and Morphological Information for Part of Speech Induction In *Proceedings of the tenth Annual Meeting of the European Association for Computational Linguistics EACL 2003*, pp. 59–66.

De Roeck, A. N., & Al-Fares, W. (2000). A Morphologically Sensitive Clustering Algorithm for Identifying Arabic Roots In *COLING-2000*, pp. 199–206.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, UK: Cambridge University Press.

Goldsmith, J. A. (2001). Unsupervised Learning of the Morphology of a Natural Language *Computational Linguistics*, *27*(2), 153–198.

Jaakkola, T. S., Diekhans, M., & Haussler, D. (2000). A discriminative framework for detecting remote protein homologies *Journal of Computational Biology*, *7*(1,2), 95–114.

Jaakkola, T., & Haussler, D. (1999). Exploiting generative models in discriminative classifiers In Kearns, M. S., Solla, S. A., & Cohn, D. A.(Eds.), *Advances in Neural Information Processing Systems 11*, pp. 487–493. San Mateo, CA. Morgan Kauffmann Publishers.

Kaplan, R. M., & Kay, M. (1994). Regular Models of Phonological Rule Systems *Computational Linguistics*, *20*(3), 331–378.

Kiraz, G. (1994). Multi-tape two-level morphology In *COLING-94*, pp. 180–186.

Koskenniemi, K. (1983). *A Two-level Morphological Processor*. Ph.D. thesis, University of Helsinki.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals *Soviet Physics Doklady*, *10*(8), 707–710.

Ling, C. X. (1994). Learning the Past Tense of English Verbs: The Symbolic Pattern Associator vs. Connectionist Models *Journal of Artifical Intelligence Research*, *1*, 209–229.

McCarthy, J., & Prince, A. (1990). Foot and Word in prosodic morphology: The Arabic Broken Plural *Natural Language and Linguistic Theory*, *8*, 209–284.

Mooney, R. J., & Califf, M. E. (1995). Induction of First-Order Decision Lists: Results on Learning the Past Tense of English Verbs *Journal of Artificial Intelligence Research*, *3*, 1–24.

Plunkett, K., & Nakisa, R. C. (1997). A Connectionist Model of the Arabic Plural System *Language and Cognitive Processes*, *12*(5/6), 807–836.

Ristad, E. S., & Yianilos, P. N. (1997). Finite Growth Models Tech.rep.CS-TR-533-96, Department of Computer Science, Princeton University. revised in 1997.

Rogati, M., McCarley, S., & Yang, Y. (2003). Unsupervised learning of Arabic stemming using a parallel corpus In *Proceedings of ACL*, pp. 391–398.

Rumelhart, D. E., & McClelland, J. L. (1986). On Learning Past Tenses of English Verbs In Rumelhart, D. E., & McClelland, J. L.(Eds.), *Parallel Distributed Processing*, Vol. 2, pp. 216–271. MIT Press, Cambridge, MA.

Schone, P., & Jurafsky, D. (2000). Knowledge-free induction of Morphology using Latent Semantic Analysis In *Proceedings of CoNLL-2000 and LLL-2000*, pp. 67–72. Lisbon, Portugal.

Sinkhorn, R. (1964). A relation between arbitrary positive matrices and doubly stochastic matrices *Annals of Mathematical Statistics*, *35*(2), 876–879.

Soules, G. W. (1991). The rate of convergence of Sinkhorn Balancing *Linear Algebra and Its Applications*, *150*(3), 3–40.

van den Bosch, A., & Daelemans, W. (1999). Memory-Based Morphological Analysis In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 285–292.

Yarowsky, D., & Wicentowski, R. (2000). Minimally Supervised Morphological Analysis by Multimodal Alignment In *Proceedings of ACL 2000*, pp. 207–216. Hong Kong.

11

Memory-based Morphological Analysis and Part-of-speech Tagging of Arabic

Antal van den Bosch[1], Erwin Marsi[1], and Abdelhadi Soudi[2]

[1] *ILK / Dept. of Language and Information Science, Faculty of Arts, Tilburg University, P.O. Box 90153, NL-5000 LE Tilburg, The Netherlands*
{Antal.vdnBosch,E.C.Marsi}@uvt.nl

[2] *Ecole Nationale de l'Industrie Minérale, Rabat, Morocco*
asoudi@gmail.com

Abstract: We explore the application of memory-based learning to morphological analysis and part-of-speech tagging of written Arabic, based on data from the Arabic Treebank. Morphological analysis is performed as a letter-by-letter classification task. Classification is performed by the k-nearest neighbor algorithm. Each classification produces a trigram of position-bound operations, each encoding segmentation, part-of-speech information, and letter transformations. The overlapping operation trigrams generated on the basis of an input word are converted into a lattice, from which all morphological analyses of the word are generated. Part-of-speech tagging is carried out separately from the morphological analyzer. A memory-based modular tagger is developed with a subtagger for known words and one for unknown words. On words not seen in training, the morphological analyzer attains a peak F-score of 0.47, while the tagger produces 66.4% correct tags. On all words, including words seen in training, the combination assigns a correct part-of-speech tag and generates all morphological analyses to about 91% of word tokens in running text

11.1 Introduction

Memory-based learning has been successfully applied to morphological analysis and part-of-speech tagging in Western and Eastern-European languages (Daelemans et al., 1996; Van den Bosch and Daelemans, 1999; Zavrel and Daelemans, 1999). With the release of the Arabic Treebank by the Linguistic Data Consortium, a large corpus has become available for Arabic that can act as training material for machine-learning algorithms. The data facilitates machine-learned part-of-speech taggers, tokenizers, and shallow parsing units such as chunkers (Diab, Hacioglu, and Jurafsky, 2004); cf. Chapter 9.

However, as argued and illustrated throughout this book, Arabic offers special challenges for data-driven and knowledge-based approaches alike. An Arabic word may be composed of a stem consisting of a consonantal root and a pattern, and may furthermore contain affixes and clitics. Arabic verbs, for instance, can be conjugated according to one of the traditionally recognized patterns. There are 15 triliteral forms, of which at least 9 are common. They represent very subtle differences. Within each conjugation pattern, an entire paradigm is found: two tenses (perfect and imperfect), two voices (active and passive) and five moods (indicative, subjunctive, jussive, imperative, and energetic). Arabic nouns show a comparably rich and complex morphological structure.

In this chapter we explore the use of memory-based learning for morphological analysis and part-of-speech (POS) tagging of written Arabic. The next section summarizes the principles of memory-based learning. Section 11.3 describes the data used throughout the study for both tasks. The subsequent three sections describe our work on memory-based morphological analysis (Section 11.4) and its integration with part-of-speech tagging (Section 11.5). The final Section 11.6 contains a short discussion of related work and offers an overall conclusion.

11.2 Memory-based Learning

Memory-based learning, also known as instance-based, example-based, or lazy learning (Aha, Kibler, and Albert, 1991; Daelemans, Van den Bosch, and Zavrel, 1999), based on the k-nearest neighbor classifier (Cover and Hart, 1967), is a supervised inductive learning algorithm for learning classification tasks. Memory-based learning treats a set of labeled (pre-classified) training instances as points in a multi-dimensional feature space, and stores them as such in an *instance base* in memory. Thus, in contrast to most other machine learning algorithms, it performs no abstraction, which naturally allows it to deal with productive but low-frequency exceptions (Daelemans, Van den Bosch, and Zavrel, 1999).

An instance consists of a fixed-length vector of n feature-value pairs, and an information field containing the classification of that particular feature-value vector. After the instance base is stored, new (test) instances are classified by matching them to all instances in the instance base, and by calculating with each match the *distance*, given by a distance function $\Delta(X, Y)$ between the new instance X and the memory instance Y. The most primitive distance function is the "overlap" function $\Delta(X, Y) = \sum_{i=1}^{n} w_i \ \delta(x_i, y_i)$, where n is the number of features, w_i is a weight for feature i, and $\delta = 0 \ if \ x_i = y_i, \ else \ 1$ is the distance per feature. In this chapter we use the somewhat more complex Modified Value Difference Metric (MVDM) distance function (Cost and Salzberg, 1993; Stanfill and Waltz, 1986), which determines the similarity of pairs of values of a feature by looking at the conditional probabilities (estimated on co-occurrence counts) of the two values conditioned on the classes: $\delta(v_1, v_2) = \sum_{i=1}^{n} |P(C_i|v_1) - P(C_i|v_2)|$.

In this chapter the weight (importance) of a feature i, w_i, is estimated by computing its *gain ratio*, GR_i. To compute a feature's GR, we first compute its *information gain* IG_i, which is the difference in uncertainty (entropy) within the set of cases between the situations without and with knowledge of the value of that feature: $IG_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v)$, where C is the set of class labels, V_i is the set of values for feature i, and $H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. The probabilities are estimated from frequency counts in the training set. To derive the GR, the feature's IG is divided by the entropy of the feature values, the *split info* $si_i = -\sum_{v \in V_i} P(v) \log_2 P(v)$: $GR_i = \frac{IG_i}{si_i}$.

Classification in memory-based learning is performed by the k-NN algorithm that searches for the k 'nearest neighbors' according to the $\Delta(X, Y)$ function. The majority class of the k nearest neighbors then determines the class of the new case. With symbolic feature values, distance ties can occur when two nearest neighbors mismatch with the test instance on the same feature value, while all three instances have different values. In the k-NN implementation[1] we used, equidistant neighbors are taken as belonging to the same k, so this implementation is effectively a k-nearest distance classifier. This implies that when $k = 1$, more than one nearest neighbor may be found, all at the same distance to the test instance.

11.3 Data

Our point of departure is the Arabic Treebank 1 (ATB1), version 2.0, distributed by LDC[2], more specifically the "after treebank" POS-tagged data, consisting of 166,068 tagged words. Tokens are vocalized and morphologically analyzed by means of Tim Buckwalter's Arabic Morphological Analyzer (Buckwalter, 2002). An example is given in Figure 11.1. The input token (`INPUT STRING`) is transliterated (`LOOK-UP WORD`) according to Buckwalter's transliteration system. All possible vocalizations and their morphological analyzes are listed (`SOLUTION`). The portion of the solution relevant to our experiments is the segmentation, with plus markers (+) as segment boundary markers, and slashes (/) as delimiters between the character strings and the part-of-speech information related to that string. For example, the segmented string `kutub/NOUN+a/CASE_DEF_ACC` represents the substring `kutub` being a noun, and `a` carrying the morpho-syntactic function `CASE_DEF_ACC` (underscores occur within multi-word tags, and have no relation with segmentation). The resulting analyses have been manually annotated with a preceding star ($\star$), marking the correct solution in the given context.

Throughout the corpus the number of analyses listed per word is not constant, probably as a result of additions and/or deletions by the annotators. As our goal is to predict all possible analyses for a given word, we first created a lexicon that maps every word to all analyses encountered and their respective frequencies; see Figure 11.2 for an example. In the course of this process, we removed all vowels, since we aim at analyzing unvoweled words, producing unvoweled analyses. Vowel generation ultimately implies stem identification, a complication which we do not address here; our goal is the segmentation of unvoweled strings and an appropriate

[1] All experiments with memory-based learning were performed with TiMBL, version 5.1 (Daelemans et al., 2004), available from `http://ilk.uvt.nl`.

[2] LDC: `http://www.ldc.upenn.edu/`.

```
      INPUT STRING: كتب

   LOOK-UPWORD: ktb
            Comment:
              INDEX:P2W20
       SOLUTION 1: (kataba) [katab-u_1] katab/PV+a/PVSUFF_SUBJ:3MS
            (GLOSS): write + he/it [verb]
     * SOLUTION 2: (kutiba) [katab-u_1] kutib/PV_PASS+a/PVSUFF_SUBJ:3MS
            (GLOSS): be written/be fated/be destined + he/it [verb]
       SOLUTION 3: (kutub) [kitAb_1] kutub/NOUN
            (GLOSS): books
       SOLUTION 4: (kutubu) [kitAb_1] kutub/NOUN+u/CASE_DEF_NOM
            (GLOSS): books + [def.nom.]
       SOLUTION 5: (kutuba) [kitAb_1] kutub/NOUN+a/CASE_DEF_ACC
            (GLOSS): books + [def.acc.]
       SOLUTION 6: (kutubi) [kitAb_1] kutub/NOUN+i/CASE_DEF_GEN
            (GLOSS): books + [def.gen.]
       SOLUTION 7: (kutubN) [kitAb_1] kutub/NOUN+N/CASE_INDEF_NOM
            (GLOSS): books + [indef.nom.]
       SOLUTION 8: (kutubK) [kitAb_1] kutub/NOUN+K/CASE_INDEF_GEN
            (GLOSS): books + [ indef.gen.]
       SOLUTION 9: (ktb) [DEFAULT] ktb/NOUN_PROP
            (GLOSS): NOT_IN_LEXICON
       SOLUTION 10: (katb) [DEFAULT] ka/PREP+tb/NOUN_PROP
            (GLOSS): like/such as + NOT_IN_LEXICON
```

Fig. 11.1. Example token from ATB 1 according to Buckwalter's transliteration (cf. Table 11.1 in Chapter 2)

identification of the morpho-syntactic function of each of the parts, so that the output can later be integrated with a part-of-speech tagger (see Section 11.5).

Also, we chose to re-attach clitic tokens (e.g. determiners and prepositions) to their host, storing them together as a single word form. The lexicon derived from the full corpus, excluding punctuation markers and words without a stored solution, contains 16,626 unique words and 113,105 analyses; an average of 6.8 analyses per word.

```
ktb
==> ktb/PV + /PVSUFF_SUBJ:3MS + 7
==> k/PREP + tb/NOUN_PROP + 7
==> ktb/PV_PASS + /PVSUFF_SUBJ:3MS + 7
==> ktb/NOUN + /CASE_DEF_ACC + 7
==> ktb/NOUN_PROP + 7
==> ktb/NOUN + /CASE_DEF_NOM + 8
==> ktb/NOUN + K/CASE_INDEF_GEN + 7
==> ktb/NOUN + 7
==> ktb/NOUN + N/CASE_INDEF_NOM + 7
==> ktb/NOUN + /CASE_DEF_GEN + 8
```

Fig. 11.2. Example of a preprocessed lexical entry for the word ktb كتب, carrying ten morphological analyses, using Buckwalter's transliteration (cf. Table 11.1 in Chapter 2)

To evaluate our system, we need data which can be regarded as realistic test material, including a typical amount of unknown words, representing any new document of text the system may be applied to. More realistically, we require any news article of the type that the ATB 1 corpus is composed of. To this purpose, we split the complete part-of-speech tagged ATB 1 corpus into eleven partitions of near-equal size. The data is shuffled randomly at the article level. One of these eleven partitions is set apart as a held-out set for later use, described further below. Subsequently, ten pairs of training and test sets are generated using the ten remaining 1/11th partitions. Each training set consists of a concatenation of nine of the ten partitions, while each tenth partition is taken as the test set accompanying the training set. Repeating this ten times, we thus create ten overlapping training sets that each consist of 9/11th of the original ATB 1 tagged corpus, and ten corresponding non-overlapping test sets that each represent a 1/11 portion of the original ATB 1 corpus. With this 10-fold cross-validation setup, we use the same training-test set partitions for training both the morphological analysis system as well as the part-of-speech tagger described in the next section.

The eleventh held-out set, the remaining 1/11th portion of the shuffled ATB 1 corpus, is used in both experimental sections to estimate the generalization performance of both modules individually, and also to estimate the joint generalization performance of the part-of-speech tagger and morphological analyzer together. We use this single held-out split for methodological reasons; both the morphological analyzer and the part-of-speech tagger are tested using a range of algorithmic parameter values on the 10-fold partitionings (e.g., the value of k of the k-nearest neighbor classifier). Since an estimated generalization performance of both modules cannot be based on the optimized test performance in a range of experiments, a fully unknown test set is needed - hence the eleventh set.

We conclude the section with some lexical statistics. Taken separately, each 1/11th partition contains about 15,100 tokens and about 4,010 unique words. The ten 9/11th training sets on average contain about 135,900 tokens, and about 15,100 unique words. Importantly, the number of unknown word tokens in the ten test sets that do not occur in their respective training sets is 977 on average, most of them occuring once (76%): about 21% of all unique words (i.e., word types). Since most unknown words have a low frequency, they only represent about 6.5% of all tokens in a test set. On the difference between word tokens and word types, the following should be noted. As will be argued in the subsequent sections, we define morphological analysis as a task on word types, and part-of-speech tagging as a task on word tokens. Henceforth, we will refer to "words" when we mean word tokens, and "word types" when we mean word types.

11.4 Morphological Analysis

The goal of the memory-based morphological analysis system we describe here is to generate no more and no less than all possible morphological analyses of an unvoweled input word that has not occurred before in the analyzer's training material.

For all occurrences of a word we want to generate the same analyses; hence, this is a task at the word type level. An analysis consists of a proper segmentation of clitics, stems, and suffixes, brought to their canonical orthographic form, and with the correct identification of the morpho-syntactic part-of-speech tag of each segment.

This means our system is literally a morphological analysis generator for word types. To measure its ability to generate no more and no less than all possible analyses, we employ the dual concepts of precision (the percentage of generated analyses that is actually correct) and recall (the percentage of target analyses that the analyzer was able to generate). Also, as said before, we focus solely on the capability of the analyzer to generate analyses for word types it has not seen before, henceforth referred to as *unknown words*. We assume that a typical morphological analysis system has a lexicon at hand, allowing the system to reproduce all morphological analyses of known word types flawlessly. The problem is, of course, that typically a non-trivial portion of all words in a text are unknown words. In our experiments an unknown set of texts contains about 6.5% unknown words, or 24% of the word types.

In this section we first describe how we created the data used in our experiments. We then describe the experiments performed on these data, focusing our analyses on the precision and recall scores on unknown words. We conclude the section with a critical discussion of our results.

The experiments on training a morphological analysis system need an additional processing step, namely the extraction of a lexicon from each training set. Each lexicon contains for each word type in the training set all possible morphological analyses. This is done as described above. The same procedure is followed for the test set, except that here we ignore the words already in the training set; we focus on unknown words only. Hence, each test set is converted to a small lexicon of all unknown words that do not occur in the corresponding training set, listing for each unknown word the one or more analyses which they actually get in the annotated corpus. The goal of our morphological system is to generate precisely these analyses.

11.4.1 Creating Instances for Morphological Analysis

The lexical entries generated by preprocessing both the training set and the test set of each partition are converted to instances suitable to memory-based learning of the mapping from words to their analyses (Van den Bosch and Daelemans, 1999). Instances represent input (the orthographic word) and their corresponding output (the morphological analysis). Since instances need to be of a fixed length and since they need to be general enough to generalize from known to unknown words, instances do not map entire words to entire analyses (which would render them case-specific), but rather represent partial fixed-width snapshots of words mapping to subsequences of the analysis. More specifically, the mapping is broken down into smaller letter-by-letter classification tasks.

The input of each instance, consisting of a fixed number of *features*, is created by sliding a window over the input word, resulting in one instance for each letter.

Using a 5–1–5 window yields 11 features, i.e. the input letter in focus, plus the five preceding and five following letters. The equality mark (=) is used as a filler symbol for positions beyond the beginning or end of the word. To illustrate this, consider the seventh analysis of the word ktb كتب (the notion of writing) in Figure 11.2, `ktb/NOUN+K/CASE_INDEF_GEN+`, representing a stem (`ktb`, noun = kutub "books"), followed by a suffix (`K`) carrying the `CASE_INDEF_GEN` function. The `K` suffix is not realized in the surface form. This three-letter word with this particular analysis then results in the three instances displayed in Figure 11.3.

The class of the first instance, `_-/NOUN-/NOUN`, is a trigram (with the character `-` as the delimiter), marking the fact that the letter `k` is the first letter of a noun stem, and that the second letter `t` is also inside the same noun stem. The class of the second instance, `/NOUN-/NOUN-DK/NOUN/CASE_INDEF_GEN`, signals in the rightmost part of the trigram that the third letter, `b`, marks the end of the noun stem and also carries the unrealized `CASE_INDEF_GEN` function; at the same time, the `DK` code denotes that a `K` was deleted. Hence, to reconstruct the analysis, a `K` needs to be reinserted. In general, the class codes making up the three parts of the class trigram always encode the part-of-speech tag of the morpheme the corresponding letter belongs to. Optionally this tag is preceded by a code representing the insertion, deletion, or replacement of one or more letters that are required to generate the underlying forms of the morphemes in the analysis, coded by the letters `I`, `D`, and `R`, respectively, followed by the letters themselves. Segmentation is encoded implicitly; whenever a letter is associated to another part-of-speech tag than its preceding neighbor, then a morpheme segmentation boundary exists between them. Based on this complex information, a complete analysis can be constructed.

In principle, unigram classes could already be used for this purpose. If predicted correctly, an analysis would follow from the straightforward concatenation of position-specific classes. However, we are faced with an average of 6.8 analyses per word. The example word ktb كتب has ten analyses, as illustrated in Figure 11.2. The first letter is associated within these ten analyses with five different unigram classes: `PV`, `PREP`, `PV_PASS`, `NOUN`, and `NOUN_PROP`. The second letter is linked to four tags: `PV`, `NOUN`, `PV_PASS`, and `NOUN_PROP`. Finally, the third letter is associated with seven different tags. If predicted in isolation, the system would have the task to pick the correct ten analyses from the maximal cartesian product of $5 \times 4 \times 9 = 180$ combined analyses. Due to their redundant overlap, the trigrams offer the key to this search. Since the *k*-nearest distances classifier is used, it will always produce all ambiguous analyses at the same distance. Hence, all 5, 4, and 9 position-specific overlapping trigrams will be present in the

```
= = = = = k t b = = =   _-/NOUN-/NOUN
= = = = k t b = = = =   /NOUN-/NOUN-DK/NOUN/CASE_INDEF_GEN
= = = k t b = = = = =   /NOUN–DK/NOUN/CASE_INDEF_GEN-_
```

Fig. 11.3. Instances for the analyses of the word ktb كتب in Figure 11.2

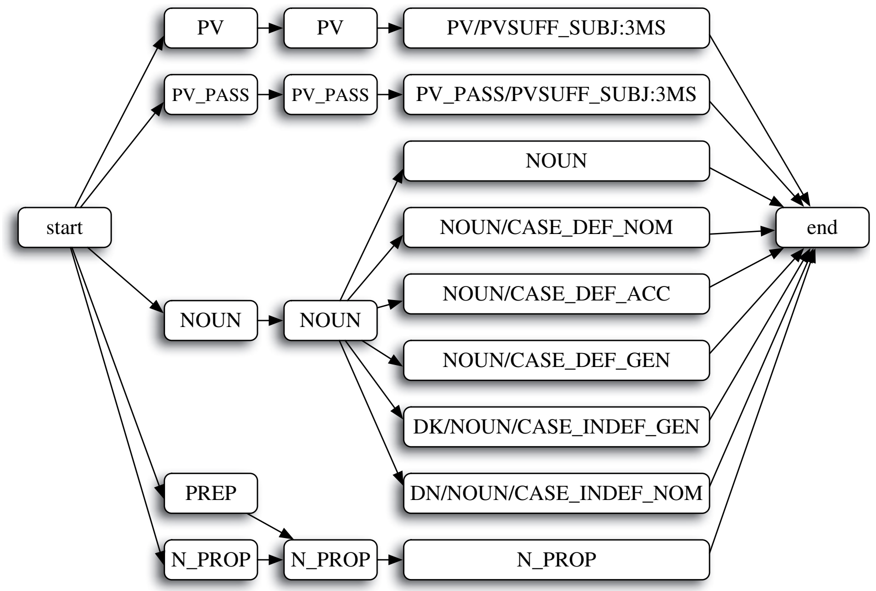


Fig. 11.4. The lattice formed by the overlapping trigrams generated by three consecutive position-specific classifications, using all nearest neighbors at distance $k = 1$. The lattice contains exactly the ten possible paths encoding the ten morphological analyses of ktb كتب

ideal case that the classifier maps to the correct nearest neighbors. The trigrams then span up a lattice, illustrated in Figure 11.4, in which exactly ten paths are possible.

While this example is perfect, less complete lattices are possible when a trigram does not match with one of the trigrams generated on the next letter. This will never occur with known words, but as said, we focus on unknown words, and then it is quite possible that a nearest-neighbor classification of the first letter of the word yields a completely different trigram than the nearest-neighbor classification of the second word. In these cases, paths in the lattice do not connect to the end node since they are generated by a trigram classification that does not match on the right hand side, or do not start in the start node since they are generated by a trigram that does not match on the left hand side. We apply the hard constraint that a valid path, encoding a complete analysis, must lead from the start node to the end node; hence, a mismatching trigram generates at least one invalid path. This also means that completely mismatching trigrams may lead to a lattice without valid paths, and no morphological analysis is generated.

Figure 11.5 displays an actual lattice generated for the unknown word AHtfAl احتفال (celebration); eight of the 24 intermediate nodes are part of invalid paths. Two of these paths do not connect to either the start or the end node; these are verbal analyses generated while focusing on the fourth letter, f ف. Eventually, six paths

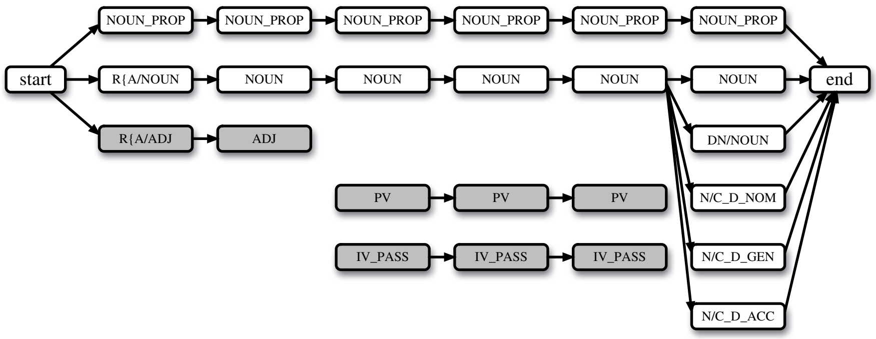


Fig. 11.5. The lattice formed by the overlapping trigrams generated by six consecutive position-specific classifications, using the five *k*-nearest distances, generated for the unknown word AHtfAl احتفال (celebration). Grey nodes in the lattice represent nodes in invalid paths, i.e. they are not used for the generation of analyses. "N/C_D" stands for `NOUN/CASE_DEF`

are valid, i.e., six analyses can be generated. Five of them are correct; the analysis with `DN/NOUN` misses a `CASE_INDEF_NOM` tag, and there is also an analysis with `CASE_INDEF_GEN` that cannot be generated from this lattice.

11.4.2 Evaluation

Performance is evaluated on the generated complete analyses (i.e. all valid paths in a generated lattices), where an analysis is considered complete if all of its part-of-speech and spelling change information is fully correct. Note again that the analyses concerns word types; the morphological analyzer is trained on word types from the training set, and is applied to unknown word types in the test set. Since we are concerned with measuring the degree of undergeneration or overgeneration of analysis generation, we quantitatively measure the performance of our system in terms of precision, recall, and F-score (Van Rijsbergen, 1979). Precision (1), the ratio of True Positives (TP) over the total number of true and False Positives (FP), measures to what extent overgeneration of analyses occurs, whereas recall (2), the ratio of true positives over the sum of true positives and False Negatives (FN), measures the amount of undergeneration. The F-score (3), the harmonic average between precison and recall, represents an overall fit between real and predicted analyses.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

$$F\text{-}score = \frac{2 * precision * recall}{precision + recall} \tag{3}$$

11.4.3 Experiments

We performed 10-fold cross-validation experiments, measuring the precision and recall on the generated analyses for the unknown words in the test sets. Figure 11.6 displays the precision-recall curve with different values of the k-parameter of the k-nearest neighbor classifier. Starting at $k = 1$, when the nearest-neighbor classifier just uses the nearest-distance instances in memory to generate the analysis, a precision level of 0.70 is attained; this means that about seven out of ten generated analyses are correct. The downside is that at $k = 1$, recall is only 0.29, meaning that the system is only able to generate slightly under one in three of the analyses it should generate. Recall can be increased to 0.42 with higher values of k, peaking at $k = 10$ (precision at that point is down to 0.40), but slightly decreasing with $k > 10$. The peak F-score of 0.47 (the highest harmonic mean of precision and recall) is attained at $k = 3$.

Clearly this performance is not impressive. We have to keep in mind, however, that the task is not an easy one: the evaluation concerns *unknown* words that were not in the learner's training material.

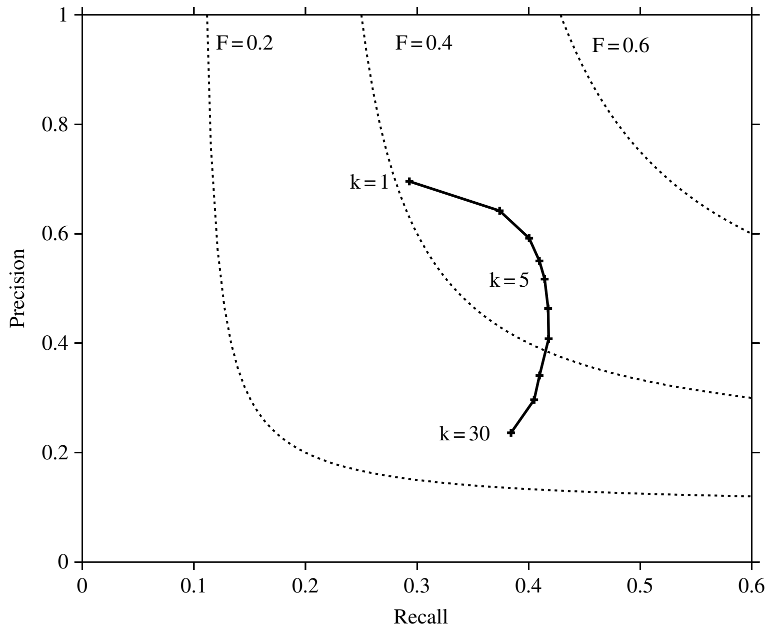


Fig. 11.6. Precision-recall curve on the generation of morphological analyses, with increasing values of k, from 1 up to 30. The F-isolines represent the lines in the space with a particular value of F in steps of 0.2

11.5 Integration with Part-of-speech Tagging

We employ MBT, a memory-based tagger-generator and tagger (Daelemans et al., 1996) to produce a Part-of-Speech (POS) tagger based on the ATB1 corpus.[3] We first describe how we prepared the corpus data. We then describe how we generated the tagger (a two-module tagger with a module for known words and one for unknown words), and subsequently we report on the accuracies obtained on test material by the generated tagger.

11.5.1 Data Preparation

While the morphological analyzer attempts to generate all possible analyses for a given word, the goal of POS tagging is to select one of these analyses as the appropriate one given the context, as the annotators of the ATB1 corpus did manually with the $\star$ marker. We developed a POS tagger that is trained to predict, for any given word, a concatenation of the POS tags of its morphemes. This is essentially the morphological analysis of a word in which segmentation information and letter transformations are discarded. Figure 11.7 shows part of a sentence where for each word the respective tag is given in the second column.

Concatenation is encoded by the delimiter +. Some words have no solution at all, but annotator comments usually tag them as proper nouns – hence we gave all of these words a NOUN_PROP tag.

Due to the concatenation of all morpho-syntactic information, the tag set is quite substantial: 306 different tags occur in the ten folds extracted from the ATB1 corpus. The five most frequent tags are PREP (13%), PUNC (10%), NOUN_PROP (7%), CONJ (4%), and DET+NOUN+CASE_DEF_GEN (4%). About 10% of the tags occur

w	CONJ
bdA	VERB_PERFECT
styfn	NOUN_PROP
knt	NOUN_PROP
nHylA	ADJ+NSUFF_MASC_SG_ACC_INDEF
jdA	ADV
,	PUNC
AlA	ADV
>n	FUNC_WORD
h	PRON_3MS
Agtsl	VERB_PERFECT
w	CONJ
...	

Fig. 11.7. Part of an ATB1 sentence with words (left) and their respective POS tags (right), according to Buckwalter's transliteration (cf. Table 11.1 in Chapter 2)

[3] In our experiments we used the MBT software package, version 2 (Daelemans et al., 2003), available from http://ilk.uvt.nl/.

only once (and will therefore never be predicted correctly), and about 33% of all tags occur less than 10 times.

We used the same partitionings of the ATB1 corpus as used to develop the morphological analyzer; we also run a 10-fold cross validation experiment. Evaluation differs, though: we measure the percentage of word tokens that are given the correct tag, i.e. a straightforward accuracy score. We also split this score into one on known words (already encountered in the training data) and unknown words (not encountered in the training data, i.e., the same words as focused on when evaluating the output of the morphological analysis system). Different from morphological analysis, we cannot assume that we are able to perform perfectly on words that are already known from a training set; in part-of-speech tagging, having seen a word in training only means that the tagger knows some of the tags (hopefully, all of the tags) that a word may have in different contexts. The tagger still needs to decide which tag is appropriate for every word token.

11.5.2 Memory-based Tagger Generator

Memory-based tagging is based on the idea that words occurring in similar contexts will have the same POS tag. A particular instantiation, MBT, was proposed by Daelemans et al. (1996). MBT has three modules. First, it has a lexicon module which stores for all words occurring in the provided training corpus their possible POS tags (tags which occur below a certain threshold, default 5%, are ignored). Second, it generates two distinct taggers; one for known words, and one for unknown words. The known-word tagger can obviously benefit from the lexicon, just as a morphological analyzer could. If a word has one unique tag in the training set, it will likely have the same tag when reoccurring in test material. If a word has a handful of possible tags, then the tagger's task is reduced to selecting the appropriate one from this limited list.

The input on which the known-word tagger bases its prediction for a given focus word consists of the following set of features and parameter settings: (1) The word itself, in a local context of the two preceding words and one subsequent word. Only the 200 most frequent words are represented as themselves; other words are reduced to a generic string – cf. (Daelemans et al., 2003) for details. (2) The possible tags of the focus word, plus the possible tags of the next word, and the *disambiguated* tags of two words to the left (which are available because the tagger operates from the beginning to the end of the sentence). The known-words tagger is based on a k-NN classifier with $k = 15$, the Modified Value Difference Metric (MVDM) distance function, inverse-linear distance weighting, and GR feature weighting. These settings were manually optimized on one of the test partitions.

The unknown-word tagger obviously does not know the possible tags the unknown word can have. Instead, it attempts to derive as much information as possible from the surface form of the word, by using its suffix and prefix letters as features. The following set of features and parameters are used: (1) The three prefix letters and the four suffix letters of the focus word (possibly encompassing the whole word); (2) The possible tags of the next word, and the *disambiguated* tags of two words to the left. The unknown-words tagger is based on a k-NN classifier with $k = 19$, the Modified Value Difference Metric (MVDM) distance function, inverse-linear distance weighting, and GR feature weighting – again, manually tuned on one test set.

11.5.3 Evaluating the Tagger

Table 11.1 lists the average accuracy (percentage of correctly tagged test words) of MBT as measured in the 10-fold cross-validation experiment. The general accuracy, 91.5%, is reasonable; known words are tagged with an accuracy of 93.3%. The unknown-words tagger has a lower accuracy than the known-words tagger, but it is able to tag 66.4% of the 6.5% unknown test words correctly nevertheless.

11.5.4 Integrating Morphological Analysis and Part-of-speech Tagging

While morphological analysis and POS tagging are ends in their own right, the usual function of the two modules in higher-level natural language processing or text mining systems is that they jointly determine for each word in a text the appropriate single morpho-syntactic analysis. In our setup, this amounts to predicting the solution that is preceded by "$\star$" in the original ATB1 data. For this purpose, the POS tag predicted by MBT, as described in the previous section, serves to select the morphological analysis that is compatible with this tag.

As a concluding experiment, we trained MBT with optimized settings on the complete concatenated ten folds of our original experiment, and tested the tagger on the eleventh held-out partition. As can be expected with somewhat more training data available, we observe a slightly higher overall tagging accuracy on the held-out data of 92.0%, with 93.4% on known words, and 71.0% on the 672 unknown words in this partition.

Subsequently we also trained a morphological analyzer on the full concatenated ten folds, and tested it on the held-out set, with the $k = 10$ setting that yielded the highest recall in the 10-fold cross-validation experiment. Recall is more important than F-score or precision, since higher recall will generally improve the likelihood of a match with the part-of-speech tags. Training the analyzer on the full ten folds and testing on the held-out set yields a precision of 0.41, a recall of 0.43, and an F-score of 0.42.

Table 11.1. POS tagging accuracies on known words, unknown words, and all test data (% correctly tagged test words), with standard deviations

Accuracy (% correct POS tags)		
Known words	Unknown words	All words
93.3 ±0.6	66.4 ±2.1	91.5 ±0.6

Finally, we computed how well the generated morphological analyses for unknown words could be integrated with the predicted tags for these words. We first computed an upper bound score by comparing the generated analyses against the gold standard annotated tags of all unknown words. It turned out that in 77.5% of all unknown words one of the analyses generated actually matches with the correct gold-standard part-of-speech tag. To illustrate this, consider the unknown word *yxfy*, which is tagged as `IV3MS+IV+IVSUFF_MOOD:I`. The three analyses generated for this word, reduced to only the concatenation of the part-of-speech tags (leaving out letter transformations and the specific segmentation position information) are

1. `NOUN_PROP`
2. `IV3MS+IV_PASS`
3. `IV3MS+IV+IVSUFF_MOOD:I`

Since one of the analyses matches the gold-standard tag, this word is counted as one of the 69.1% of the unknown words of which the full analysis could be generated, i.e., of which the solution marked with a $\star$ in the ATB1 corpus could be identified.

The actual agreement with the predicted tag is 75.0%, which is also high; however, the most relevant score is the intersection between the cases where both the tagger is correct, and one of the morphological analyses matches with the tag. In 82.2% of the cases in which the predicted tag is correct, a matching morphological analysis is also generated. Measured at the level of all unknown words, including the words that receive an incorrect tag, we find that of all unknown words in the held-out set 58.1% are assigned both a correct tag and a completely correct and matching morphological analysis, including segmentation and letter transformations.

11.6 Discussion and Conclusion

In this chapter we investigated the application of memory-based learning (*k*-nearest neighbor classification) to morphological analysis and POS tagging of written Arabic, using the ATB1 corpus as training and testing material. The morphological analyzer, when optimized on recall, was shown to attain a precision of 0.41, a recall of 0.43, and an F-score of 0.42 on *unknown* word types in held-out data when predicting all aspects of the analysis: part-of-speech tags of the segments, the positions of the segmentations, and all letter transformations between the surface form and the analysis. The POS tagger, in turn, attained an accuracy of 66.4% on unknown words, and 91.5% on all words (including known words) in held-out data. A combination of the two which selects one full morpho-syntactic analysis out of the generated analyses, matching the part-of-speech predicted by the tagger, yields a joint accuracy of 58.1% fully correctly predicted tags and corresponding full analysis for unknown words.

Extrapolating this number to a score on all words in a text, i.e. including the known words, we assume (safely) that our training-set-based lexicon always produces a matching analysis for all known words, for which we have observed the 93.3% accuracy of the part-of-speech tagger. Since known words make up 93.5% of test data, on average, we estimate that we can generate correct tags with complete morphological analyses for $(0.935 \times 0.933) + (0.065 \times 0.581) = 91.0\%$ of all words in unseen text.

The application of machine learning methods to Arabic morphology and POS tagging appears to be somewhat limited and recent, compared to the vast descriptive and rule-based literature particularly on morphology (Beesley, 1990; Beesley, 1998; Kay, 1987; Kiraz, 1994; Soudi, 2002). We are not aware of any machine-learning approach to Arabic morphology. POS tagging, on the other hand, seems to have attracted some focus. Freeman (2001) describes initial work in developing a POS tagger based on transformational error-driven learning (i.e. the Brill tagger), but does not provide performance analyses. Khoja (2001) reports a 90% accurate morpho-syntactic statistical tagger that uses the Viterbi algorithm to select a maximally-likely part-of-speech tag sequence over a sentence. In Chapter 9 of this book, Diab et al. describe a part-of-speech tagger based on support vector machines that is trained on tokenized data, reporting a tagging accuracy of 95.5%.

The use of trigrams in the output space has been proposed by Van den Bosch and Daelemans (2006), and demonstrated on morphological analysis of Dutch and English words by Van den Bosch, Schuurman, and Vandeghinste (2006). In this chapter, trigram classes are used differently, however; here, the problem is not only in optimizing the class label prediction, but also in limiting the overgeneration of analyses.

We make two final remarks. First, memory-based morphological analysis of Arabic words is feasible, but its main limitation is its inability to recognize the stem of an unknown word, and consequently the appropriate vowel insertions. Also, its guess on the possible POS tags of an unknown word turned out to be less useful in our tagging approach than using the raw prefix and suffix letters of the words themselves, as witnessed by the scores on unknown words of the POS subtagger specialized in unknown words. Second, memory-based POS tagging of written Arabic text appears to be successful, because performance is comparable to that for other languages. The POS tagging task as we define it, is deliberately separated from the problem of vowel insertion, which is in effect the problem of stem identification. We therefore consider the automatic identification of stems as a component of full morpho-syntactic analysis of written Arabic an important issue for future research.

Acknowledgements

The work of the first two authors is funded by the Netherlands Organisation for Scientific Research (NWO). The authors wish to thank Walter Daelemans and Sander Canisius for discussions and feedback.

References

Aha, D. W., D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Beesley, K. 1990. Finite-state description of Arabic morphology. In *Proceedings of the Second Cambridge Conference: Bilingual Computing in Arabic and English*, page no pagination.

Beesley, K. 1998. Consonant spreading in Arabic stems. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Montréal, Quebec, Canada*, pp. 117–123.

Buckwalter, T. 2002. Buckwalter Arabic morphological analyzer version 1.0. Technical Report LDC2002L49, Linguistic Data Consortium. available from: `http://www.ldc.upenn.edu/`.

Cost, S. and S. Salzberg. 1993. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

Cover, T. M. and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

Daelemans, W., A. Van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.

Daelemans, W., J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I. Dagan, editors, *Proceedings of the Fourth Workshop on Very Large Corpora*, pp. 14–27. ACL SIGDAT.

Daelemans, W., J. Zavrel, A. Van den Bosch, and K. Van der Sloot. 2003. MBT: Memory based tagger, version 2.0, reference guide. Technical Report ILK 03-13, ILK Research Group, Tilburg University.

Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2004. TiMBL: Tilburg Memory Based Learner, version 5.1.0, reference guide. Technical Report ILK 04-02, ILK Research Group, Tilburg University.

Diab, M., K. Hacioglu, and D. Jurafsky. 2004. Automatic tagging of arabic text: From raw text to base phrase chunks. In *Proceedings of HLT-NAACL 2004*, pp. 149–152, Boston, MA.

Freeman, A. 2001. Brill's POS tagger and a morphology parser for Arabic. In *ACL/EACL-2001 Workshop on Arabic Language Processing: Status and Prospects*, Toulouse, France. Available on: `http://www.elsnet.org/acl2001-arabic.html`.

Kay, M. 1987. Non-concatenative finite-state morphology. In *Proceedings of the third Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2–10, Copenhagen, Denmark.

Khoja, S. 2001. APT: Arabic part-of-speech tagger. In *Proceedings of the Student Workshop at NAACL-2001*, pp. 20–25.

Kiraz, G. 1994. Multi-tape two-level morphology: A case study in semitic non-linear morphology. In *Proceedings of COLING'94*, volume 1, pp. 180–186.

Soudi, A. 2002. *A Computational Lexeme-based Treatment of Arabic Morphology*. Ph.D. thesis, Mohamed V University (Morocco) and Carnegie Mellon University (USA).

Stanfill, C. and D. Waltz. 1986. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, December.

Van den Bosch, A. and W. Daelemans. 1999. Memory-based morphological analysis. In *Proceedings of the 37th Annual Meeting of the ACL*, pp. 285–292, San Francisco, CA. Morgan Kaufmann.

Van den Bosch, A. and W. Daelemans. 2006. Improving sequence segmentation learning by predicting trigrams. In *Proceedings of the Ninth Conference on Natural Language Learning, CoNLL-2005*, pp. 80–87, Ann Arbor, MI.

Van den Bosch, A., I. Schuurman, and V. Vandeghinste. 2006. Transferring PoS-tagging and lemmatization tools from spoken to written Dutch corpus development. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC-2006*, Trento, Italy.

Van Rijsbergen, C. J. 1979. *Information Retrieval.* Buttersworth, London.

Zavrel, J. and W. Daelemans. 1999. Recent advances in memory-based part-of-speech tagging. In *VI Simposio Internacional de Comunicacion Social*, pp. 590–597.

PART IV

Integration of Arabic Morphology in Larger Applications

12

Light Stemming for Arabic Information Retrieval

Leah S. Larkey[1], Lisa Ballesteros[2] and Margaret E. Connell[3]

[1] *Chiliad Publishing, 44 Belchertown Rd, Amherst, MA 01002*
LarkeyLeah@gmail.com

[2] *Computer Science Dept., Mt. Holyoke College, South Hadley, MA 01075*
lballest@mtholyoke.edu

[3] *Univ. of Massachusetts, Dept. of Computer Science, Amherst, MA 01003*
connell@cs.umass.edu

Abstract: Computational Morphology is an urgent problem for Arabic Natural Language Processing, because Arabic is a highly inflected language. We have found, however, that a full solution to this problem is not required for effective information retrieval. Light stemming allows remarkably good information retrieval without providing correct morphological analyses. We developed several light stemmers for Arabic, and assessed their effectiveness for information retrieval using standard TREC data. We have also compared light stemming with several stemmers based on morphological analysis. The light stemmer, light10, outperformed the other approaches. It has been included in the Lemur toolkit, and is becoming widely used Arabic information retrieval

12.1 Introduction

The central problem of Information Retrieval (IR) is to find documents that satisfy a user's information need, usually expressed in the form of a query. This active research area has seen great progress in recent decades, which everyone has experienced in searching the internet. Initially, most IR research was carried out in English and fueled by the annual Text Retrieval Conferences (TREC) sponsored by NIST (the National Institute of Standards and Technology). NIST has accumulated large amounts of standard data (text collections, queries, and relevance judgments) so that IR researchers can compare their techniques on common data sets. More recently, IR research involving other languages has flourished. TREC now includes multilingual data and in recent years, other organizations sponsor similar annual evaluations for European languages (CLEF) and Asian languages (NTCIR) (Chinese, Japanese, and Korean). Arabic began to be included in the TREC cross-lingual track in 2001, and in the TDT (topic detection and tracking) evaluations in 2001 [47]. The availability of standard Arabic data sets from the NIST and the Linguistic Data Consortium (LDC) has in turn spurred a huge acceleration in progress in information retrieval and other natural language processing applications involving Arabic.

Any discussion of multilingual retrieval requires a distinction between monolingual retrieval in multiple languages, and cross-lingual or cross-language retrieval. In monolingual retrieval, queries are issued in the same language as the documents in the collection being searched. In cross-lingual retrieval, queries are issued in a different language than the documents in the collection. A central problem in both monolingual and cross-lingual streams of IR research is the *vocabulary mismatch* problem. The same information need can be expressed using different terminology (the disease *bilharzia* is also called *schistosomiasis*), or a key term can have different spellings (e.g. *theater* versus *theatre*), or may be inflected differently in the query than in the relevant documents. We discuss in Section 12.1.1 the kinds of variability that lead to a particularly acute vocabulary mismatch problem in Arabic information retrieval. Morphological variation in IR has generally been handled by stemming, an unsophisticated but effective approach to morphology which we discuss in Section 12.1.2.

12.1.1 Arabic Morphology and Orthography

The morphological and orthographic complexity of Arabic (see Chapter 3 of this volume) makes it particularly difficult to develop natural language processing applications for Arabic information retrieval.

Distributional analyses of Arabic newspaper text show empirically that there is more lexical variability in Arabic than in the European languages for which most IR and NLP work has been performed. Arabic text has more words occurring only once and more distinct words than English text samples of comparable size.[1] The token to type ratio (mean number of occurrences over all distinct words in the sample) is smaller for Arabic texts than for comparably sized English texts [28].

For information retrieval, this abundance of forms, lexical variability, and orthographic variability, all result in a greater likelihood of mismatch between the form of a word in a query and the forms found in documents relevant to the query. In cross-language retrieval there is an additional serious mismatch problem between query terms and the forms found in the bilingual dictionaries that are used in cross-language retrieval.

To deal with this variability, most morphological analyzers attempt to insert missing short vowels and other diacritics. However for information retrieval, the opposite approach is more typical. Text is normalized by removing many diacritics and short vowels.

[1] We use the term *word* in simple sense of text segmented at white space or punctuation, without any morphological analysis.

12.1.2 Stemming in Information Retrieval

Stemming is another one of many tools besides normalization that is used in information retrieval to combat this vocabulary mismatch problem. Stemmers equate or *conflate* certain variant forms of the same word like (*paper, papers*) and (*fold, folds, folded, folding…*). In this work, we use the term *stemming* to refer to any process which conflates related forms or groups forms into equivalence classes, including but not restricted to suffix stripping. In this section we review general approaches to stemming over many languages. We focus on Arabic in the next section. Most approaches fall into two classes: affix removal and statistical stemming.

12.1.2.1 Affix Removal

In English and many other western European languages, stemming is primarily a process of suffix removal [41][50]. Such stemmers do not conflate irregular forms such as (*goose, geese*) and (*swim, swam*). These stemmers are generally tailored for each specific language. Their design requires some linguistic expertise in the language and an understanding of the needs of information retrieval. Stemmers have been developed for a wide range of languages including Malay [54], Latin [29], Indonesian [8], Swedish [12], Dutch [35], German [44], French [45], Slovene [49], and Turkish [21]. The effectiveness of stemming across languages is varied and influenced by many factors. A reasonable summary is that stemming doesn't hurt retrieval; it either makes little difference or it improves effectiveness by a small amount [31]. Stemming is considered to aid recall more than precision [35]. That is, stemming allows a search engine to find more relevant documents, but may not improve its ability to rank the best documents at the top of the list. Stemming appears to have a larger positive effect when queries and/or documents are short [36], and when the language is highly inflected [48][49], suggesting that stemming should improve Arabic information retrieval.

12.1.2.2 Statistical Techniques

Statistical methods can provide a more language-independent approach to conflation. Related words can be grouped based on various string-similarity measures. Such approaches often involve n-grams. Equivalence classes can be formed from words that share word-initial letter n-grams or a threshold proportion of n-grams throughout the word, or by refining these classes with clustering techniques. This kind of statistical stemming has been shown to be effective for many languages, including English, Turkish, and Malay [21][23][24][46].

Statistical techniques have widely been applied to automatic morphological analysis in computational linguistics [9][17][22][26][27][30][32][34]. For example, Goldsmith finds the best set of frequently occurring stems and suffixes using an information theoretic measure [26]. Oard et al. consider the most frequently occurring word-final n-grams (1, 2, 3, and 4-grams) to be suffixes [46].

Stem classes can also be built or refined using co-occurrence analysis, which Xu and Croft proposed as a promising language-independent approach to stemming [55]. Stemmers make two kinds of errors. Weak stemmers fail to conflate related forms that should be grouped together. Strong stemmers tend to form larger stem classes in which unrelated forms are erroneously conflated. Most stemmers fall between these two extremes and make both kinds of errors. Xu and Croft employ a corpus analysis approach which is particularly suited to splitting up stem classes created by strong stemmers. The stem-classes are reclustered based on a co-occurrence measure, which is language independent in that it can be applied to any set of stem classes. Xu and Croft applied their technique to effectively stem English and Spanish and obtained two important results. First, one can refine an already good stemmer by co-occurrence analysis and improve retrieval effectiveness. Second, one can start with a strong crude stemmer like an n-gram stemmer and use co-occurrence analysis to yield stem classes that work as well as a sophisticated stemmer. They demonstrated an improvement in retrieval effectiveness for English and Spanish after clustering conventional and n-gram based stem classes.

12.1.3 Stemming and Morphological Analysis in Arabic for Information Retrieval

The factors described in Section 12.1.1 make Arabic very difficult to stem. The issue of whether roots or stems are the desired level of analysis for IR has been one complication that has given rise to additional approaches to stemming for Arabic besides affix removal and the statistical stemming approaches described above. Other approaches include manual dictionary construction, morphological analysis, and new statistical methods involving parallel corpora.

12.1.3.1 Manual Construction of Dictionaries

Early work on Arabic stemming used manually constructed dictionaries. Al-Kharashi and Evens worked with small text collections, for which they manually built dictionaries of roots and stems for each word to be indexed [4]. This approach is obviously impractical for realistic sized corpora.

12.1.3.2 Affix Removal

The affix removal approach is generally called *light stemming* when applied to Arabic, referring to a process of stripping off a small set of prefixes and/or suffixes, without trying to deal with infixes, or recognize patterns and find roots. Light stemming was used for Arabic by some authors without details in work prior to ours [3][18]. No explicit lists of strippable prefixes and/or suffixes or algorithms had been published at the time we did this research. Our light stemmer, *light10*, strips off initial و (*w*=and), some prepositions, definite articles (ال، وال، بال، كال، فال، لل) and suffixes (ها، ان، ات، ون، ين، يه، ية، ه، ة، ي). (More detail can be found in Section 12.2.2.) It was designed to strip off strings that were frequently found as prefixes or suffixes, but infrequently found at the beginning or ending of stems. It was not intended to be exhaustive. Darwish introduced the *Al-Stem* light stemmer at TREC 2002 [16], and demonstrated that it was less effective than light10. Chen and Gey [13] introduced a light stemmer similar to light10, but that removed more prefixes and suffixes. It was shown to be more effective than Al-Stem, but was not directly compared to light10.

Although light stemming can correctly conflate many variants of words into large stem classes, it can fail to conflate other forms that should go together. For example, broken (irregular) plurals for nouns and adjectives do not get conflated with their singular forms, and past tense verbs do not get conflated with their present tense forms, because they retain some affixes and internal differences. In spite of the simplicity and shortcomings of light stemming, more sophisticated approaches have not proven to be more effective for information retrieval.

12.1.3.3 Statistical Stemming

Although n-gram systems have been used for many different languages, one would not expect them to perform well on infixing languages like Arabic. However, Mayfield et al. have developed a system that combines word-based and 6-gram based retrieval, which performs remarkably well for many languages [43] including Arabic [42].

De Roeck and Al-Fares [18] used clustering on Arabic words to find classes sharing the same root. Their clustering was based on morphological similarity, using a string similarity metric tailored to Arabic morphology, which was applied after removing “a small number of obvious affixes.” They evaluated the technique by comparing the derived clusters to “correct” classes. They did not assess the performance in an information retrieval context.

We applied Xu and Croft’s co-ocurrence method to Arabic [55]. We assumed that initial n-gram based stem classes were probably not the right starting point for languages like Arabic. However, co-occurrence or other clustering techniques can be applied to Arabic without using n-grams. Instead, we formed classes of words that mapped onto the same string if vowels were removed, and used co-occurrence measures to split these classes further. The co-occurrence method did not work as well for Arabic as it did for English and Spanish. It did not produce a stemmer that worked as well as light10. We found that while one could improve a mediocre stemmer with this technique, its effectiveness was still far from the level attained by a high-quality stemmer like light10. And the light10 stemmer could not be further improved by co-occurrence analyses. Perhaps this is because of the big stemming effect in Arabic compared to English or Spanish.

A promising new class of statistical stemmers makes use of parallel corpora. Chen and Gey [13] used a parallel English-Arabic corpus and an English stemmer to cluster Arabic words into stem classes based on their mappings to English stem classes. Rogati, McCarley, and Yang [51] use a statistical machine translation approach that learns to split words into prefix, stem, and suffix by training on a small hand annotated training set and using a parallel corpus These approaches work well considering how automated they are, but they are not as effective in an IR evaluation as a good light stemmer.

12.1.3.4 Morphological Analysis

It is often assumed that stemming is just a quick and dirty way to approximate morphological analysis, and that the best way to stem would be to perform a correct morphological analysis and then use some valid morphological unit for indexing and retrieval. For Arabic, this unit has often been thought to be the root. Several morphological analyzers have been developed for Arabic [2][6][7][15][33][19] but few have received a standard IR evaluation Most such morphological analyzers find the root, or any number of possible roots for each word. A morphological analyzer called Sebawai, developed by Kareem Darwish [14] was used by some of the TREC participants in 2001 [15][33], but it was not directly compared with light stemming.

In our earlier research we evaluated a simple morphological analyzer from Khoja and Garside [33], which first peels away layers of prefixes and suffixes, then checks a list of patterns and roots to determine whether the remainder could be a known root with a known pattern applied. If so, it returns the root. Otherwise, it returns the original word, unmodified. This system also removes terms that are found on a list of 168 Arabic stop words. It was almost as effective as light stemming, but tended to fail on foreign words, which it left unchanged rather than removing definite articles and obvious affixes. Taghva, Elkhoury, and Coombs [53] have developed a system that finds Arabic roots somewhat like Khoja's approach, but without using a root dictionary or lexicon, and which performs as well as a light stemmer.

Tim Buckwalter's morphological analyzer [10] is different from the others in that it returns stems rather than roots. It is based on a set of lexicons of Arabic stems, prefixes, and suffixes, with truth tables indicating their legal combinations. The BBN group used this table-based stemmer in TREC-2001 [56], but did not compare it with light stemming. The Buckwalter stemmer is now available from LDC [39], and is evaluated as a stemmer in the present study. Finally, Diab, Hacioglu, and Jurafsky [20] developed a set of tools for Arabic morphological analyses which learn tokenization, lemmatization, part-of-speech assignment, and phrase chunking, automatically using SVM (support vector machines), a machine learning categorization tool. Their tools are trained on a sample of the Arabic Tree Bank [40], which is a portion of the AFP database which has been analyzed by the Buckwalter morphological analyzer, and hand-corrected. They claim above 99% accuracy on tokenization, and 95.49% accuracy on POS tagging. We derive some stemmers from these tools, as part of the present study.

Early published comparisons of stems versus roots for information retrieval have claimed that roots are superior to stems, based on small, nonstandard test sets [1][4]. Recent work at TREC has found no consistent differences between roots and stems [15]. We found a small increase in effectiveness when we combined roots and stems [38]. However, we feel that roots versus stems is not the most interesting question to investigate. As this book makes clear, morphological analysis of Arabic is now an active research area, and many systems are being developed to return more complete analyses of Arabic words. A more interesting question is how to use morphological analysis to aid information retrieval, and in particular, to aid stemming. The new work in this chapter attempts to use morphological analysis to get to something better than a root for indexing.

The present study expands upon work we published at SIGIR in 2002 [37]. At that time, we developed several light stemmers, compared their effectiveness on an IR task with each other and with that of a morphological analyzer available at that time. We also experimented with a co-occurrence approach to improving stemming. The present research expands that study in several ways. First, the light stemmer we eventually settled upon (and used in TREC 2002) was slightly different from the best one reported at SIGIR. We compare it with the other stemmers here. Second, we use 75 queries from TREC 2001 and TREC 2002 to evaluate stemmers here, providing results that are more reliable than the 25 queries from TREC 2001 used in the previous study. Third, we evaluate stemming approaches based on two morphological analyzers that were not available when the earlier study was carried out.

12.2 Review of 2002 Stemming Experiments

In this section we review the light stemming experiments from the SIGIR article, but include in addition the modified stemmer, light10. This set of experiments was carried out using the TREC 2001 corpus and queries.

12.2.1 Experimental Method

The TREC-2001 Arabic corpus, also called the AFP_ARB corpus, consists of 383,872 newspaper articles in Arabic from *Agence France Presse*. This fills up almost a gigabyte in UTF-8 encoding as distributed by the Linguistic Data Consortium. There were 25 topics with relevance judgments, available in Arabic, French, and English, with *Title*, *Description*, and *Narrative* fields [25]. We used the Arabic titles and descriptions as queries in monolingual experiments, and the English titles and descriptions in cross-language experiments.

Corpus and queries were converted to CP1256 encoding and indexed using an in-house version of the INQUERY retrieval engine [11]. Arabic strings were treated as a simple string of bytes, regardless of how they would be rendered on the screen. Text was broken up into words at any white space or punctuation characters, including Arabic punctuation. Stop words were removed, using a stop word list from Khoja [33]. Words of one-byte length (in CP1256 encoding) were not indexed. The experiments reported here used INQUERY for retrieval.

Except for the *raw* condition, in which no normalization or stemming was used, the corpus and queries were normalized according to the following steps:

- Remove punctuation
- Remove diacritics (primarily weak vowels). Some entries contained weak vowels, in particular, the dictionaries used in cross-language experiments. Removal made everything consistent.
- Remove non letters
- Replace آ , إ , and أ with ا
- Replace final ى with ي
- Replace final ة with ه

For the normalized conditions and the stemming conditions, we normalized and stemmed all tokens before indexing the corpus, and normalized and stemmed the queries with the same stemmer for retrieval. Arabic queries were expanded using the technique of local context analysis, adding 50 terms from the top-10 documents, as described in detail in [38]. Expansion was performed in order to show the ultimate level of performance attainable using the stemmers in the context of our whole system.

12.2.2 Light Stemmers

Our guiding principle in designing the light stemmers was heuristic. A light stemmer is not dictionary driven, so it cannot apply a criterion that an affix can be removed only if what remains is an existing Arabic word. In fact, we suspect that part of the success of such stemmers is that they can blindly work on words even if they are not found in a word list. The attempt was to try to remove strings that would be found reliably as affixes far more often than they would be found as the beginning or end of an Arabic stem without affixes. We also benefited from discussions with some colleagues at TREC-2001, particularly M. Aljlayl. We tried several versions of light stemming, all of which followed the same steps:

1. Remove و ("and") for light2, light3, and light8, and light10 if the remainder of the word is 3 or more characters long. Although it is important to remove و, it is also problematic, because many common Arabic words begin with this character, hence the stricter length criterion here than for the definite articles.
2. Remove any of the definite articles if this leaves 2 or more characters.
3. Go through the list of suffixes once in the (right to left) order indicated in Table 12.1, removing any that are found at the end of the word, if this leaves 2 or more characters.

Table 12.1. Strings removed by light stemming

	Remove prefixes	Remove Suffixes
Light1	ال، وال، بال، كال، فال	none
Light2	ال، وال، بال، كال، فال، و	none
Light3	"	ه، ة
Light8	"	ها، ان، ات، ون، ين، يه، ية، ه، ة، ي
Light10	ال، وال، بال، كال، فال، لل، و	"

The strings to be removed are listed in Table 12.1. The "prefixes" are actually prepositions, definite articles and a conjunction. The light stemmers do not remove any strings that would be considered Arabic prefixes.

12.2.3 Results of Monolingual Stemmer Comparisons

Figure 12.1 shows precision at 11 recall points for the primary stemmers tested. *Raw* means no normalization or stemming. *Norm* refers to normalization with no stemming. *Light1, Light2, Light3, Light8, and Light10* refer to the light stemmers described above.

Table 12.2 shows uninterpolated average precision for the basic stemmers. For raw, normalized, and light stemming conditions performance is better with each successive increment in degree of stemming. Each of these increments is statistically significant except light10 versus light8.[2] As these results indicate, light stemming is remarkably effective. Light10 has become widely used, and has been included in the Lemur toolkit, a set of software tools for research in language modeling and information retrieval [5].

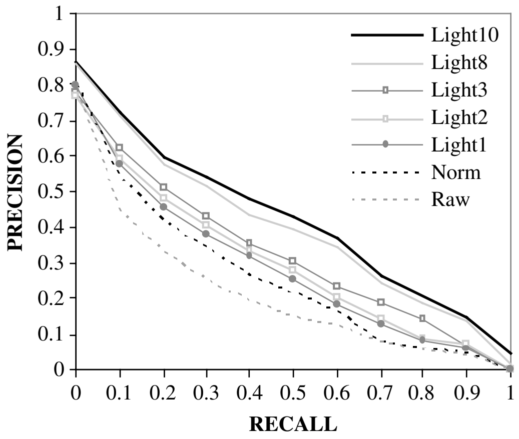


Fig. 12.1. Monolingual 11 point precision for basic stemmers, unexpanded queries

[2] All significance tests were conducted using the Wilcoxon test 0 with a criterion of p<.05 for significance.

Table 12.2. Monolingual average precision for basic stemmers, unexpanded queries

Stemmer	Average Precision	Percent Change
raw	.196	
norm	.241	+22.9
light1	.273	+39.3
light2	.291	+48.3
light3	.317	+61.8
light8	.390	+98.7
light10	.413	+100.1

12.2.4 Comparison with Morphological Analysis

The Khoja stemmer described in 12.1.3 was used to find roots for indexing and retrieval. Average precision for the Khoja stemmer is .341, significantly worse than light10 ($p<.01$). A comparison of this approach with a raw, normalized, and light2 and light10 stemmers can be seen in Figure 12.2. A similar experiment with query expansion showed similar results, seen in Figure 12.3. In Figure 12.3, *Raw* is the original *raw* condition with unexpanded queries, and *RawExp* refers to the raw condition with query expansion. As expected, average precision is higher with expanded queries, but the same pattern of results holds. In particular, the light10 stemmer is significantly more effective than the khoja stemmer.

12.2.5 Cross-language Retrieval

The Khoja morphological analyzer was also compared with the stemmers in a cross-language retrieval experiment, for generality. The cross-language experiments

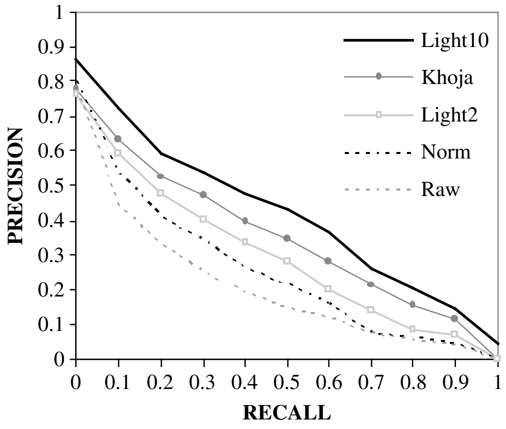


Fig. 12.2. Khoja morphological analyzer versus light stemming, unexpanded queries

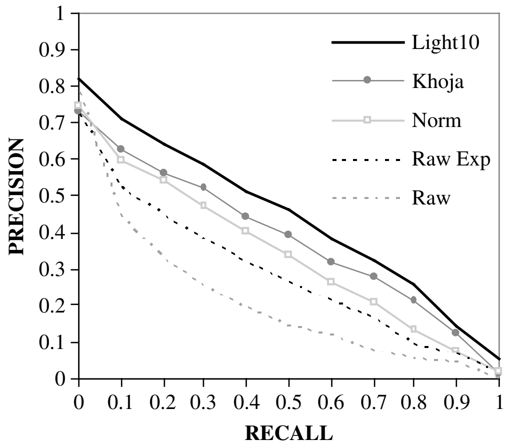


Fig. 12.3. Khoja morphological analyzer versus light stemming, expanded queries

reported here were carried out using the 25 English TREC-2001 queries and the same Arabic AFP_ARB corpus used for the monolingual experiments. Our approach was the common dictionary-based approach, in which each English query word was looked up in a bilingual dictionary. All the Arabic translations for that word were gathered inside an INQUERY #syn (synonym) operator. For an Arabic-English dictionary, we used a lexicon collected from several online English-Arabic and Arabic-English resources on the web, described more completely in [38]. Query expansion was carried out in conjunction with stemming. When English queries were expanded, 5 terms were added from the top-10 documents. When Arabic queries were expanded, 50 terms were added from the top-10 documents, as described in [38].

Figure 12.4 shows precision on unexpanded queries for cross-language retrieval at 11 recall points for raw, norm (normalization and stop word removal), light10 (light10 stemming with stop word removal), and khoja stemmers. Figure 12.5 shows the same information for retrieval with query expansion.

Table 12.3 shows uninterpolated average precision for unexpanded and expanded queries.

The cross-language results are somewhat different from the monolingual results in comparing the light stemmers with the Khoja morphological analyzer. Raw retrieval without any normalization or stemming is far worse for cross-language retrieval than for monolingual retrieval. This is probably because many of the Arabic words occurred in vocalized form (with diacritics) in the online dictionary we used for cross-language retrieval. Without normalization these dictionary entries do not match their counterparts in the corpus. Other differences from the monolingual case are that the *light10* stemmer is far better than the root stemmer, *khoja*, which is no better than normalization for cross-language retrieval. For

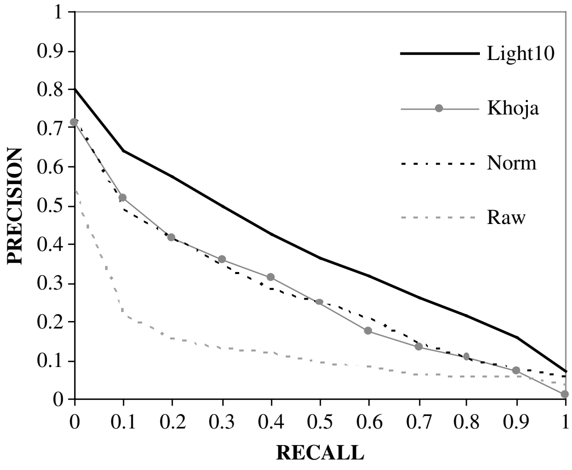


Fig. 12.4. Cross-Language 11 point precision for unexpanded queries

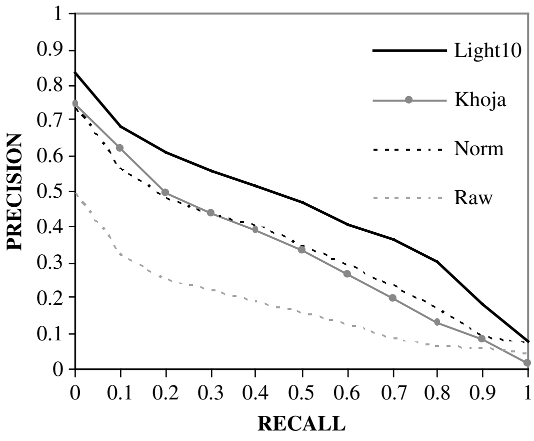


Fig. 12.5. Cross-language 11 point precision for expanded queries

Table 12.3. Cross-language average precision different stemmers, unexpanded and expanded queries

Stemmer	Raw	Norm	Khoja	Light10
Average Precision	.113	.262	.260	.384
Percent Change		+133	+130	+240
With English Query Expansion				
Average Precision	.139	.306	.308	.425
Percent Change		+120	+121	+206
With English and Arabic Query Expansion				
Average Precision	.163	.336	.321	.447
Percent Change		+106	+97	+174

cross-lingual retrieval, roots are probably even less appropriate as look-up units than they are for monolingual retrieval. In cross-lingual retrieval based on dictionary look-up, if we look up the root for each query word, we get far too many translations, and most of them are incorrect.

12.2.6 Discussion

Although stemming is difficult in a language with complex morphology like Arabic, it is particularly important. For monolingual retrieval, we saw around 100% increase in average precision from raw retrieval to the best stemmer. The best stemmer in our experiments, light10, was very simple and did not try to find roots or take into account most of Arabic morphology. It is probably not essential for the stemmer to yield the correct forms, whether stems or roots. It is sufficient for it to group together most of the forms that belong together.

12.3 New Studies of Stemming via Morphological Analysis

Since 2002, more morphological analysis tools have become available. It is also clear that there are probably better ways to use morphological analysis in stemming than simply to use the roots for indexing. In this part of the chapter, we report research on using the Buckwalter morphological analyzer, and the Diab tokenizer and part of speech tagger to aid the stemming process.

12.3.1 Buckwalter Morphological Analyzer

Tim Buckwalter's morphological analyzer has been made available through the Linguistic Data Consortium (LDC) [39]. It takes as input Arabic words with or without short vowels and performs morphological analysis and POS tagging using three dictionaries and three compatibility tables. The three dictionaries list possible prefixes, Arabic stems, and possible suffixes. The three compatibility tables indicate (1) compatible prefix/stem category pairs, (2) compatible prefix/suffix category pairs, and (3) compatible stem/suffix category pairs. The analyzer performs tokenization, word segmentation, dictionary lookup, compatibility checks, and lists all the possible analyses of each word. For example, for the word الشمالية (AlšmAlyħ), we get the following output according to Buckwalter's transliteration:

```
INPUT STRING: الشمالية
LOOK-UP WORD: Al$mAlyp
  SOLUTION 1: (Al$amAliy~ap) [$amAliy~_1] [$amAliy~]Al/DET+$amAliy~/ADJ+ap/NSUFF_FEM_SG
      (GLOSS): the + north/northern + [fem.sg.]
  SOLUTION 2: (Al$imAliy~ap) [$imAliy~_1] [$imAliy~] Al/DET+$imAliy~/ADJ+ap/NSUFF_FEM_SG
      (GLOSS): the + leftist + [fem.sg.]
```

Note that the second field in square brackets in each solution line is one we added to the morphological analyzer program, AraMorph.pl, to give the stem in a form that is more useful to us. The example illustrates some interesting properties of the analyzer. Although much of our corpus does not include short vowels, the analyses have short vowels. In fact, the stems yielded by the two different solutions above differ only in the short vowels.

It is straightforward to use this morphological analysis for stemming, because it analyzes tokens into up to three parts: prefix, stem, and suffix. To stem we simply remove all prefixes and suffix and use the remaining stems, normalized to be comparable to our light stemmers. A potential problem is in dealing with multiple different analyses for the same word. However, once short vowels were removed and other normalization was performed, many of the different analyses actually yielded the same stem. In the example above, the two different stems, šamAliy~ and šimAliy~, both become the same stem, šmAly, after normalization. Ultimately, the vast majority of words had unique stems. In one sample of 18,035 words, 14,878 (82%) were found to have exactly one solution, 2322 (16%) had more than one solution, and 829 (less than 1%) had no solution.

In particular, the following steps were performed on each file of the AFP_ARB corpus:

1. Run the modified version of AraMorph.pl, to find all the analyses for each word
2. Normalize each stem in each solution by removing short vowels, converting all forms of *alif* to bare *alif*, and changing *alif maqS ra* (ى) to *y* ' (ي).
3. If there is exactly one normalized stem, replace the word with the stem. If there were no solutions found, or more than one distinct normalized stem, use the normalized form of the original word.

To address the problem of what to do if the morphological analyzer gave more than one possible stem, Xu, Fraser, and Weischedel [57] implemented a system that used both analyses when a word had more than one solution, but found the results were not significantly different from one that left words with multiple analyses unstemmed. We decided to implement a second version of a Buckwalter-based stemmer (*Buckwalter+*) that applied light10 to words if the analyzer found zero analyses or more than one analysis.

12.3.2 Diab Tokenizer, Lemmatizer and POS Tagger

The Diab morphological analysis tools are available for download on the internet. Their distribution ArabicSVMTools [19] includes the models they trained, which we used without training our own models. Their tokenization segments clitics (prepositions, conjunctions, and some pronouns) from stems, and the part of speech tagger labels each segment with one of 24 parts of speech from a tag set collapsed from the 135 tags created by Buckwalter's AraMorph. We noted, first of all, that the tokenization part of the process separates some of the same segments that a stemmer should remove – it separates و (w) from the beginnings of words, and determiners like ال (Al). It also separates some prepositions like ب (b) and ل (l), which light10 does not do, unless they precede ال (Al). It also separates some suffixes, like possessive pronoun enclitics. The POS tagger then tags these segments with a POS label, which lets us identify closed-class segments and remove them to accomplish stopword removal. It also allows us to remove additional suffixes contingent on part of speech.

To use the tagger for stemming, we first modified our query and corpus files to contain one sentence per line, because the analyzer operates on sentences. We then ran the tokenizer, lemmatizer, and POS tagger on the sentences, We removed segments with the following tags: CC, DT, RP, PRP, PRP$, CD, IN, WP, WRB, PUNC, NUMERIC_COMMA (conjunction, determiner, particle, personal pronoun, possessive personal pronoun, cardinal number, subordinating conjunction or preposition, relative pronoun, wh-adverb, punctuation). This amounts to a much weaker stemmer than light10, because it removes almost no suffixes. Therefore we tested three other stemmers derived from this morphological analyzer. Our goal was to remove possible plural and dual endings only from words identified as plural nouns and adjectives. Unfortunately, while singular and plural nouns (and singular and plural proper nouns) received distinct tags, adjectives all received the same tag, JJ, so we could not easily determine which were plural or dual. Therefore, we tried two versions of plural suffix removal, described below as Diab2 and Diab3. In an analogue to the Buckwalter+ condition, in which we blindly performed light10 stemming if a word did not yield a unique stem, we also have a Diab+ condition, in which we remove light10 suffixes regardless of part of speech.

To summarize the 4 stemmers derived from the morphological analysis tools:

Diab: tokenization, morphological analysis, remove closed class segments
Diab+: Diab, then remove light10 suffixes
Diab2: Diab, then remove possible plural and dual endings (At, wn, yn, w, An, y) from segments marked as plural nouns or plural proper nouns
Diab3: Diab, then remove (At, wn, yn, w, An, y, yħ, ħ) from any segments marked as nouns or adjectives.

12.3.3 Comparison of New Morphological Stemmers with Light Stemmer

These experiments were carried out in much the same way as those in Section 12.2.3, except for a larger query set. In addition to the 25 queries from TREC2001, there were 50 queries from TREC2002 for a total of 75. The same Arabic corpus was used for retrieval. Figure 12.6 and Figure 12.7 show monolingual retrieval for the 75 queries.

Table 12.4 shows average precision for all the stemming conditions tested, with and without query expansion. The boldface type indicates that the average precision was significantly worse than the corresponding light10 condition.

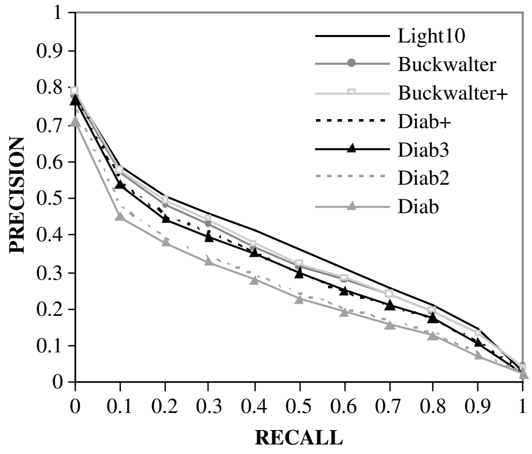


Fig. 12.6. Monolingual 11 point precision for 75 unexpanded queries

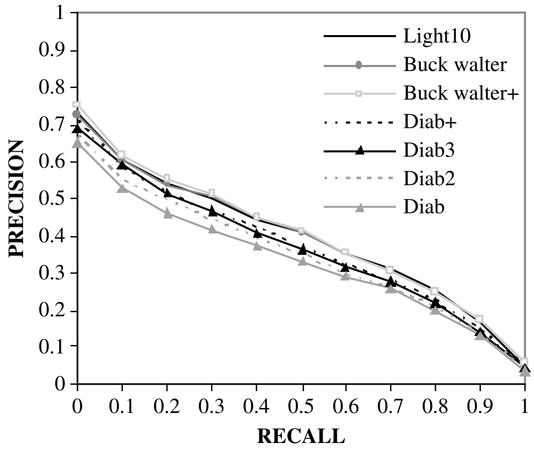


Fig. 12.7. Monolingual 11 point precision for 75 expanded queries

Table 12.4. Average Precision for 75 expanded queries, comparison of morphological stemmers with light10

Stemmer	Unexpanded	Expanded
Light10	.353	.387
Buckwalter	.330	.386
Buckwalter+	.334	.390
Diab	.247	.322
Diab2	.257	.336
Diab3	.302	.354
Diab+	.302	.356

Without query expansion, all of the morphological analysis conditions are significantly worse than light10. With query expansion, the Buckwalter stemmers equal the performance of light10. For both expanded and unexpanded queries, all the Diab stemmers are significantly worse than the Buckwalter stemmers. Diab+ and Diab3 were almost identical, that is, blindly removing the set of light10 suffixes from all words gives the same performance as removing suffixes from nouns and adjectives. Diab+ and Diab3 were significantly better than Diab2, which is significantly better than Diab.

The poor performance of the basic Diab stemmer is not surprising - it is performing stemming that is most comparable to light3 in Section 12.2.3. But we expected it to be more improved by POS-dependent suffix removal, and by its more complete stop word removal. An examination of the query set and a sample AFP article after tokenization and POS tagging showed more tokenization mistakes than we expected. For example, the tokenizer sometimes did not separate the definite article ال (Al).

The queries were made up of a non-sentence (title) line, followed by one to three sentences of description, as in Table 12.5 and Table 12.6. Because the tokenizer was trained on complete sentences, it did not work well on titles. It often failed to segment *Al* when the first word of a title was a noun, as in Table 12.5. Table 12.6 shows an example of incorrect segmentation of *b* from the front of a word in a title, but not in the complete sentence (the description).

Table 12.5. Example of Tokenization, TREC 2002 query AR30

	Title	Description
English	Iraqi satellite television	What is the importance of satellite television in Iraq?
Arabic	**التلفزيون الفضائي في العراق**	**ما اهمية التلفزيون الفضائي في العراق؟**
Transliterated	Altlfzywn AlfDAŷy fy AlςrAq	mA Ahmyħ Altlfzywn AlfDAŷy fy AlςrAq?
Tokenized	Altlfzywn Al fDAŷy fy Al ςrAq	mA Ahmyħ Al tlfzywn Al fDAŷy fy Al ςrAq?

Table 12.6. Example of Tokenization, TREC2002 query AR32

	Title	Description
English	Caspian Beluga Conservation	What Beluga conservation projects are present in the Caspian region?
Arabic	**صيانة بلوغا في بحرقزوين**	**ما هي المشاريع لصيانة البلوغا في بحر قزوين؟**
Transliterated	SyAnħ blwγA fy bHrqzwyn	mA hy Al mšAryς l SyAnħ Al blwγA fy bHr qzwyn
Tokenized	SyAnħ b lwγA fy bHrqzwyn	mA hy Al mšAryς l SyAnħ Al blwγA fy bHr qzwyn

Performance was hurt by these tokenization errors. Note in the two examples that the tokenization errors were serious - they occurred in important content words in the query, but the same words were correctly tokenized in the description part of the query. Performance would have been hurt even more on title-only queries.

12.4. Conclusions

Stemming has a large effect on Arabic information retrieval, far larger than the effect found for most other languages. For monolingual retrieval we have demonstrated improvements of around 100% in average precision due to stemming and related processes, and an even larger effect for dictionary-based cross-language retrieval. This stemming effect is very large, compared to that found in many other stemming studies, but is consistent with the hypothesis of Popovič and Willett [49] and Pirkola [48] that stemming should be particularly effective for languages with more complex morphology.

The best stemmer was a light stemmer that removed stop words, definite articles, and و ("and") from the beginning of words, and a small number of suffixes from the end of words (*light10*). With query expansion, *light10* yielded results comparable to that of the top performers at TREC, monolingual and cross-language. We have now compared light stemming with several different stemming approaches based on morphological analysis: indexing roots returned by morphological analysis, indexing stems returned by morphological analysis, and somewhat more intelligent stemming based on part-of-speech assignments. Although we can equal the light stemmer with stemming based on Buckwalter's morphological analysis, we have not been able to attain significantly better performance using morphological analysis.

Given the morphological complexity of Arabic, why would a morphological analyzer not perform better than such a simple stemmer? We hypothesize several factors. First, morphological analyzers make mistakes, particularly on names. In dictionary-driven Buckwalter approach, words are exempted from stemming if they are not found in the lexicon. With the Diab approach, we observed many mistakes of tokenization in morphological analysis, which prevented words from getting the correct part of speech, and therefore did not undergo the correct POS-dependent modifications. If one took a sample of Arabic text with complete sentences, the tokenization and POS tagging would have fewer errors. Note, however, that Arabic text contains so many definite articles that one could obtain the claimed >99% tokenization accuracy simply by removing *Al* from the beginnings of words.

Second, models used in IR treat documents and queries as "bags of words," or at best, bags of unigrams, bigrams, and trigrams. Our current retrieval models may not be able to use the information provided by morphological analysis.

Third, light stemming is robust. It does not require complete sentences. It does not try to handle every single case. It is sufficient for information retrieval that many of the most frequently occurring forms of a word be conflated. If an occasional form is missed, it is likely that other forms of the same word occur with it in the same documents, so the documents are likely to be retrieved anyway.

Fourth, it is still not clear what the correct level of conflation should be for IR. Clearly, we do not want to represent Arabic words by their roots, and equate all words derived from the same root. But we still believe that light stemmers are too weak. None of the light stemmers correctly groups broken plurals with their singular forms.

These studies are only a beginning. We have not ruled out the possibility that a better morphological analyzer, and better use of morphological analysis to conflate words, could work better than a light stemmer. We have only tried a few obvious alternatives. Ultimately, one would like to be able to conflate all the inflected forms of a noun together, including broken plurals, and all the conjugations of a verb, which we cannot do today. Clearly, there is room for future work that makes intelligent use of morphological analysis in information retrieval.

Acknowledgments

We would like to thank Shereen Khoja for providing her stemmer, Nicholas J. DuFresne for writing some of the stemming and dictionary code, Fang-fang Feng for helping with dictionary collection over the web, Mohamed Taha Mohamed, Mohamed Elgadi, and Nasreen Abdul-Jaleel for help with the Arabic language, Victor Lavrenko for the use of his vector and language modeling code. This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

[1] Abu-Salem, H., Al-Omari, M., and Evens, M. Stemming methodologies over individual query words for Arabic information retrieval. *JASIS*, 50 (6), pp. 524–529, 1999.

[2] Al-Fedaghi, S. S. and Al-Anzi, F. S. A new algorithm to generate Arabic root-pattern forms. In *Proceedings of the 11th national computer conference*. King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, pp. 391–400, 1989.

[3] Aljlayl, M., Beitzel, S., Jensen, E., Chowdhury, A., Holmes, D., Lee, M., Grossman, D., and Frieder, O. IIT at TREC-10. In *TREC 2001*. Gaithersburg: NIST, pp. 265–275, 2001.

[4] Al-Kharashi, I. and Evens, M. W. Comparing words, stems, and roots as index terms in an Arabic information retrieval system. *JASIS*, 45 (8), pp. 548–560, 1994.

[5] Allan, J. Callan, J. Collins-Thompson, K. Croft, B. Feng, F. Fisher, D. Lafferty, J. Larkey, L. Truong, T. N. Ogilvie, P. Si, L. Strohman, T. Turtle, H. and Zhai, C. The Lemur toolkit for language modeling and information retrieval. http://www.lemurproject.org/˜lemur

[6] Al-Shalabi, R. *Design and implementation of an Arabic morphological system to support natural language processing*. PhD thesis, Computer Science, Illinois Institute of Technology, Chicago, 1996.

[7] Beesley, K. R. Arabic finite-state morphological analysis and generation. In *COLING-96: Proceedings of the 16th international conference on computational linguistics*, vol. 1, pp. 89–94, 1996.

[8] Berlian, V., Vega, S. N., and Bressan, S. Indexing the Indonesian web: Language identification and miscellaneous issues. Presented at Tenth International World Wide Web Conference, Hong Kong, 2001.

[9] Brent, M. R. Speech segmentation and word discovery: A computational perspective. *Trends in Cognitive Science*, 3 (8), pp. 294–301, 1999.

[10] Buckwalter, T. *Qamus: Arabic lexicography*. http://www.qamus.org/

[11] Callan, J. P., Croft, W. B., and Broglio, J. TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, 31 (3), pp. 327–343, 1995.

[12] Carlberger, J., Dalianis, H., Hassel, M., and Knutsson, O. Improving precision in information retrieval for Swedish using stemming. In *Proceedings of NODALIDA '01–13th Nordic conference on computational linguistics*. Uppsala, Sweden, 2001. http://www.nada.kth.se/~xmartin/papers/-Stemming_NODALIDA01.pdf

[13] Chen, A. and Gey, F. Building an Arabic stemmer for information retrieval. In *TREC 2002*. Gaithersburg: NIST, pp 631–639, 2002.

[14] Darwish, K. Building a shallow morphological analyzer in one day. ACL 2002 Workshop on Computational Approaches to Semitic languages, pp. 47–54, July 11, 2002.

[15] Darwish, K., Doermann, D., Jones, R., Oard, D., and Rautiainen, M. TREC-10 experiments at Maryland: CLIR and video. In *TREC 2001*. Gaithersburg: NIST, pp 549–562, 2001.

[16] Darwish, K. and Oard, D.W. CLIR Experiments at Maryland for TREC-2002: Evidence combination for Arabic-English retrieval. In *TREC 2002*. Gaithersburg: NIST, pp 703–710, 2002.

[17] de Marcken, C. *Unsupervised language acquisition*. PhD thesis, MIT, Cambridge, 1995.

[18] De Roeck, A. N. and Al-Fares, W. A morphologically sensitive clustering algorithm for identifying Arabic roots. In *Proceedings ACL-2000*. Hong Kong, pp 199–206, 2000.

[19] Diab, M. ArabicSVMTools. http://www.stanford.edu/~mdiab/software/ArabicSVMTools.tar.gz. 2004.

[20] Diab, M., Hacioglu, K., and Jurafsky, D. Automatic tagging of Arabic text: From raw test to base phrase chunks. In *Proceedings of HLT-NAACL*, pp 149–152, 2004. http://www.stanford.edu/~mdiab/papers/ArabicChunks.pdf.

[21] Ekmekcioglu, F. C., Lynch, M. F., and Willett, P. Stemming and n-gram matching for term conflation in Turkish texts. *Information Research News*, 7 (1), pp. 2–6, 1996.

[22] Flenner, G. Ein quantitatives Morphsegmentierungssytem fur Spanische Wortformen. In *Computatio linguae II*, U. Klenk, Ed. Stuttgart: Steiner Verlag, pp. 31–62, 1994.

[23] Frakes, W. B. Stemming algorithms. In *Information retrieval: Data structures and algorithms*, W. B. Frakes and R. Baeza-Yates, Eds. Englewood Cliffs, NJ: Prentice Hall, Chapter 8, 1992.

[24] Freund, E. and Willett, P. Online identification of word variants and arbitrary truncation searching using a string similarity measure. *Information Technology: Research and Development*, 1, pp. 177–187, 1982.

[25] Gey, F. C. and Oard, D. W. The TREC-2001 cross-language information retrieval track: Searching Arabic using English, French, or Arabic queries. In *TREC 2001*. Gaithersburg: NIST, pp 16–26, 2002.

[26] Goldsmith, J. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27 (2), pp. 153–198, 2000.

[27] Goldsmith, J., Higgins, D., and Soglasnova, S. Automatic language-specific stemming in information retrieval. In *Cross-language information retrieval and evaluation: Proceedings of the CLEF 2000 workshop*, C. Peters, Ed.: Springer Verlag, pp. 273–283, 2001.

[28] Goweder, A. and De Roeck, A. Assessment of a significant Arabic corpus. Presented at the Arabic NLP Workshop at ACL/EACL 2001, Toulouse, France, 2001. http://www.elsnet.org/arabic2001/goweder.pdf

[29] Greengrass, M., Robertson, A. M., Robyn, S., and Willett, P. Processing morphological variants in searches of Latin text. *Information Research News*, 6 (4), pp. 2–5, 1996.

[30] Hafer, M. A. and Weiss, S. F. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10, pp. 371–385, 1974.

[31] Hull, D. A. Stemming algorithms - a case study for detailed evaluation. *JASIS*, 47 (1), pp. 70–84, 1996.

[32] Janssen, A. Segmentierung Franzosischer Wortformen in Morphe ohne Verwendung eines Lexikons. In *Computatio linguae*, U. Klenk, Ed. Stuttgart: Steiner Verlag, pp. 74–95, 1992.

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

[34] Klenk, U. Verfahren morphologischer Segmentierung und die Wortstruktur im Spanischen. In *Computatio Linguae*, *Aufsätze zur algorithmischen und quantitativen Analyse der Sprache*, U. Klenk, Ed. Stuttgart: Steiner Verlag, pp 110–124, 1992.

[35] Kraaij, W. and Pohlmann, R. Viewing stemming as recall enhancement. In *Proceedings of ACM SIGIR96*. pp. 40–48, 1996.

[36] Krovetz, R. Viewing morphology as an inference process. In *Proceedings of ACM SIGIR93*, pp. 191–203, 1993.

[37] Larkey, Leah S., Ballesteros, L., and Connell, M. (2002) Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis In *Proceedings of the 25th annual international conference on research and development in information retrieval (SIGIR 2002)*, Tampere, Finland, August 11–15, 2002, pp. 275–282.

[38] Larkey, L. S. and Connell, M. E. Arabic information retrieval at UMass in TREC-10. In *TREC 2001*. Gaithersburg: NIST, 2001.

[39] LDC, Linguistic Data Consortium. Buckwalter Morphological Analyzer Version 1.0, LDC2002L49, 2002. http://www.ldc.upenn.edu/Catalog/.

[40] LDC, Linguistic Data Consortium. Arabic Penn TreeBank 1, v2.0. LDC2003T06, 2003. http://www.ldc.upenn.edu/Catalog/

[41] Lovins, J. B. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11, pp. 22–31, 1968.

[42] Mayfield, J., McNamee, P., Costello, C., Piatko, C., and Banerjee, A. JHU/APL at TREC 2001: Experiments in filtering and in Arabic, video, and web retrieval. In *TREC 2001*. Gaithersburg: NIST, pp 332–341, 2001.

[43] McNamee, P., Mayfield, J., and Piatko, C. A language-independent approach to European text retrieval. In *Cross-language information retrieval and evaluation: Proceedings of the CLEF 2000 workshop*, C. Peters, Ed.: Springer Verlag, pp. 129–139, 2000.

[44] Monz, C. and de Rijke, M. Shallow morphological analysis in monolingual information retrieval for German and Italian. In *Cross-language information retrieval and evaluation: Proceedings of the CLEF 2001 workshop*, C. Peters, Ed.: Springer Verlag, 2001. http://staff.science.uva.nl/~christof/Papers/clef-2001-post.pdf

[45] Moulinier, I., McCulloh, A., and Lund, E. West group at CLEF 2000: Non-English monolingual retrieval. In *Cross-language information retrieval and evaluation: Proceedings of the CLEF 2000 workshop*, C. Peters, Ed.: Springer Verlag, pp. 176–187, 2001.

[46] Oard, D. W., Levow, G. -A., and Cabezas, C. I. CLEF experiments at Maryland: Statistical stemming and backoff translation. In *Cross-language information retrieval and evaluation: Proceedings of the CLEF 2000 workshop*, C. Peters, Ed.: Springer Verlag, pp. 176–187, 2001.

[47] NIST. Topic Detection and Tracking Resources. http://www.nist.gov/speech/tests/tdt/resources.htm. Created 2000, updated 2002.

[48] Pirkola, A. Morphological typology of languages for IR. *Journal of Documentation*, 57 (3), pp. 330–348, 2001.

[49] Popovic, M. and Willett, P. The effectiveness of stemming for natural-language access to Slovene textual data. *JASIS*, 43 (5), pp. 384–390, 1992.

[50] Porter, M. F. An algorithm for suffix stripping. *Program*, 14 (3), pp. 130–137, 1980.

[51] Rogati, M., McCarley, S., and Yang, Y. Unsupervised learning of Arabic stemming using a parallel corpus. In *Proceedings ACL-2003,* Sapporo, Japan, pp. 391–398, July 2003. http://acl.ldc.upenn.edu/acl2003/main/pdf/Rogati.pdf

[52] Siegel, S. *Nonparametric statistics for the behavioral sciences*. New York: McGraw-Hill, 1956.

[53] Taghva, K., Elkoury, R., and Coombs, J. Arabic Stemming without a root dictionary. 2005. www.isri.unlv.edu/publications/isripub/Taghva2005b.pdf

[54] Tai, S. Y., Ong, C. S., and Abdullah, N. A. On designing an automated Malaysian stemmer for the Malay language. (poster). In *Proceedings of the fifth international workshop on information retrieval with Asian languages,* Hong Kong, pp. 207–208, 2000.

[55] Xu, J. and Croft, W. B. Corpus-based stemming using co-occurrence of word variants. *ACM Transactions on Information Systems*, 16 (1), pp. 61–81, 1998.

[56] Xu, J., Fraser, A., and Weischedel, R. TREC 2001 cross-lingual retrieval at BBN. In *TREC 2001*. Gaithersburg: NIST, pp. 68–78, 2001.

[57] Xu, J., Fraser, A., and Weischedel, R. Empirical studies in strategies for Arabic retrieval. In *Sigir 2002*. Tampere, Finland: ACM, pp. 269–274, 2002.

13

Adapting Morphology for Arabic Information Retrieval*

Kareem Darwish[1] and Douglas W. Oard[2]

[1] *IBM Technology Development Center, P.O. Box 166, El-Ahram, Giza, Egypt*
kareem@darwish.org

[2] *College of Information Studies & UMIACS, University of Maryland, College Park, MD 20742*
oard@glue.umd.edu

Abstract: This chapter presents an adaptation of existing techniques in Arabic morphology by leveraging corpus statistics to make them suitable for Information Retrieval (IR). The adaptation resulted in the development of Sebawai, an shallow Arabic morphological analyzer, and Al-Stem, an Arabic light stemmer. Both were used to produce Arabic index terms for Arabic experimentation. Sebawai is concerned with generating possible roots and stems of a given Arabic word along with probability estimates of deriving the word from each of the possible roots. The probability estimates were used as a guide to determine which prefixes and suffixes should be used to build the light stemmer Al-Stem. The use of the Sebawai generated roots and stems as index terms along with the stems from Al-Stem are evaluated in an information retrieval application and the results are compared

13.1 Introduction

Due to the morphological complexity of the Arabic language, Arabic morphology has become an integral part of many Arabic Information Retrieval (IR) and other natural language processing applications. Arabic words are divided into three types: noun, verb, and particle (Abdul-Al-Aal, 1987). Nouns and verbs are derived from a closed set of around 10,000 roots (Ibn Manzour, 2006). The roots are commonly three or four letters and are rarely five letters. Arabic nouns and verbs

* All the experiments for this work were performed while the first author was at the University of Maryland, College Park.

are derived from roots by applying templates to the roots to generate stems and then introducing prefixes and suffixes. Table 13.1 shows some templates for 3-letter roots. Tables 13.2 and 13.3 show some of the possible prefixes and suffixes and their corresponding meaning. The number of unique Arabic words (or surface forms) is estimated to be 6 x 10^{10} words (Ahmed, 2000). Table 13.4 shows some of the words that may be generated from the root ktb – كتب.

Further, a word may be derived from several different roots. For example the word AymAn – ايمان can be derived from five different roots. Table 13.5 shows possible roots for the word AymAn – ايمان and the meaning of the word based on each. For the purposes of this chapter, a word is any Arabic surface form, a stem is a word without any prefixes or suffixes, and a root is a linguistic unit of meaning, which has no prefix, suffix, or infix. However, often irregular roots, which contain double or weak letters, lead to stems and words that have letters from the root that are deleted or replaced.

Table 13.1. Some templates to generate stems from roots with examples from the root (ktb – كتب)

Template	Stem	Meaning
CCC – فعل	ktb – كتاب	books, wrote, etc.
mCCwC – مفعول	mktwb – مكتوب	something written
CCAC – فعال	ktAb – كتاب	book
CCA<u>C</u>yC – فعاعيل	ktAtyb – كتاتيب	Qur'an school
CACC – كاتب	kAtb – كاتب	writer
CcwC – فعول	ktwb – كتوب	skilled writer

Table 13.2. Some example prefixes and their meanings

Prefix	w – و	K – ك	f – ف	l – ل	Al – ال	wAl – وال
Meaning	and	like	Then	to	the	and the

Table 13.3. Some example suffixes and their meanings

Prefix	h – ه	K – ك	hm – هم	km – كم	hA – ها	y – ي
Meaning	his	your (sg.)	Their	your (pl.)	her, its	my

Table 13.4. Some words that can be derived from the root ktb – **كتب**

Prefix	ktb – كتاب	wktAbh – وكتابه	yktb – يكتب	ktAbhm – كتابهم	mktbħ – مكتبة	AlkAtb – الكاتب
Meaning	book	and his book	he writes	their book	library	the writer

Table 13.5. Possible roots for the word AymAn – **ايمان** along with meaning

Root	Meaning
Amn – أمن	peace or faith
Aym – أيم	two poor people
mAn – مأن	will he give support
ymn – يمن	Covenants
ymA – يمأ	will they (fm.) point to

For Arabic IR, several early studies suggested that indexing Arabic text using roots significantly increases retrieval effectiveness over the use of words or stems (Abu-Salem et al., 1999; Al-Kharashi & Evens, 1994; Hmeidi et al., 1997). However, the studies used small test collections of only hundreds of documents and the morphology in many of the studies was done manually. Performing morphological analysis for Arabic IR using existing Arabic morphological analyzers, most of which use finite state transducers (Antworth, 1990; Kiraz, 1998; Koskenniemi, 1983), is problematic for two reasons. First, they were designed to produce as many analyses as possible without indicating which analysis is most likely. This property of the analyzers complicates retrieval, because it introduces ambiguity in the indexing phase as well as the search phase of retrieval. Second, the use of finite state transducers inherently limits coverage, which is the number of words that the analyzer can analyze, to the cases programmed into the transducers. A later study by Aljlayl et al. (2001) on a large Arabic collection of 383,872 documents suggested that lightly stemmed words, where only common prefixes and suffixes are stripped from words, were perhaps better index terms for Arabic. Aljlayl et al. did not report the list of prefixes and suffixes used in the light stemmer. If indeed lightly stemmed words work well, then determining the list of prefixes and suffixes to be removed by the stemmer is desirable. This chapter will focus on two aspects of Arabic morphology. The first is adapting existing Arabic morphological analysis using corpus statistics to attempt to produce the most likely analysis of a word and to improve coverage. The resulting analyzer is called Sebawai. The second is to construct a set of common prefixes and suffixes suitable for light stemming. The resulting light stemmer is called Al-Stem.

Section 13.2 will provide some background on Arabic Morphology and Arabic IR. Section 13.3 will provide a system description of Sebawai along with an evaluation of its correctness and coverage. Section 13.4 describes the development of Al-Stem. Section 13.5 compares Sebawai and Al-Stem in the context of an IR application. Section 13.6 addresses some of the shortcomings of Sebawai and concludes the chapter.

13.2 Background

13.2.1 Arabic Morphology

Significant work has been done in the area of Arabic morphological analysis. Some of the approaches include:

1. The Symbolic Approach: In this approach, morphotactic (rules governing the combination of morphemes, which are meaning bearing units in the language) and orthographic (spelling rules) rules are programmed into a Finite State Transducer (FST). Koskenniemi proposed a two-level system for language morphology, which led to Antworth's two-level morphology system PC-KIMMO (Antworth, 1990; Koskenniemi, 1983). Later, Beesley and Buckwalter developed an Arabic morphology system, ALPNET, which uses a slightly enhanced implementation of PC-KIMMO (Beesley et al., 1989; Kiraz, 1998). However, this approach was criticized by Ahmed (2000) for requiring excessive manual processing to state rules in an FST and for the ability to only analyze words that appear in Arabic dictionaries. Kiraz (1998) summarized many variations of the FST approach.
2. Unsupervised Machine Learning Approach: Goldsmith (2000) developed an unsupervised learning automatic morphology tool called AutoMorphology. This system is advantageous because it could automatically learn the most common prefixes and suffixes from just a word-list. However, such a system would not be able to detect infix and uncommon prefixes and suffixes.
3. Statistical Rule-Based Approach: This approach uses rules in conjunction with statistics. This approach employs a list of prefixes, a list of suffixes, and templates to extract a stem from a word and a root from a stem. Possible prefix-suffix-template combinations are constructed for a word. Hand-crafted rules are used to eliminate impossible combinations and the remaining combinations are then statistically ranked. RDI's system called MORPHO3 utilizes such a model (Ahmed, 2000). Such an approach achieves broad morphological coverage of the Arabic language.
4. Light Stemming Based Approach: In this approach, leading and trailing letters in a word are removed if they match entries in lists of common prefixes and suffixes respectively. The advantage of this approach is that it requires no morphological processing and is hence very efficient. However, incorrect prefixes and suffixes are routinely removed. This approach was used to develop

Arabic stemmers by Aljlayl et al. (2001), Darwish & Oard (2002a), and Larkey, Ballesteros & Connell (2002).

13.2.2 Arabic Information Retrieval

Most early studies of character-coded Arabic text retrieval relied on relatively small test collections (Abu-Salem et al., 1999; Al-Kharashi & Evens, 1994; Darwish & Oard, 2002a; Darwish & Oard, 2002b). The early studies suggested that roots, followed by stems, were the best index terms for Arabic text. Recent studies are based on a single large collection (from TREC-2001/2002), (Darwish & Oard, 2002b; Gey & Oard, 2001) and suggest that perhaps light stemming and character n-grams are the best index terms. The studies examined indexing using words, word clusters (Xu et al., 2001), terms obtained through morphological analysis (e.g., stems and roots (Al-Kharashi & Evens, 1994; Aljlayl et al., 2001)), light stemming, and character n-grams of various lengths (Darwish & Oard, 2002a; Mayfield et al., 2001). The effects of normalizing alternative characters, removal of diacritics and stop-word removal have also been explored (Chen & Gey, 2001; Mayfield et al., 2001; Xu et al., 2001).

13.3 System Description

This section describes the development of an Arabic morphological analyzer called Sebawai, which adapts a commercial Arabic morphological analyzer called ALPNET (Beesley, 1996; Beesley et al., 1989) to find the most likely analysis and to improve coverage. ALPNET is based on a finite state transducer that uses a set of 4,500 roots. Sebawai uses corpus based statistics to estimate the occurrence probabilities of templates, prefixes, and suffixes. Sebawai trains on a list of word-root pairs to:

1. Derive templates that produce stems from roots,
2. Construct a list of prefixes and suffixes, and
3. Estimate the occurrence probabilities of templates, stems, and roots.

The words in the word-root pairs were extracted from an Arabic corpus.

13.3.1 Acquiring Word-Root Pairs

The list of word-root pairs may be constructed either manually, using a dictionary, or by using a preexisting morphological analyzer such as ALPNET (Ahmed, 2000; Beesley, 1996; Beesley et al., 1989) as follows:

1. <u>Manual construction of word-root pair list:</u> Building the list of several thousand pairs manually is time consuming, but feasible. Assuming that a person who

knows Arabic can generate a root for a word every 5 seconds, the manual process would require about 14 hours of work to produce 10,000 word-root pairs.

2. Automatic construction of a list using dictionary parsing: Extracting word-root pairs from an electronic dictionary is feasible. Since Arabic words are looked up in a dictionary using their root form, an electronic dictionary can be parsed to generate the desired list. Figure 13.1 shows an example of a dictionary

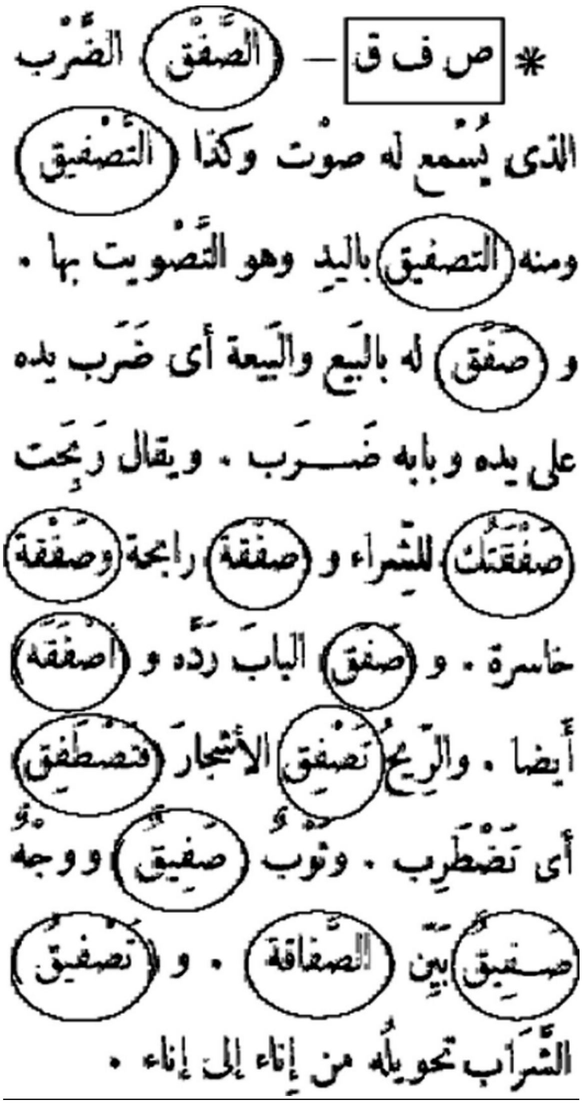

* ص ف ق — الصَّفْقُ الضَّرْبُ
الذي يُسْمَعُ له صوْتٌ وكذا التَّصفيق
ومنه التَّصفيق باليدِ وهو التَّصْويت بها .
و صَفَقَ له بالبَيع والبَيعة أى ضَرَبَ يده
على يده وبابه ضَــرب . ويقال رَبِحَت
صَفْقَتُكَ للشِراء و صَفْقَة رابحة وصَفْقة
خاسرة . و صَفَقَ الباب رَدَّه و أصْفَقَه
أيضا . والرِّيحُ تُصَفِّقُ الأشجار فَتَصْطَفِقُ
أى تَضْطَرِب . وثوب صَفِيقٌ ووجه
صَـفِيقٌ بيِّن الصَّفَاقَة . و تَصْفِيقُ
الشَّرَاب تحويلُه من إناء إلى إناء .

Fig. 13.1. An example of a dictionary entry where the root is inside a rectangle and words derived from the root are circled

entry for a root and words in the entry that are derived from the root. However, some care should be given to throw away dictionary examples and words unrelated to the root. Further, the distribution of words extracted from a dictionary might not be representative of the distribution of words in a text corpus.

3. Automatic construction using a pre-existing morphological analyzer: This process is simple, but requires the availability of an analyzer and a corpus. It has the advantage of producing analyses for large numbers of words extracted from a corpus.

For the purposes of this research, the third method was used to construct the list of word-root pairs. Two lists of Arabic words were analyzed by ALPNET and then the output was parsed to generate the word-root pairs. One list was extracted from a small corpus of Arabic text, called Zad. The Zad corpus is comprised of topics from a 14th century religious book called *Zad Al-Me'ad*. The list contained 9,606 words that ALPNET was able to analyze successfully. The original list was larger, but the words that ALPNET was unable to analyze were excluded. This list will be referred to as the ZAD list. The other list was extracted from the Linguistic Data Consortium (LDC) Arabic corpus containing AFP newswire stories (Graff & Walker, 2001). This list contained 562,684 words. Of these words, ALPNET was able to analyze 270,468 words successfully and failed to analyze 292,216 words. The subsets which ALPNET was able to analyze or failed to analyze will be referred to as the LDC-Pass and LDC-fail lists respectively. ALPNET failed to analyze words belonging to the following subsets:

a. Named entities and Arabized words, which are words that are adopted from other languages: An example of these includes the words krdtš – كردتش (Karadish) and dymwqrATyħ – ديموقراطية (Democracy).
b. Misspelled words
c. Words with roots not in the root list: An example of that is the word jwAĎ – جواظ, (seldom used word meaning pompous).
d. Words with templates not programmed into the finite state transducer. ALPNET uses a separate list of allowed templates for each root. These lists are not comprehensive. An example of that is the word msylmħ – مسيلمة (miniature of mslmħ – مسلمة, a person who is safe or submitting).
e. Words with complex prefixes or suffixes: An example of that is the word bAlxrAfAt – بالخرافات (with the superstitions). ALPNET was able to analyze xrAfAt – خرافات (superstitions) and AlxrAfAt – الخرافات (the superstitions).

It is noteworthy that whenever ALPNET successfully analyzed a word, the generated analyses were always correct.* In the cases when ALPNET provided more than one analysis for a word (thus more than one possible root), all combinations of the word and each of the possible roots were used for training. Although using all the combinations might have distorted the statistics, there was no automatic method for determining which word-root combinations to pick from the possible ones for a word. Also, using all the pairs had the effect of expanding the number of training examples on which the analyzer can be trained.

13.3.2 Training

Sebawai aligns characters in a word and the corresponding root, from the word-root pairs, using regular expressions to determine the prefix, suffix, and stem template. As in Table 13.6, given the pair (wktAbhm – وكتابهم, ktb – كتب), the training module would generate w – و as the prefix, hm – هم as the suffix, and CCAC as the stem template (C's represent the letters in the root). The module increases the observed number of occurrences of the prefix w – و, the suffix hm – هم, and the template "CCAC" by one. The module takes into account the cases where there are no prefixes or suffixes and denotes either of them with the symbol "#".

Table 13.6. The decomposition of the word wktAbhm – **وكتابهم** with root ktb – **كتب**

Parts	Prefix	stem – CCAC – **فعال**				suffix
Root		k – ك	t – ت		b – ب	
Word	w – و	k – ك	t – ت	A – ا	b – ب	hm – هم

After that, the lists of prefixes, suffixes, and templates are read through to estimate probabilities by dividing the occurrence of each by the total number of word-root pairs. The probabilities being calculated are given for character strings S1 and S2 and template T as:

$$P\,(S1\ begins\ a\ word\,,\ S1\ is\ a\ prefix\,) = \frac{(No.\ of words\ with prefix\ S1\,)}{(\ Total No.\ of training\ words\,)}$$

$$P\,(S2\ begins\ a\ word\,,\ S2\ is\ a\ suffix\,) = \frac{(No.\ of words\ with\ suffix\ S2\,)}{(\ Total No.\ of training words\,)}$$

$$P\,(\,T is\ a\ template\,) = \frac{(No.\ of words\ with\ template\ T\,)}{(Total\ No.\ of training\ words\,)}$$

* A random set of a 100 words, for which ALPNET produced analyses, were manually examined to verify their correctness. ALPNET produced correct analyses for every word in the set.

Notice that Sebawai's stems are slightly different from standard stems, in that standard stems may have letters added in the beginning. Linguistics stems are often referred to in Arabic as tfςylAt – تفعيلات or standard stem templates. For example, the template mCCwC has "m" placed before the root making "m" a part of the stem template. However, the training module of Sebawai has no prior knowledge of standard stem templates. Therefore, for the template "mCCwC," the training module treated "m" as a part of the prefix list and the extracted template is "CCwC."

13.3.3 Root Detection

Sebawai detects roots by reading in an Arabic word and attempts to generate feasible prefix-suffix-template combinations. The combinations are produced by generating all possible ways to break the word into three parts provided that the initial part is on the list of prefixes, the final part is in the list of suffixes, and the middle part is at least 2 letters long. The initial and final parts can also be null. Sebawai then attempts to fit the middle part into one of the templates. If the middle part fits into a template, the generated root is checked against a list of possible Arabic roots. The list of Arabic roots, which was extracted from a large Arabic dictionary, contained approximately 10,000 roots (Ibn Manzour, 2006). For example, the Arabic word AymAn – ايمان has the possible prefixes " #" , A – ا, and Ay – ايـ, and the possible suffixes " #" , n – ن, and An – ان. Table 13.7 shows the possible analyses of the word.

The stems that Sebawai deemed as not feasible were AymA – ايما and ym – يم. Although, AymA – ايما is not feasible, ym – يم is actually feasible (comes from the root ymm – يمم). This problem will be addressed in the following subsection. The possible roots are ordered according to the product of the probability that a prefix S1 would be observed, the probability that a suffix S2 would be observed, and the probability that a template T would be used as follows:

*P(root) = P(S1 begins a word, S1 is a prefix) * P(S2 ends a word, S2 is a suffix) * P(T is a template)*

Table 13.7. Possible analyses for the word AymAn – ايمان

Stem	Prefix	Template	Suffix	Root
AymAn – ايمان	#	CyCAC -- فيعال	#	Amn – أمن
ymAn – يمأن	A – ا	CCAC – فعال	#	ymn – يمن
mAn – مأن	Ay – ايـ	CCC – فعل	#	mAn – مأن
Aym – أيم	#	CCC – فعل	An – ان	Aym – أيم
ymA – يمأ	A – ا	CCC – فعل	n – ن	ymA – يمأ

13.3.4 Extensions

- Handling Cases Where Sebawai Failed

As seen above, Sebawai deemed the stem ym – يم not feasible, while in actuality the stem maps to the root ymm – يمم. The stem ym – يم has only two letters because the third letter was dropped. The third letter is often dropped when it is identical to the second letter in the root. Sebawai initially failed to analyze all 2-letter stems. Two letter stems can be derived from roots with long vowels and where the second and third letters are the same.

The Arabic long vowels are A – ا, y – ي, and w – و. The weak letters are frequently substituted for each other in stem templates or dropped all together. For example, the word qAl – قال, has the root qwl –قول , or qyl – قيل, which would make the word mean 'he said' or 'he napped' respectively. Also, the stem f – ف has the root wfy – وفي where the letters w – و and y – ي are missing. To compensate for these problems, two letter stems were corrected by introducing new stems that are generated by doubling the last letter (to produce ymm – يمم from ym – يم) and by adding weak letters before or after the stem. As for stems with a weak middle letter, new stems are introduced by substituting the middle letter with the other weak letters. For example, for qAl – قال, the system would introduce the stems qwl – قول and qyl – قيل. This process over-generates potential roots. From the three potential roots qAl – قال, qwl – قول, and qyl – قيل, qAl – قال is not a valid root and is thus removed (by comparing to the list of valid roots). To account for the changes, the following probabilities were calculated: (a) the probability that a weak letter would be transformed into another weak letter, (b) the probability that a two letter word would have a root with the second letter doubled (such as ymm – يمم), and (c) the probability that a two letter word was derived from a root by dropping an initial or trailing weak letter. The new probability of the root becomes:

*P(root) = P(S1 begins a word, S1 is a prefix) * P(S2 ends a word, S2 is a suffix) * P(T is a template) **

P(letter substitution or letter addition)

- Simplifying Assumptions and Smoothing

The probabilities of stems, suffixes, and templates are assumed to be independent. The independence assumption is made to simplify the ranking, but is not necessarily a correct assumption because certain prefix-suffix combinations are not allowed. As for smoothing the prefix and suffix probabilities, Witten-Bell discounting was used (Jurafsky & Martin, 2000). The smoothing is necessary because many prefixes and suffixes were erroneously produced. This is a result of word-root pair letter alignment errors. Using this smoothing strategy, if a prefix or a suffix is observed only once, then it is removed from the respective list.

- Manual Processing

The list of templates was manually reviewed by an Arabic speaker (the first author) to insure the correctness of the templates. If a template was deemed not correct, it was removed from the list. Checking the list required approximately an hour.

- Particles

To account for particles, which are function words (equivalent to prepositions and pronouns in English) that are typically removed in retrieval applications, a list of Arabic particles was constructed with aid of An-Nahw Ash-Shamil, an Arabic grammar book (Abdul-Al-Aal, 1987). If the system matched a potential stem to one of the words on the particle list, the system indicated that the word is a particle. Note that particles are allowed to have suffixes and prefixes. A complete list of the particles is included in the distribution of Sebawai (Darwish, 2002).

- Letter Normalizations

The system employs a letter normalization strategy in order to account for spelling variations and to ease analysis. The first normalization deals with the letters ي (ya) and ى (ý – *alif maqSūra*). Both are normalized to y – ي. The reason behind this normalization is that there is not a single convention for using y – ي or ý – ى when either appears at the end of a word (Note that Y – ى only appears at the end of words). In the Othmani script of the Holy Qur'an for example, any y – ي is written as ý – ى when it appears at the end of a word. The second normalization is that ء (' – hamza), ا (A – *alif*), آ (Ā – *alif mamduuda*), أ (Â – *alif* with *hamza* on top), ؤ (ŵ – *hamza* on w), إ (Ǎ – *alif* with *hamza* on the bottom), and ئ (ŷ – *hamza* on ya) are normalized to A – ا (A – *alif*). The reason for this normalization is that all forms of hamza are represented in dictionaries as one in root form, and people often misspell different forms of *alif*.

Lastly, words are stripped of all diacritics.

13.3.5 Evaluation

To evaluate Sebawai, it was compared to ALPNET. The two analyzers were compared to test Sebawai's highest ranked analysis and the coverage of Sebawai and ALPNET. The experiments performed involved training Sebawai using the LDC-pass word-root list and testing on the ZAD list. Of the 9,606 words in the ZAD list, Sebawai analyzed 9,497 words and failed on 112. For the roots generated by Sebawai, the top generated root was automatically compared to the roots generated by ALPNET. If the root is on the list, it is considered correct. Using this method, 8,206 roots were considered correct. However, this automatic evaluation had two flaws:

1. The number of Arabic roots in ALPNET's inventory is only 4,500 roots while the number of roots used by Sebawai is more than 10,000. This could lead to false negatives in the evaluation.

2. ALPNET often under-analyzes. For example the word fy – في could be the particle fy – في or could be a stem with the root fyy – فيي. ALPNET only generates the particle fy – في. This also could lead to false negatives.

Therefore manual examination of rejected analyses was necessary. However, due to the large number of rejected analyses, 100 rejected analyses from the automatic evaluation were randomly selected for examination to estimate the shortfall of the automatic evaluation. Of the 100 rejected roots, 46 were correct and 54 were incorrect. Figure 13.2 and Table 13.8 present a summary of the results.

For the LDC-Fail list, Sebawai analyzed 128,169 (43.9%) words out of 292,216 words. To verify the correctness of Sebawai's analyses, 100 analyses were taken at random from the list for manual examination. Of the 100 analyses, 47 were actually analyzed correctly. Extrapolating from the results of the manual examination, Sebawai successfully analyzed approximately 21% of the words in the LDC-Fail list. Table 13.9 presents a summary of the results.

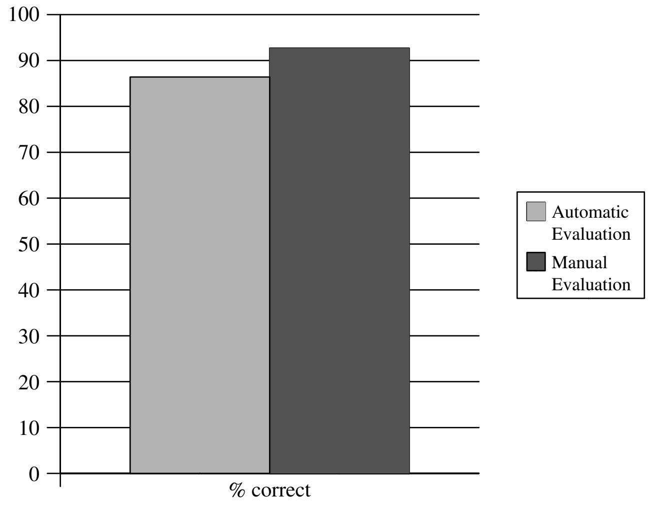


Fig. 13.2. A summary of the results for the correctness of Sebawai's analysis

Table 13.8. A summary of the results for the correctness of Sebawai's analysis

No. of Words	No. of Failures	No. Correct – Automatic Evaluation	No. Correct – Manual Evaluation
9, 606	112 (1.2%)	8,206 (86.4)	8,800 (92.7%)

Table 13.9. Summary of the results for the LDC-Fail list

No. of Words	No. of Analyzed Words	Estimate of Correctly Analyzed
292216	128,169 (43.9%)	58,000 (21%)

The evaluation clearly shows the effectiveness of Sebawai in achieving its two primary goals. Namely, Sebawai provides ranking to the possible analysis and expands the coverage beyond ALPNET. For words that ALPNET was able to analyze, Sebawai ranked the correct analysis first for nearly 93% of the words. Further, the analyzer was able to correctly analyze 21% of the words that ALPNET was unable to analyze. Also, due to the fact that prefixes, suffixes, and templates were derived automatically, Sebawai was developed very rapidly.

13.3.6 Shortcomings of Sebawai

Since analysis is restricted to a choice from a fixed set of roots, Sebawai does not stem Arabized words and named entities. For example, the English word Britain is transliterated as bryTAnyA – بريطانيا. From bryTAnyA – بريطانيا, some of the words that can be generated are bryTAny – بريطاني (British), AlbryTAny – البريطاني (the British), and AlbryTAnyyn – البريطانيين (Englishmen). Sebawai is unable to analyze 1-letter Arabic words, which have 3-letter roots. For example, the word q – ق means "protect (in the form of command)." Since they are very rare, they may not appear in the training set. Sebawai is unable to analyze individual Arabic words that constitute complete sentences. For example, the word AnlzmkmwhA – أنلزمكموها means "will we forcefully bind you to it?" These also are rare and may not appear in a training set.

13.4 Light Stemming

To build the light stemmer, Al-Stem, the lists of prefixes and suffixes generated in the process of training Sebawai and their corresponding probabilities were examined. If a prefix or a suffix had a probability of being an affix above 0.5, it was considered a candidate for building Al-Stem. The list of prefix and suffix candidates was manually examined in consultation with Leah Larkey from the University of Massachusetts at Amherst and some of the affixes were removed based on intuition and knowledge of Arabic.

The final lists of prefixes and suffixes are as follows:

- Prefixes: wAl – وال, fAl – فال, bAl – بال, bt – بت, yt – يت, lt – لت, mt – مت, wt – وت, st – ست, nt – نت, bm – بم, lm – لم, wm – وم, km – كم, fm – فم, Al – ال, ll – لل, wy – وي, ly – لي, sy – سي, fy – في, wA – وا, fA – فا, lA – لا, and bA – با.

- Suffixes: At – ات, wA – وا, wn – ون, wh – وه, An – ان, ty – تي, th – ته, tm – تم, km – كم, hm – هم, hn – هن, hA – ها, yħ – ية, tk – تك, nA – نا, yn – ين, yh – يه, ħ – ة, h – ه, y – ي, A – ا.

Since the result of light stemming may or not be a real stem, no effort was made to evaluate the correctness of the stemming.

13.5 Evaluating Sebawai and Al-Stem in IR

IR experiments were done on the LDC LDC2001T55 collection, which was used in the Text REtrieval Conference (TREC) 2002 cross-language track. For brevity, the collection is referred to as the TREC collection. The collection contains 383,872 articles from the Agence France Press (AFP) Arabic newswire. Fifty topics were developed cooperatively by the LDC and the National Institute of Standards and Technology (NIST), and relevance judgments were developed at the LDC by manually judging a pool of documents obtained from combining the top 100 documents from all the runs submitted by the participating teams to TREC's cross-language track in 2002. The number of known relevant documents ranges from 10 to 523, with an average of 118 relevant documents per topic (Oard & Gey, 2002). This is presently the best available large Arabic information retrieval test collection. The TREC topic descriptions include a title field that briefly names the topic, a description field that usually consists of a single sentence description, and a narrative field that is intended to contain any information that would be needed by a human judge to accurately assess the relevance of a document (Gey & Oard, 2001). Two types of queries were formed from the TREC topics:

a. The title and description fields (td). This is intended to model the sort of statement that a searcher might initially make when asking an intermediary such as a librarian for help.
b. The title field only (t). The title field in recent TREC collections is typically designed as a model for Web queries, which typically contain only 2 or 3 words.

Experiments were performed for each query length with the following index terms:

- w: words.
- ls: lightly stemmed words, obtained using Al-Stem.
- Two ways of obtaining stems:
    - s: top stem found by the Sebawai morphological analyzer (Darwish, 2002).
    - s-ma: top ranking stem found by Sebawai, produced also by ALPNET (Beesley, 1996; Beesley et al., 1989), if ALPNET produced an analysis; otherwise the top stem found by Sebawai. Recall that for words that it can analyze, ALPNET produces an unranked set of analyses, but it fails to produce an analysis more often than Sebawai.
- Two ways of obtaining roots:
    - r: top root found by the Sebawai morphological analyzer (Darwish, 2002).
    - r-ma: top ranking root found by Sebawai, produced also by ALPNET, if ALPNET produced an analysis; otherwise the top root found by Sebawai.

For the experiments, a vector space model information retrieval system called PSE that uses the Okapi BM-25 term weights was used (Robertson & Sparck Jones, 1997). To observe the effect of alternate indexing terms mean uninterpolated average precision was used as the measure of retrieval effectiveness. To determine if the difference between results was statistically significant, a Wilcoxon signed-rank test, which is a nonparametric significance test for correlated samples, was used with p values less than or equal to 0.05 to claim significance.

Figure 13.3 summarizes the results of these runs, and Tables 13.10 and 13.11 present the statistical significance test results for the t and td queries respectively. These results indicate that light stems did not only statistically significantly yield better results than words, but light stems yielded better results than stems and roots. Also, filtering Sebawai's candidates using ALPNET hurt retrieval considerably. This could be due to ALPNET frequently producing analyses that did not include Sebawai's top ranked analysis and thus randomizing the choice of analysis to the detriment of retrieval effectiveness. Recall that ALPNET produces analysis in random order. As indicated earlier, some early work with small test collections (Al-Kharashi & Evens, 1994; Hmeidi et al., 1997) suggested that roots were a better choice than stems, but the experiments presented here found just the opposite. One possible explanation for this is that earlier test collections contained at most a few hundred documents, and scaling up the size of the collection by several orders of magnitude might reward the choice of less ambiguous terms. An alternative explanation is that Sebawai's morphological analysis might not be sufficiently accurate.

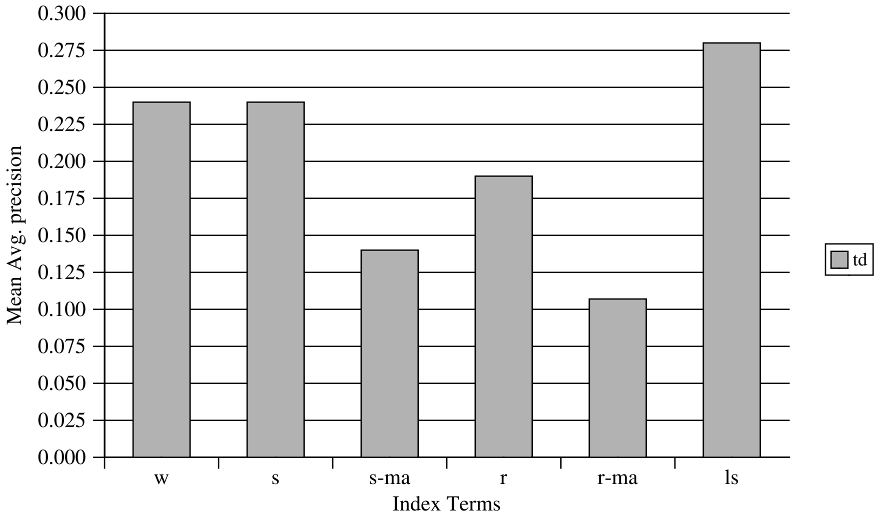


Fig. 13.3. Comparing index terms on the TREC collection for title only (t) and title+ description queries

Table 13.10. Comparing index terms using title only queries on the TREC collection using the p values of the Wilcoxon signed-ranked test – the cell is shaded if difference is statistically significant

w	*S*	*r*	*ls*	
	0.24	0.39	0	*w*
		0.01	0	*s*
			0	*r*
				ls

Table 13.11. Comparing index terms using title+ description queries on the TREC collection using the p values of the Wilcoxon signed-ranked test – the cell is shaded if difference is statistically significant

w	*s*	*r*	*ls*	
	0.92	0.38	0	*w*
		0.02	0	*s*
			0	*r*
				ls

13.6 Conclusion

This chapter presented an adaptation of existing Arabic morphological analysis techniques to make them suitable for the requirements of IR applications by leveraging corpus statistics. As a result of the adaptation, Sebawai, a new freely distributable Arabic morphological analyzer, and Al-Stem, an Arabic light stemmer were constructed. Al-Stem was used extensively by many research groups for comparative Arabic IR evaluations (Chen & Gey, 2002; Darwish & Oard 2002a; Fraser et al., 2002; Larkey et al., 2002; Oard & Gey, 2002).

The morphological analysis techniques described here could benefit from better word models that incorporate statistics on legal prefix-suffix combination (currently Sebawai assumes independence between prefixes and suffixes) and sentence level models that incorporate context to improve ranking. All these techniques can potentially be used in developing morphological analyzers for other morphologically similar languages such as Hebrew.

References

Abdul-Al-Aal, A. (1987). *An-Nahw Ashamil*. Cairo, Egypt: Maktabat Annahda Al-Masriya.

Abu-Salem, H., Al-Omari, M. & Evens, M. (1999). Stemming Methodologies Over Individual Query Words for Arabic Information Retrieval. *Journal of the American Society for Information Science and Technology*, *50*(6), 524–529.

Ahmed, M. (2000). *A Large-Scale Computational Processor of Arabic Morphology and Applications*. Faculty of Engineering, Cairo University, Cairo, Egypt.

Aljlayl, M., Beitzel, S., Jensen, E., Chowdhury, A., Holmes, D., Lee, M., Grossman, D. & Frieder, O (2001). IIT at TREC-10. In *Proceedings of the Tenth Text REtrieval Conference* (pp.265–274), Gaithersburg, MD. http://trec.nist.gov/pubs/trec10/papers/IIT-TREC10.pdf

Al-Kharashi, I. & Evens, M. (1994). Comparing Words, Stems, and Roots as Index Terms in an Arabic Information Retrieval System. *Journal of the American Society for Information Science and Technology, 45*(8), 548–560.

Antworth, E. (1990). PC-KIMMO: a two-level processor for morphological analysis. In *Occasional Publications in Academic Computing*. Dallas, TX: Summer Institute of Linguistics.

Beesley, K. (1996). Arabic Finite-State Morphological Analysis and Generation. In *Proceedings of the International Conference on Computational Linguistics* (COLING-96, vol. 1, pp. 89–94).

Beesley, K., Buckwalter, T. & Newton, S. (1989). Two-Level Finite-State Analysis of Arabic Morphology. In *Proceedings of the Seminar on Bilingual Computing in Arabic and English,* Cambridge, England.

Chen, A. & Gey, F. (2001). Translation Term Weighting and Combining Translation Resources in Cross-Language Retrieval. In *Proceedings of the Tenth Text REtrieval Conference* (pp. 529–533), Gaithersburg, MD. http://trec.nist.gov/pubs/trec10/papers/berkeley_trec10.pdf

Chen, A. & Gey, F. (2002). Building an Arabic Stemmer for Information Retrieval. In *Proceedings of the Eleventh Text REtrieval Conference*, Gaithersburg, MD. http://trec.nist.gov/pubs/trec11/papers/ucalberkeley.chen.pdf

Darwish, K. (2002). Building a Shallow Arabic Morphological Analyzer in One Day. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages* (pp. 1–8). Philadelphia, PA.

Darwish, K. & Oard, D. (2002a). CLIR Experiments at Maryland for TREC 2002: Evidence Combination for Arabic-English Retrieval. In *Proceedings of the Eleventh Text REtrieval Conference*, Gaithersburg, MD. http://trec.nist.gov/pubs/trec11/papers/umd.darwish.pdf

Darwish, K. & Oard, D. (2002b). Term Selection for Searching Printed Arabic. In *Proceedings of the Special Interest Group on Information Retrieval Conference* (SIGIR, pp. 261–268), Tampere, Finland.

Fraser, A., Xu, J. & Weischedel, R. (2002). TREC 2002 Cross-lingual Retrieval at BBN. In *Proceedings of the Eleventh Text REtrieval Conference*, Gaithersburg, MD. http://trec.nist.gov/pubs/trec11/papers/bbn.xu.cross.pdf

Gey, F. & Oard, D. (2001). The TREC-2001 Cross-Language Information Retrieval Track: Searching Arabic Using English, French or Arabic Queries. In *Proceedings of the Tenth Text REtrieval Conference,* Gaithersburg, MD. http://trec.nist.gov/pubs/trec10/papers/clirtrack.pdf

Goldsmith, J. (2000). *Unsupervised Learning of the Morphology of a Natural Language*. Retrieved from http://humanities.uchicago.edu/faculty/goldsmith/

Graff, D. & Walker, K. (2001). *Arabic Newswire Part 1.* Linguistic Data Consortium, Philadelphia. LDC catalog number LDC2001T55 and ISBN 1-58563-190-6.

Hmeidi, I., Kanaan, G. & Evens, M. (1997). Design and Implementation of Automatic Indexing for Information Retrieval with Arabic Documents. *Journal of the American Society for Information Science and Technology, 48*(10), 867–881.

Ibn Manzour (2006). *Lisan Al-Arab*. Retrieved from http://www.muhaddith.org/

Jurafsky, D. & Martin, J. (2000). *Speech and Language Processing*. Saddle River, NJ: Prentice Hall.

Kiraz, G. (1998). Arabic Computational Morphology in the West. In *Proceedings of The 6th International Conference and Exhibition on Multi-lingual Computing,* Cambridge.

Koskenniemi, K. (1983). *Two Level Morphology: A General Computational Model for Word-form Recognition and Production.* Department of General Linguistics, University of Helsinki.

Larkey, L., Allen, J., Connell, M. E., Bolivar, A. & Wade, C. (2002). UMass at TREC-2002: Cross Language and Novelty Tracks. In *Proceedings of the Eleventh Text REtrieval Conference*, Gaithersburg, MD. http://trec.nist.gov/pubs/trec11/papers/umass.wade.pdf

Larkey, L., Ballesteros, L. & Connell, M. (2002). Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis. In *Proceedings of the Special Interest Group on Information Retrieval* (SIGIR, pp. 275–282), Tampere, Finland.

Mayfield, J., McNamee, P., Costello, C., Piatko, C. & Banerjee, A. (2001). JHU/APL at TREC 2001: Experiments in Filtering and in Arabic, Video, and Web Retrieval. In *Proceedings of the Tenth Text REtrieval Conference,* Gaithersburg, MD. http://trec.nist.gov/pubs/trec10/papers/jhuapl01.pdf

Oard, D. & Gey, F. (2002). *The TREC 2002 Arabic/English CLIR Track*. In *Proceedings of the Eleventh Text REtrieval Conference*, Gaithersburg, MD. http://trec.nist.gov/pubs/trec11/papers/OVERVIEW.gey.ps.gz

Robertson, S. & Sparck Jones, K. (1997). *Simple Proven Approaches to Text Retrieval.* Cambridge University Computer Laboratory.

Xu, J., Fraser, A. & Weischedel, R. (2001). Cross-Lingual Retrieval at BBN. In *Proceedings of the Tenth Text REtrieval Conference* (pp. 68–75), Gaithersburg, MD. http://trec.nist.gov/pubs/trec10/papers/BBNTREC2001.pdf

14

Arabic Morphological Representations for Machine Translation

Nizar Habash

Center for Computational Learning Systems, Columbia University
habash@cs.columbia.edu

Abstract: Arabic has a very rich morphology characterized by a combination of templatic and affixational morphemes, complex morphological rules, and a rich feature system. This complexity makes working with Arabic as a source of target language in machine translation (MT) a challenge for two reasons. First, it is not clear what the right representation is for two reasons. First, it is not clear what the right representation is for Arabic words given a specific MT approach or system. And secondly, there are many MT-relevant resources for Arabic morphology, lexicography and syntax (e.g., morphological analyzers, dictionaries and treebanks) that adopt various representations that are not necessarily compatible with each other. The result is that for MT researchers, there is a need to experiment with and to relate multiple representations used by different resources or components to each other within a single system. In this chapter, we describe different Arabic morphological representations used by MT-relevant natural language processing resources and tools and we discuss their usability in different MT approaches. We also present a common framework for relating different levels of representations to each other

14.1 Introduction

Arabic has a very rich morphology characterized by a combination of templatic and affixational morphemes, complex morphological rules, and a rich feature system. This complexity makes working with Arabic as a source or target language in Machine Translation (MT) a challenge for two reasons. First, it is not clear what the right representation is for Arabic words given a specific MT approach or system. It is not even clear whether the same representation is optimal for every component in an MT system, e.g., word alignment versus decoding in statistical MT or parsing versus structural transfer in symbolic MT. Secondly, there are many MT-relevant resources for Arabic morphology, lexicography and syntax (e.g., morphological analyzers, dictionaries and treebanks) that adopt various representations that are not necessarily compatible with each other. For example, dictionaries use the notion of a *lexeme* that is different from the root/pattern/vocalism and stem/affix representations used by many morphological analyzers. And statistical parsers can be content with a minimally tokenized inflected undiacritized word as the proper level of representation for Arabic, which is different from input text and also potentially different from later processing steps. The result is that for MT researchers, there is a need to experiment with and to relate multiple representations used by different resources or components to each other within a single system. This challenge has different implications for research in statistical MT, symbolic MT or hybrid approaches to MT.

In this chapter, we describe different Arabic morphological representations used by MT-relevant Natural Language Processing (NLP) resources and tools and we discuss their usability in different MT approaches. We also present a common framework for relating different levels of representations to each other. We motivate the lexeme-and-feature level of representation as a common representation to analyze to. From that representation, we can regenerate to other desirable shallower representation. This framework allows for easy navigation between representations used by different resources. It also allows for exploring the effect of using different representations in MT. The interaction between analysis and generation makes this framework direction-independent, i.e., useful for working with Arabic as a source or target MT language. Finally, we describe and evaluate ALMORGEANA, a large-scale system for analysis and generation from/to the lexeme-and-feature representation. We also discuss how to use it to relate different morphological representations for Arabic.

Section 14.2 introduces different representations in Arabic morphology.[1] Section 14.3 discusses the role of morphological representations in different approaches to MT. Section 14.4 and Section 14.5 describe ALMORGEANA and how it can be used for navigating among different morphological representations, respectively.

14.2 Representations of Arabic Morphology

In discussing representations of Arabic morphology, it is important to separate two different aspects of morphemes: *type* versus *function*. Morpheme *type* refers to the different formal kinds of morphemes and their interactions with each other. A distinguishing feature of Arabic (in fact, Semitic) morphology is the presence of templatic morphemes in addition to affixational morphemes. Morpheme *function* refers to the distinction between derivational morphology and inflectional morphology. These two aspects, type and function, are independent, i.e., a morpheme type does not determine its function and vice versa. This independence complicates the task of deciding on the proper representation of morphology in different NLP resources and tools. This section introduces these two aspects and their interactions in more detail.

[1] Additional discussions of Arabic morphological phenomena are presented in Chapter 3 and in the four chapters in Part 2 of this book. See Chapter 15 for a discussion of Arabic generation in the context of MT.

14.2.1 Morpheme Type: Templatic vs. Affixational

Arabic has three categories of morphemes: templatic morphemes, affixational morphemes, and Non-Templatic Word Stems (NTWS). Templatic morphemes come in three types that are equally needed to create a templatic word stem: roots, patterns and vocalisms. The root morpheme is a sequence of three, four or five consonants (termed *radicals*) that signifies some abstract meaning shared by all its derivations. For example, the words كَتَب *katab* 'to write', كَاتِب *kaAtib* 'writer', and مَكْتُوب *maktuwb* 'written' all share the root morpheme (كتب) *ktb* 'writing-related'. The pattern morpheme is an abstract template in which roots and vocalisms are inserted.[2] For example, the verbal pattern *tV1V22V3* indicates that a non-root consonant (t) is added and that the second root radical is doubled. The vocalism morpheme specifies which vowels to use with a pattern. A word stem is constructed by interleaving a root, a pattern and a vocalism. For example, the word stem كَتَب *katab* 'to write' is constructed from the root كتب *ktb*, the pattern *1V2V3* and the vocalism *aa*.

Arabic affixes can be prefixes such as +سَ *sa*+ 'will/[future]', suffixes such as ُونَ+ +*uwna* '[masculine plural]' or circumfixes such as نَ++تَ *ta*++*na* '[subject imperfective 2nd person feminine plural]'. Some of the affixes are clitics, such as the conjunction +وَ *wa*+ 'and', the preposition (+لِ *li*+ 'to/for', and the pronominal object/possessive clitics (e.g. هَا+ +*haA* 'her/it/its'). Others are bound morphemes.

Finally, NTWS are word stems that are not derivable from templatic morphemes. They tend to be foreign names (e.g., وَاشِنْطُن *waAšinTun* 'Washington').

An Arabic word is constructed by first creating a word stem from templatic morphemes or using a NTWS, to which affixational morphemes are then added. For example, the word وَسَيَكْتُبُونَهَا *wasayaktubuwnahaA* has two prefixes, one circumfix and one suffix in addition to a root, a pattern and a vocalism:

```
(1)  wa+  sa+   y+    [ktb+V12V3+au] +uwna  +haA
     and+ will+ 3rd+ write           +plural +it
     'And they will write it'
```

The process of combining morphemes can involve a number of phonological, morphological and orthographic rules that modify the form of the created word; it is not always a simple interleaving and concatenation of its morphemic components. One example is the feminine morpheme, ة+ +*ħ (ta marbuta)*, which is turned into ت+ +*t* when followed by a possessive clitic: أَمِيرَة+هُم *Âamiyraħu+hum* 'princess+their' is realized as أَمِيرَتُهُم *Âamiyratuhum* 'their princess'. Another example is the deletion of the Alif (ا) of the definite article +ال *Al*+ when preceded

[2] In this chapter, numbers, (1, 2, 3, 4, or 5), are used in a pattern to indicate radical position as opposed to the common practice in the literature of using the symbol *C*. The symbol *V* is used to indicate a vocalism position.

by the preposition +لِ *l*+ 'for'. For example, لِ+ال+بَيت *li+Al+bayt* 'for+the+house' is realized as لِلبَيت *lilbayt* 'for the house'. These rules clearly complicate the process of analyzing and generating Arabic words.

14.2.2 Morpheme Function: Derivational vs. Inflectional

The distinction between derivational and inflectional morphology in Arabic is similar to that in other languages. Derivational morphology is concerned with creating new words from other words, a process in which the core meaning of the word is modified. For example, the Arabic كَاتِب *kaAtib* 'writer' can be seen as derived from the root كتب *ktb* the same way the English *writer* can be seen as a derivation from *write*. Although compositional aspects of derivations do exist, the derived meaning is often idiosyncratic. For example, the *masculine* noun مَكْتَب *maktab* 'office/bureau/agency' and the *feminine* noun مَكْتَبَة *maktabaħ* 'library/bookstore' are derived from the root كتب *ktb* 'writing-related' with the pattern+vocalism *ma12a3*, which indicates location. The exact type of the location is thus idiosyncratic, and it is not clear how the nominal gender difference can account for the semantic difference.

On the other hand, in inflectional morphology, the core meaning of the word remains intact and the extensions are always predictable. For example, the semantic relationship between كَاتِب *kaAtib* 'writer' and كُتَّاب *kut~aAb* 'writers' maintains the sense of the kind of person described, but only varies the number. The change in number in this example is accomplished using templatic morphemes (pattern and vocalism change). This form of plural construction in Arabic is often called "broken plural" to distinguish it from the strictly affixational "sound plural" (e.g. كَاتِب+ات *kaAtib+aAt* 'writers [fem]').

Broken plurals are one example highlighting the independence of morpheme type from morpheme function: templatic morphemes can be derivational or inflectional, with the exception of the roots, which are always derivational. Similarly, the majority of affixational morphemes are inflectional but there are some affixational derivational morphemes: the adjective كُتُبِيّ *kutubiy~* 'book-related' is derived from the noun كُتُب *kutub* 'books' using the affix +ِيّ *+iy~*.

14.2.3 Arabic Morphological Representations

Given the variability in the relationship between morpheme type and function in addition to the presence of phonological, morphological, and orthographic adjustment phenomena, there are many ways to represent Arabic words in terms of their morphological units. Table 14.1 illustrates some of these possible representations using the example وَلِكَتَبَتِهِم؟ *walikatabatihim?* 'and for their writers?'.

There are many variations among these different representations: (a.) whether they address inflectional/derivational phenomena, templatic/affixational phenomena

Table 14.1. Morphological representations of Arabic words

Representation	Example	Found where?
Natural Token	wlktbthm?	naturally occurring text
Simple Token	wlktbthm ?	common preprocessing for NLP [29]
Segmentation	wl+ ktb +thm ?	[11, 12]
	w+ l+ ktbt +hm ?	[51, 21]
	w+ l+ ktb +t +hm ?	[40, 39]
Normalized Segmentation	w+ l+ ktb*ħ* +hm ?	Penn Arab Treebank [41, 29]
	w+ l+ ktb +*ħ* +hm ?	[59, 29]
Templatic Segmentation	w+ l+ ktb+1V2V3a*ħ*+aa +hm ?	[33]
Morphemes and Features	w+/CONJ l+/PREP kataba*ħ* +hm/P:3MP ? ktb&CaCaCa*ħ* w+ l+ +P:3MP ? ktb +PL w+ l+ +GEN +P:3MP ?	[6, 11, 12, 29]
Lexeme and Features	[kAtib w+ l+ PL P:3MP] [?]	ALMORGEANA, [27], dictionaries (lexeme only)

or both, (b.) whether they preserve or resolve ambiguity,[3] and (c.) which degree of abstraction from allomorphs (actual form of morpheme after applying various adjustment rules) they use. And since any subset (or all) of the morphemes can be separated from the word and/or be normalized, there is a very large space of possible specific representations to select from.

The *natural token* refers to the way Arabic words appear in actual text where they are undiacritized and segmented only using white space. Punctuations, for example, could be attached to the word string in this representation. All naturally occurring Arabic text is in this representation. *Simple tokenization* separates punctuation but maintains the morphological complexity of the Arabic word tokens. There is no change in ambiguity compared to the natural token.

Segmentation is the simplest way to dissect an Arabic word. It is strictly defined here to exclude any form of orthographic, morphological or phonological normalization. Segmentation splits up the letters into segments that correspond to clusters of a stem plus one or more affixational morphemes. There are many ways to segment an Arabic word as Table 14.1 shows. Segmentation can select a subset of analyses of a word. For example, segmenting للجنة *lljnħ* into *l+l+jnħ* (*li+l+jan~aħ* 'to Paradise'

[3] This discussion does not address the issue of morphological disambiguation, which is outside the scope of this chapter [26, 54, 30].

or *li+l+jin~aħ* ‘to insanity/mania’) is selecting a subset of analyses excluding *l+ljnħ* (*li+lajnaħ* ‘to a committee’ or *li+l~ajnaħ* ‘to the committee’).

Normalized segmentation abstracts away from some of the adjustment phenomena discussed earlier. In the example in Table 14.1, the form of the segmented word stem is *ktbħ* not *ktbt*. Normalization disambiguates the unnormalized segmented form *ktbt* (‘he/she/you[sg.] wrote’ or ‘writers’). The Penn Arabic Treebank [41] uses a normalized segmentation that breaks up a word into four regions: conjunction, particle, normalized word stem and pronominal clitic.

Templatic segmentation is a deeper level of segmentation that involves normalization by definition. Here, the root, pattern and vocalism are separated. Up to this level of representation, the tokens are driven by a templatic/affixational view of morphology rather than a derivational/inflectional view. The introduction of features at the next level of representation, *morphemes and features*, abstracts away from different morphemes that at an underlying level signify the same feature. For example, The affixational morphemes يـ++ـُونَ *y++uwna*, يـ++ـُوا *y++uwA* and +ـُوا *+uwA* all realize the third person masculine plural subject for different verb aspect/mood combinations. There are many different degrees to the transition from morphemes to features. A combination of both is often used.

The final representation is *lexeme and features*. The lexeme can be defined as an abstraction over a set of word forms differing only in inflectional morphology. The lexeme itself captures a specific meaning that does not change with inflectional variations. The traditional citation form of a lexeme used in dictionaries is the perfective third person masculine singular for verbs and the singular masculine form for nouns and adjectives. If there is no masculine form, the feminine singular is used. As such, the Lexeme [كَاتِب] [kaAtib] ‘writer’ normalizes over all the different inflectional forms of كَاتِب *kaAtib* such as كَاتِبَان *kaAtibaAn* ‘two writers’, كَتَبَة *katabaħ* ‘writers’, and كَاتِبَة *kaAtibaħ* ‘female writer’. Lexemes as opposed to stems provide a desirable level of abstraction that is to a certain degree language independent for applications such as MT. Lexemes are also less abstract than roots and patterns which tend to be too vague semantically and derivationally unpredictable, making them less useful in practice for MT.

The next section discusses how these different levels of representation interact with different MT approaches.

14.3 Morphological Representations for Machine Translation

In statistical approaches to MT, a translation model is trained on word-aligned [46] parallel text of source and target languages [9, 10, 35, 37, 36]. The translation model is then used to generate multiple target language hypotheses from the source language input. The target hypotheses are typically ranked using a log-linear combination of a variety of features [45]. Statistical MT has been quite successful in producing good quality[4] MT on the genre it is trained on in much faster time than symbolic approaches. For statistical MT, in principle, it doesn't matter what level of morphological representation is used as long as the input is on the same level as the data used in training. Practically however there are certain concerns with issues such as sparsity, ambiguity, language-pair differences in morphological complexity, and training-data size. Shallower representations such as simple tokenization tend to maintain distinctions among morphological forms that might not be relevant for translation, thus increasing the sparsity of the data. This point interacts with the MT language pair: for example, normalizing subject inflections of Arabic verbs when translating to a morphologically poor language like English might be desirable since it reduces sparsity without potentially affecting translation quality. If the target language is morphologically rich, such as French, that would not be the case. This, of course, may not be a problem when large amounts of training data are available. Additionally, transforming the training text to deeper representations comes at a cost since selecting a deeper representation involves some degree of morphological disambiguation, a task that is typically neither cheap nor foolproof [26].

The anecdotal intuition in the field of statistical MT is that reduction of morphological sparsity often improves translation quality. This reduction can be achieved by increasing training data or via morphologically-driven preprocessing [22]. Recent investigations of the effect of morphology on MT quality focused on morphologically rich languages such as Catalan [49], Czech [22], German [43], Serbian [49] and Spanish [34, 49]. These studies examined the effects of various kinds of tokenization, lemmatization and part-of-speech (POS) tagging and showed a positive effect on MT quality.

Specifically for Arabic, Lee [39] investigated the use of automatic alignment of POS-tagged English and affix-stem segmented Arabic to determine appropriate tokenizations of Arabic. Her results show that morphological preprocessing helps but only for the smaller corpora sizes she investigated. As size increases, the benefits diminish. Habash and Sadat [29, 52] reached similar conclusions on a much larger set of experiments including multiple preprocessing schemes reflecting different levels of morphological representation and multiple techniques for disambiguation/tokenization. Two of their techniques used the ALMORGEANA system described later in this chapter. They showed that specific preprocessing decisions can have a positive effect when decoding text with a different genre than that of the training data (in essence another form of data sparsity). They also demonstrated gains in MT quality through combination of different preprocessing schemes. Additional similar results were reported using specific preprocessing schemes and techniques [59, 51, 21, 44].

Research in the use of different morphological representations of Arabic in Example-based MT, a corpus-based approach related to statistical MT [55, 14], is promising, at least in terms of improved coverage of training examples [48].

[4] The question of how to judge the quality of MT, i.e., MT Evaluation, is outside the scope of this chapter. Currently, the most accepted yet still controversial approaches are automatic, e.g., BLEU [47, 13] and METEOR [5].

Finally, the newest addition to research on morphology within phrase-based statistical MT is Moses, a decoder for factored [8] phrase-based translation models. Moses allows using a mix of different levels of morphological representation.[5] At the time of writing this chapter, no work on Arabic factored translation models have been done.

In symbolic approaches to MT, such as transfer-based or interlingual MT, linguistically motivated rules (morphological, syntactic and/or semantic) are manually or semi-automatically constructed to create a system that translates the source language into the target language [20]. Symbolic MT approaches tend to capture more abstract generalities about the languages they translate between compared to statistical MT. This comes at a cost of being more complex than statistical MT, involving more human effort, and depending on already existing resources for morphological analysis and parsing. This dependence on already existing resources highlights the problem of variation in morphological representations for Arabic. In a typical situation, the input/output text of an MT system is in natural or simplified tokenization. But, a statistical parser (such as [16] or [7]) trained out-of-the-box on the Penn Arabic Treebank assumes the same kind of tokenization (4-way normalized segments) used by the treebank. This means that a separate tokenizer is needed to convert input text to this representation [19, 26]. Moreover, the output of such a parser, being in normalized segmentation, will not contain morphological information such as features or lexemes that are important for translation: Arabic-English dictionaries use lexemes and proper translation of features, such as number and tense, requires access to these features in both source and target languages. As a result, additional conversion is needed to relate the normalized segmentation to the lexeme and feature levels. Of course, in principle, the treebank and parser could be modified to be at the desired level of representation (i.e., lexeme and features). But this can be a rather involved task for researchers interested in MT. We are aware of the following published research on Arabic symbolic MT: [4, 53] (within the transfer approach) and [58, 56, 1] (within the interlingua approach). Given the inhibiting costs of building large scale symbolic MT system, they tend to be developed by commercial institutions, which are less inclined to publicize their trade secrets.[6]

Finally, the current hybridization direction in the field of MT is interested in exploring statistical and symbolic combinations of resources and tools within and beyond the level of morphology. Some hybrids rooted in statistical MT include syntactic information as part of the preprocessing phase [17], the decoding phase [50] or the n-best rescoring phase [45]. Such approaches will share challenges relevant to both statistical and symbolic MT when extended to Arabic. A detailed discussion of such challenges are presented in the context of extending a Generation Heavy MT system, a hybrid approach rooted in symbolic MT [23], to Arabic [25].

[5] Moses was developed during the 2006 summer workshop at Johns Hopkins University as an enhancement to Pharaoh [36]. See http://www.clsp.jhu.edu/ws2006/groups/ossmt/ and http://www.statmt.org/moses/ for more details.

[6] Two of the top Arabic MT companies using rule-based MT systems are Apptek (http://www.apptek.com/) and Sakhr (http://www.sakhr.com/).

In the next section, we describe ALMORGEANA (Arabic Lexeme-base MORphological GEnerator/ANAlyzer). ALMORGEANA is a morphological analysis and generation system built on top of the Buckwalter analyzer databases, which are at a different level of representation (3-way segmentation). Being an analysis and generation system, it can be used with MT systems analyzing or generating Arabic. ALMORGEANA relates the deepest level of representation (lexeme and features) to the shallowest (simple tokenization).[7] This wide range together with bidirectionality (analysis/generation) allows using ALMORGEANA to navigate between different levels of representations as will be discussed in Section 14.5. Morphological disambiguation, or the selection of an analysis from a list of possible analyses, is a different task that is out of the scope of this chapter although it is quite relevant to MT [26, 54, 30].

14.4 ALMORGEANA

ALMORGEANA is a large-scale lexeme-based Arabic morphological analysis and generation system.[8] ALMORGEANA uses the databases of the Buckwalter Arabic morphological analysis system with a different engine focused on generation from and analysis to the lexeme-and-feature level of representation. The building of ALMORGEANA didn't just involve the reversal of the Buckwalter analyzer engine, which only focuses on analysis, but also extending it and its databases to be used in a lexeme-and-feature level of representation for both analysis and generation.

The next section reviews other efforts on morphological analysis and generation in Arabic. Section 14.4.2 introduces the Buckwalter analyzer's database and engine. Section 14.4.3 describes the different components of ALMORGEANA. An evaluation of ALMORGEANA is discussed in Section 14.4.4.

14.4.1 Morphological Analysis and Generation

Arabic morphological analysis has been the focus of researchers in natural language processing for a long time. This is due to features of Arabic Semitic morphology such as optional diacritization and templatic morphology. Numerous forms of morphological analyzers have been built for a wide range of application areas from Information Retrieval (IR) to MT in a variety of linguistic theoretical contexts [3, 2, 6, 11, 12, 18, 33, 27].

Arabic morphological generation, by comparison, has received little attention although the types of problems in generation can be as complex as in analysis.

[7] Going to natural tokenization is a trivial step where, for example, punctuation marks are attached to preceding words.

[8] A previous publication about ALMORGEANA focused on the generation component of the system which was named Aragen [24].

Finite-State Transducer (FST) approaches to morphology [38] and their extensions for Arabic such as the Xerox Arabic analyzer [6] are attractive for being generative models. However, a major hurdle to their usability is that lexical and surface levels are very close [32]. Thus, generation from the lexical level is not useful to many applications such as symbolic MT where the input to a generation component is typically a lexeme with a feature list. A solution to this problem was proposed by [32], which involved composition of multiple FSTs that convert input from a deep level of representation to the lexical level. However, there are still many restrictions on the order of elements presented as input and their compatibility.[9] The MAGEAD (Morphological Analysis and Generation for Arabic and its Dialects) system attempts to design an end-to-end lexeme-and-features to surface FST-based system for Arabic [28]. As of the time of the writing of this chapter, MAGEAD's coverage is limited to verbs in Modern Standard Arabic and Levantine Arabic. The only work on Arabic morphological generation that focuses on generation issues within a lexeme-based approach is done by [15, 57]. Their work uses transformational rules to address the issue of stem change in various prefix/suffix contexts. Their system is a prototype that lacks in large-scale coverage.

There are certain desiderata that are expected from a morphological analysis/generation system for any language. These include (1) coverage of the language of interest in terms of both lexical coverage (large scale) and coverage of morphological and orthographic phenomena (robustness); (2) the surface forms are mapped to/from a deep level of representation that abstracts over language-specific morphological and orthographic features; (3) full reversibility of the system so it can be used as an analyzer or a generator; (4) usability in a wide range of natural language processing applications such as MT or IR; and finally, (5) availability for the research community. These issues are essential in the design of ALMORGEANA for Arabic morphological analysis and generation. ALMORGEANA[10] is a lexeme-based system built on top of a publicly available large-scale database, Buckwalter's lexicon for morphological analysis.

14.4.2 Buckwalter Morphological Analyzer

The Buckwalter morphological analyzer uses a concatenative lexicon-driven approach where morphotactics and orthographic rules are built directly into the lexicon itself instead of being specified in terms of general rules that interact to realize the output [11, 12]. The system has three components: the lexicon, the compatibility tables and the analysis engine. An Arabic word is viewed as a concatenation of three regions, a prefix region, a stem region and a suffix region. The prefix

[9] Other work on using FSTs designed for analysis in generation is discussed in [42].

[10] The ALMORGEANA engine can be freely downloaded under an OpenSource license for research purposes from http://www.ccls.columbia.edu/cadim/resources.html. The lexical databases need to be acquired independently from the Linguistic Data Consortium (LDC) as part of the Buckwalter Arabic Morphological Analyzer [11, 12].

وَ /wa	Pref-Wa	*and*
بِـ /bi	NPref-Bi	*by/with*
وَبِـ /wabi	NPref-Bi	*and + by/with*
الـ /Al	NPref-Al	*the*
بِالـ /biAl	NPref-BiAl	*with/by + the*
وَبِالـ /wabiAl	NPref-BiAl	*and + with/bythe*
ـَة /ap	NSuff-ap	*[fem.sg.]*
ـَتَانِ /atAni	NSuff-atAn	*two*
ـَتَيْنِ /atayoni	NSuff-tayn	*two*
ـَتَاهُ /atAhu	NSuff-atAh	*his/its two*
ـات/At	NSuff-At	*[fem.pl.]*

;;1_ كَتَب/katab-u_1		
كَتَب/katab	PV	*write*
كْتُب/kotub	IV	*write*
كُتِب/kutib	PV_Pass	*be written*
كْتَب/kotab	IV_Pass_yu	*be written*
;;1_ كِتَاب/kitAb_1		
كِتَاب/kitAb	Ndu	*book*
كُتُب/kutub	N	*books*
;;1_ كِتَابَة/kitAbap_1		
كِتَاب/kitAb	Nap	*writing*

Fig. 14.1. Some Buckwalter lexical entries

and suffix regions can be null. Prefix and suffix lexicon entries cover all possible concatenations of Arabic prefixes and suffixes, respectively. For every lexicon entry, a morphological compatibility category, an English gloss and occasional Part-Of-Speech (POS) data are specified. Stem lexicon entries are clustered around their specific lexeme, which is not used in the analysis process. Figure 14.1[11] shows sample entries: the first six in the left column are prefixes; the rest in that column are suffixes; the right column contains seven stems belonging to three lexemes. The stem entries also include English glosses which allows the lexicon to function as a dictionary. However, the presence of inflected forms, such as passives and plurals among these glosses makes them less usable as lexemic translations.

Compatibility tables specify which morphological categories are allowed to co-occur. For example, the morphological category for the prefix conjunction وَ/`wa` *wa+* 'and', `Pref-Wa`, is compatible with all noun stem categories and perfect verb stem categories. However, `Pref-Wa` is not compatible with imperfective verb stems because they must contain a subject prefix. Similarly, the stem كِتَاب/`kitAb` *kitaAb* of the the lexeme 1_كِتَاب/`kitAb_1` *kitaAb* 'book' has the category (`Ndu`), which is not compatible with the category of the feminine marker ة/`ap` *ah̄*: `NSuff-ap`. The same stem, كِتَاب/`kitAb` *kitaAb*, appears as one of the stems of the lexeme 1_كِتَابَة/`kitAbap_1` *kitaAbah̄* 'writing' with a category that *requires* a suffix with the feminine marker. Cases such as these are quite common and pose a challenge to the use of stems as tokens since they add unnecessary ambiguity.

The analysis algorithm is rather simple since all of the hard decisions are coded in the lexicon and the compatibility tables: Arabic words are segmented into all possible sets of prefix, stem and suffix strings. In a valid segmentation, the three strings exist in the lexicon and are three-way compatible (prefix-stem, stem-suffix and prefix-suffix).

[11] The Buckwalter transliteration is preserved in examples of Buckwalter lexicon entries (see Chapter 2).

14.4.3 ALMORGEANA Components

14.4.3.1 Input/Output

In generation mode, the input to ALMORGEANA is a *feature-set*, a set of lexeme and features from a closed class of inflectional phenomena. The output of generation is one or more word strings in simple tokenization. In analysis mode, the input is the string and the output a set of possible feature-sets. The features in a feature-set include number, gender and case inflections, which do appear in other languages, but also prefix conjunctions and prepositions that are written as part of the word in Arabic orthography. Table 14.2 lists the different features and their possible values.

The first column includes the names of the features. The second and third column list the possible values they can have and their definitions, respectively. The last column lists the default value assigned during generation in case a feature is unspecified based on its type. There are two types of features: obligatory and optional. Obligatory features, such as verb subject or noun number, require a value to be specified. Therefore, in case of under-specification, all possible values are generated. Optional features, such as conjunction, preposition or pronominal object/possessive clitics, on the other hand can be absent. The pronominal features, subject, object and possessive, are defined in terms of sub-features specifying person, gender and number. In case any of these sub-features is under-specified, they are expanded to all their possible values. For example, the subject feature `S:2`, as in the case of the English pronoun 'you' (which is under-specified for gender and number), is expanded to `(S:2MS S:2FS S:2D S:2MP S:2FP)`. If no POS is specified, it is automatically determined by the lexeme and/or features. For example, the presence of a definite article implies the lexeme is a noun or an adjective; whereas a verbal particle or a subject/object implies the lexeme is a verb.[12]

The following is an example of an Arabic word and its lexeme-and-feature representation in ALMORGEANA.

(2) `[kitAb_1 POS:N PL Al+ l+]`
لِلكُتُبِ *lilkutubi*
'for the books'

The feature-set in this example consists of the nominal lexeme `kitAb_1` 'book' with the feature `PL` '*plural*', the definite article `Al+` 'the' and the prefix preposition `l+` 'to/for'.

14.4.3.2 Preprocessing Buckwalter Lexicons

ALMORGEANA uses the Buckwalter lexicon described in Section 14.4.2 *as is*. The lexicon is processed in ALMORGEANA to index entries based on inferred sets of

[12] Other POS not included in Table 14.2 are D *Determiner*, C *Conjunction*, NEG *Negative particle*, NUM *Number*, AB *Abbreviation*, IJ *Interjection*, and PX *Punctuation*.

Table 14.2. ALMORGEANA features

Feature	Value	Definition	Default
Part-of-Speech	POS:N	*Noun*	automatically
	POS:PN	*Proper Noun*	determined
	POS:V	*Verb*	
	POS:AJ	*Adjective*	
	POS:AV	*Adverb*	
	POS:PRO	*Pronoun*	
	POS:P *and others*	*Preposition*	
Conjunction	w+	*'and'*	none
	f+	*'and, so'*	
Preposition	b+	*'by, with'*	none
	k+	*'like'*	
	l+	*'for, to'*	
Verbal Particle	s+	*'will'*	none
	l+	*so as to*	
Definite Article	Al+	*the*	none
Verb Aspect	PV	*Perfective*	all
	IV	*Imperfective*	
	CV	*Imperative*	
Voice	PASS	*Passive*	all
Gender	FEM	*Feminine*	all
	MASC	*Masculine*	
Subject	S:PerGenNum	**Per***son = {1,2,3}*	all
Object	O:PerGenNum	**Gen***der = {M,F}*	none
Possessive	P:PerGenNum	**Num***ber = {S,D,P}*	none
Mood	MOOD:I	*Indicative*	all
	MOOD:S	*Subjunctive*	
	MOOD:J	*Jussive*	
Number	SG	*Singular*	all
	DU	*Dual*	
	PL	*Plural*	
Case	NOM	*Nominative*	all
	ACC	*Accusative*	
	GEN	*Genetive*	
Definiteness	INDEF	Indefinite	all
Possession	POSS	Possessed	all

features values (or *feature-keys*) that are used to map features in the input feature-sets to proper lexicon entries. This task is trivial for cases where the lexicon entry provides all necessary information. For example, verb voice and aspect are always part of the stem: the feature-key for *kutib*, the stem of the passive perfective form of the verb كَتَبَ/`katab` is `katab+PV+PASS`.

Many lexicon entries, however, lack feature specifications. One example is broken plurals, which appear under their lexeme cluster, but are not marked in any way for plurality (see the entry for كُتُب/kutub in Figure 14.1). Detecting when a stem is plural is necessary to include the feature *plural* in the feature-key for that stem. Using the English gloss to detect the presence of a broken plural is a possible solution. However, it fails for adjectival entries since English adjectives do not inflect for plurality, e.g. كَبِير/kabiyr (SG) and كِبَار/kibAr (PL) are both glossed as '*big*'. Additionally, some sound plural stems in the lexicon are glossed as plurals. The Buckwalter categories are not helpful on their own for this task. For example, the presence of a stem with morphological category N is ambiguous as to being a broken plural or a singular nominalization of a form I verb [11]. The solution for this problem stems from the observation that a singular verbal nominalization is its own *lexeme*, whereas a broken plural is always listed under a lexeme that is in a singular base form. A broken plural is by definition a major change in the form of the lexeme. Therefore, if a stem under a lexeme has the morphological category N, Ndip, or Nap (all of which can mark a broken plural) AND it is **not** a subset string of the lexeme, it is considered a broken plural. This technique works for entries considered part of the same lexeme in the Buckwalter lexicon. Entries that treat a broken plural as a separate lexeme will not be processed correctly, e.g. the lexeme إِخْوَة *Ǎixwaħ* 'brothers'.

14.4.3.3 Analysis and Generation

Analysis in ALMORGEANA is similar to Buckwalter's analyzer (Section 14.4.2). The difference lies in an extra step that uses feature-keys associated with stem, prefix and suffix to construct a feature-set for the lexeme-and-feature output. In the case of failed analysis, a back-off step is explored where prefix and suffix substrings are sought. If a compatible pair is found, the stem is used as a degenerate lexeme and the features are constructed from the feature-keys associated with the prefix and suffix.

The process of generating from feature-sets is also similar to Buckwalter analysis except that feature-keys are used instead of string sequences. First, the feature-set is expanded to include all forms of underspecified obligatory features, such as case, gender, number, etc. Next, all feature-keys in the ALMORGEANA lexicon that fully match any subset of the expanded feature-set are selected. All combinations of feature-keys that completely cover the features in the expanded feature-set are matched up in prefix-stem-suffix triples. Then, each feature-key is converted to its corresponding prefix, stem or suffix. The same compatibility tables used in Buckwalter analysis are used to accept or reject prefix-stem-suffix triples. Finally, all unique accepted triples are concatenated and output. In the case that no surface form is found, a back-off solution that attempts to regenerate after discarding one of the input features is explored. If the back-off fails, typically due to a missing lexical entry, a baseline Arabic morphological generator is used.

The baseline generator uses a simple concatenative word structure rule and a small lexicon. The lexicon contains 70 entries that map all features to most common surface realizations. For example, FEM maps to (ـَة /ap *aħ*, ـَت /at, and ϕ) and PL maps to (ات /At, ـِينَ /iyna, ـِي /iy, ـُونَ /uwna and ـُو /uw). Subtleties of feature interaction are generally ignored except for the case of subject and verb aspect since the circumfix realization of subjects in the imperfective/imperative form is rather complex to model concatenatively. The only word structure rule used in the baseline generator is the following:

```
<WORD> ::= (w|f) (s|l|b|k) Al <SubjectAspect>
<Lexeme>
<AspectSubject> <Gender> <Number> <Object> <Possessive>
```

14.4.4 Evaluation

ALMORGEANA uses the databases of the Buckwalter analyzer; therefore, its coverage is equivalent to the coverage of these lexicons. In this section, we evaluate ALMORGEANA engine for analysis and generation only.[13]

A sample text of over one million Arabic words from the UN Arabic-English corpus [31] was used in this evaluation. For each unique word in the text, ALMORGEANA is used in analysis mode to produce feature-sets. The resulting feature-sets are then input to two systems: the complete ALMORGEANA as described earlier *and* the baseline generator used as back-off to ALMORGEANA generation. For each feature-set, there are two sets of words: (a) words that analyze into the feature-set (A words) and (b) words that are generated from the feature-set (G words) (see Figure 14.2). The bigger the intersection between the two sets (C words), the better the performance of a system. Generated words that are not part of the intersection (C words) are Overgenerated words (O words). Words that analyze into the feature-set but are not generated are Undergenerated words (U words). In principle, U words are definite signs of problems in the generation system; whereas, O words can be correct but unseen in the analyzed text.

A system's Undergeneration Error (UnderErr) is defined as the ratio of U words to A words. Overgeneration Error (OverErr) is defined as the ratio of O words to G words. These two measure are equivalent to (1 - Recall) and (1 - Precision) respectively, if the set of A words paired with a feature-set is considered a gold standard to

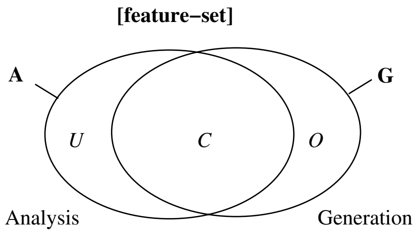


Fig. 14.2. ALMORGEANA evaluation

[13] The evaluation described here was run over the Buckwalter lexicons (version 1) [11].

be replicated in reverse by a generation system. The Combined Undergeneration and Overgeneration Error (CombErr) is calculated as (1 - *the corresponding F-score*):[14]

$$UnderErr = \frac{U}{A} = \frac{A-C}{A}, OverErr = \frac{O}{G} = \frac{G-C}{G},$$
$$CombErr = 1 - (\frac{2 \times (1 - UnderErr) \times (1 - OverErr)}{(1 - UnderErr) + (1 - OverErr)})$$

The evaluation text contained 63,066 undiacritized unique words, which were analyzed into 118,835 unique feature-sets corresponding to 14,883 unique lexemes. The number of unique diacritized words corresponding to the text words is 104,117. The evaluation was run in two modes controlling for the type of matching between A words and G words: diacritized (or diacritization-sensitive) and undiacritized. Evaluation results comparing ALMORGEANA to the baseline are presented in Table 14.3. The baseline system is almost six times faster than ALMORGEANA[15], but it had high undergeneration and overgeneration error rates. Both error rates were reduced in the undiacritized mode, where some erroneous output became ambiguous with correct output. ALMORGEANA, by comparison, reduced the combined error rate from the baseline by over 84%.

Many of the overgeneration errors are false alarms. They include cases of overgeneration of broken plurals, some of which are archaic or genre-specific but correct. For example, the word for ‘sheik’, شَيخ *šayx*, has three uncommon broken plurals in addition to the common شُيُوخ *šuyuwx*: أَشيَاخ *ÂšyaAx*, مَشَايِخ *mašaAyix*, and مَشَائِخ *mašaAŷix*. Another very common overgeneration error resulted from the underspecification of some mood-specific vocalic verbal suffixes in the Buckwalter lexicon. Arabic hollow verbs, for example, undergo a stem change in the jussive mood (from يَقُول *yaquwl* to يَقُل *yaqul*), which is indistinguishable in the analysis.

Table 14.3. Evaluation results

System	UnderErr	OverErr	CombErr	Time (secs)
ALMORGEANA *diacritized*	0.39%	12.22%	6.68%	1,769
ALMORGEANA *undiacritized*	0.38%	12.42%	6.79%	1,745
Baseline *diacritized*	43.90%	60.99%	53.98%	281
Baseline *undiacritized*	32.84%	47.93%	41.34%	293

[14] I would like to thank Christian Monson for suggesting this formula to computing CombErr. A previously published formula was biased toward underestimating the combined error [24].

[15] The experiments were run on a Dell Inspiron machine with Pentium 4 CPU, 512 MB RAM and 2.66 GHz.

Undergeneration errors stem exclusively from lexicon errors. These are not many and they can be expected in a manually created database. One example is caused by a missing lexeme comment in the Buckwalter lexicon which resulted in pairing all the forms of the verb رَأَى *raÂaý* 'to see' to the lexeme that appears just before it, رَاوَنْد *raAwand* 'rhubarb'. Such cases suggest a valuable use of ALMORGEANA as a debugging tool for the Buckwalter lexicon.[16]

14.5 Interoperability of Morphological Representations

This section describes how ALMORGEANA can be used to navigate between different levels of morphological representation. An Arabic word in simple tokenization can be analyzed using ALMORGEANA to multiple possible lexeme-and-feature analyses. This automatically gives us access to the lexeme-and-feature level and also the three-way segmentation used by Buckwalter's lexicons. To generate an intermediate representation such as the normalized segmentation used by the Penn Arabic Treebank [41], the features for conjunction, preposition and pronominal object/possessive can be stripped from the lexeme-and-feature analyses. The remaining features and lexeme are then used to generate the word stem using ALMORGEANA to guarantee a normalized form. The stripped features are also trivially generated and positioned relative to the word stem: [conjunction] [preposition] [word-stem] [pronoun]. Table 14.4 shows the different analyses for each word in the sentence. وقد كاتبته فتحية لمدة سنتين *wqd kAtbth ftHyħ lmdħ sntyn. 'and Fathia continued to correspond with him for two years'*. The correct Penn Arabic Treebank tokenization for this example is و قد كاتبت ه فتحية ل مدة سنتين. *w qd kAtbt h ftHyħ l mdħ sntyn.*

The ambiguity inherent in both the analysis and generation processes results in multiple possibilities (column 3 in Table 14.4). To select a specific segmentation, any of a set of possible techniques can be used such as rule-based heuristics or language models trained on text in the correct tokenization. For example, in the case of the Penn Arabic Treebank, the already tokenized text of the treebank can be used to build a language model for ranking/selecting among options produced by this technique (similar to [40]). Alternatively, machine learning over the features of the annotated words in the Penn Arabic Treebank can be used to select among the different analyses (similar to [26, 54, 30]).[17]

We developed a general tokenizer, TOKAN, as an implementation of this analyze-then-regenerate approach to tokenization. TOKAN is built on top of ALMORGEANA. TOKAN takes as input (a.) disambiguated ALMORGEANA analyses and (b.) a token

[16] All of the errors described here are for version 1 of the Buckwalter analyzer only [11]. We did not conduct a similar study on version 2 of the Buckwalter analyzer [12].

[17] The Morphological Analysis and Disambiguation for Arabic (MADA) tool [26] is a disambiguation system fully integrated with ALMORGEANA. More information on MADA is available at http://www.ccls.columbia.edu/cadim/resources.html.

Table 14.4. Normalized segmentation example

Word	Analysis	Segments
wqd	[qad~_1 POS:N w+ +SG +MASC gloss:size/physique]	wqd
	[qad_2 POS:F w+ gloss:may/might]	
	[qad_1 POS:F w+ gloss:has/have]	
	[qid~_1 POS:N w+ +SG +MASC gloss:thong/strap]	
	[waq~ad_1 POS:V +PV +S:3MS gloss:kindle/ignite]	wqd
	[waqod_1 POS:N +SG +MASC gloss:fuel/burning]	
	[waqadi_1 POS:V +PV +S:3MS gloss:ignite/burn]	
kAtbth	[kAtib_1 POS:N +FEM +SG +P:3MS gloss:author/writer/clerk]	kAtb$\hbar$ h
	[kAtib_2 POS:AJ +FEM +SG +P:3MS gloss:writing]	
	[kAtab_1 POS:V +PV +S:3FS +O:3MS gloss:correspond_with]	kAtbt h
	[kAtab_1 POS:V +PV +S:1S +O:3MS gloss:correspond_with]	
	[kAtab_1 POS:V +PV +S:2FS +O:3MS gloss:correspond_with]	
	[kAtab_1 POS:V +PV +S:2MS +O:3MS gloss:correspond_with]	
ftHy$\hbar$	[taHiy~ap_1 POS:N +FEM +SG f+ gloss:greeting/salute]	f tHy$\hbar$
	[fatHiy~ap_1 POS:PN gloss:Fathia]	ftHy$\hbar$
lmd$\hbar$	[mud~ap_1 POS:N +FEM +SG l+ gloss:interval/period]	l md$\hbar$
sntyn	[sinot_1 POS:N +MASC +DU +ACCGEN gloss:cent]	sntyn
	[sanap_1 POS:N +FEM +DU +ACCGEN gloss:year]	
.	[. POS:PX gloss:.]	.

definition sequence that specifies which features are to be extracted from the word and where they should be placed. For example, the token definition for splitting off the conjunction *w+* only is `"w+ REST"`. This token definition specifies that the conjunction *w+* is split from the word and whatever is left (REST) is regenerated after the conjunction *w+*. Similarly, the token definition for the Penn Arab Treebank tokenization is `"w+ f+ l+ k+ b+ REST +O: +P:"`.[18] ALMORGEANA and TOKAN have been used in both statistical and symbolic MT systems [29, 25].

14.6 Conclusions

In this chapter, we described obstacles facing MT researchers when working with Arabic resources in differing morphological representations. The lexeme-and-feature level of representation has been motivated and, ALMORGEANA, a large-scale system for analysis and generation from/to that level has been described and evaluated. We presented a framework using ALMORGEANA for navigating between Arabic

[18] More information on TOKAN is available at http://www.ccls.columbia.edu/cadim/resources.html.

morphological representations. This framework is useful for research exploring the effects of using different Arabic representations in MT.

Acknowledgments

This work has been supported, in part, by Army Research Lab Cooperative Agreement DAAD190320020, NSF CISE Research Infrastructure Award EIA0130422, Office of Naval Research MURI Contract FCPO.810548265, NSF Award #0329163 and Defense Advanced Research Projects Agency Contract No. HR0011-06-C-0023. I would like to thank Owen Rambow, Mona Diab, Bonnie Dorr, Tim Buckwalter, Michael Subotin and Christian Monson for helpful discussions.

References

[1] Azza Abdel-Monem, Khaled Shaalan, Ahmed Rafea, and Hoda Baraka. A Proposed Approach for Generating Arabic from Interlingua in a Multilingual Machine Translation System. In *Proceedings of the 4th Conference on Language Engineering*, pp. 197–206, 2003. Cairo, Egypt.

[2] Imad Al-Sughaiyer and Ibrahim Al-Kharashi. Arabic Morphological Analysis Techniques: A Comprehensive Survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213, 2004.

[3] Muhammed Aljlayl and Ophir Frieder. On Arabic Search: Improving the Retrieval Effectiveness via a Light Stemming Approach. In *Proceedings of ACM Eleventh Conference on Information and Knowledge Management, Mclean, VA*, pp. 340–347, 2002.

[4] Haytham Alsharaf, Sylviane Cardey, Peter Greenfield, and Yihui Shen. Problems and Solutions in Machine Translation Involving Arabic, Chinese and French. In *Proceedings of the International Conference on Information Technology*, pp. 293–297, Las Vegas, Nevada, 2004.

[5] Satanjeev Banerjee and Alon Lavie. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, Ann Arbor, Michigan, 2005. Association for Computational Linguistics.

[6] Kenneth Beesley. Arabic Finite-State Morphological Analysis and Generation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pp. 89–94, Copenhagen, Denmark, 1996.

[7] Daniel Bikel. Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine. In *Proceedings of International Conference on Human Language Technology Research (HLT)*, pp. 24–27, 2002.

[8] Jeff A. Bilmes and Katrin Kirchhoff. Factored Language Models and Generalized Parallel Backoff. In *Proceedings of the Human Language Technology Conference/North American Chapter of Association for Computational Linguistics (HLT/NAACL-03)*, pp. 4–6, Edmonton, Canada, 2003.

[9] Peter Brown, John Cocke, Stephen Della-Pietra, Vincent Della-Pietra, Fredrick Jelinek, John Lafferty, Robert Mercer, and Paul Roossin. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16:79–85, June 1990.

[10] Peter Brown, Stephen Della-Pietra, Vincent Della-Pietra, and Robert Mercer. The Mathematics of Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[11] Tim Buckwalter. Buckwalter Arabic Morphological Analyzer Version 1.0, 2002. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.

[12] Tim Buckwalter. Buckwalter Arabic Morphological Analyzer Version 2.0, 2004. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Cat alog No.: LDC2004L02, ISBN 1-58563-324-0.

[13] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the 11th conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, pp. 249–256, Trento, Italy, 2006.

[14] Michael Carl and Andy Way. *Recent Advances in Example-Based Machine Translation.* Kluwer Academic Publishers, Dordrecht, Holland, 1988.

[15] Violetta Cavalli-Sforza, Abdelhadi Soudi, and Teruko Mitamura. Arabic Morphology Generation Using a Concatenative Strategy. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP 2000)*, pp. 86–93, Seattle, Washington, USA, 2000.

[16] Michael Collins. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the ACL (jointly with the 8th Conference of the EACL)*, pp. 16–23, Madrid, Spain, 1997.

[17] Michael Collins, Philipp Koehn, and Ivona Kucerova. Clause Restructuring for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 531–540, Ann Arbor, Michigan, 2005.

[18] Kareem Darwish. Building a Shallow Morphological Analyzer in One Day. In *Proceedings of the workshop on Computational Approaches to Semitic Languages in the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pp. 47–54, Philadelphia, PA, USA, 2002.

[19] Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks. In *Proceedings of the 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, pp. 149–152, Boston, MA, 2004.

[20] Bonnie J. Dorr, Pamela W. Jordan, and John W. Benoit. A Survey of Current Research in Machine Translation. In M. Zelkowitz, editor, *Advances in Computers, Vol. 49*, pp. 1–68. Academic Press, London, 1999.

[21] Anas El Isbihani, Shahram Khadivi, Oliver Bender, and Hermann Ney. Morpho-syntactic arabic preprocessing for arabic to english statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pp. 15–22, New York City, June 2006. Association for Computational Linguistics.

[22] Sharon Goldwater and David McClosky. Improving Statistical MT Through Morphological Analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 676–683, Vancouver, Canada, 2005.

[23] Nizar Habash. *Generation Heavy Hybrid Machine Translation.* PhD thesis, University of Maryland College Park, 2003.

[24] Nizar Habash. Large Scale Lexeme Based Arabic Morphological Generation. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN-04)*, pp. 271–276, 2004. Fez, Morocco.

[25] Nizar Habash, Bonnie Dorr, and Christof Monz. Challenges in Building an Arabic-English GHMT System with SMT Components. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA06)*, pp. 56–65, Cambridge,MA, 2006.

[26] Nizar Habash and Owen Rambow. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 573–580, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[27] Nizar Habash and Owen Rambow. MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 681–688, Sydney, Australia, July 2006. Association for Computational Linguistics.

[28] Nizar Habash, Owen Rambow, and George Kiraz. Morphological Analysis and Generation for Arabic Dialects. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages at 43rd Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 17–24, Ann Arbor, Michigan, 2005.

[29] Nizar Habash and Fatiha Sadat. Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proceedings of the 7th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL06)*, pp. 49–52, New York, NY, 2006.

[30] Jan Hajič, Otakar Smrž, Tim Buckwalter, and Hubert Jin. Feature-based Tagger of Approximations of Functional Arabic Morphology. In Ma. Antonia Martí Montserrat Civit, Sandra Kübler, editor, *Proceedings of Treebanks and Linguistic Theories (TLT)*, pp. 53–64, Barcelona, Spain, 2005.

[31] Xu Jinxi. UN Parallel Text (Arabic-English), LDC Catalog No.: LDC2002E15, 2002. Linguistic Data Consortium, University of Pennsylvania.

[32] Lauri Karttunen, Ronald Kaplan, and Annie Zaenen. Two-level Morphology with Composition. In *Proceedings of Fourteenth International Conference on Computational Linguistics (COLING-92)*, pp. 141–148, Nantes, France, July 20–28 1992.

[33] George Kiraz. Multi-tape Two-level Morphology: A Case study in Semitic Non-Linear Morphology. In *Proceedings of Fifteenth International Conference on Computational Linguistics (COLING-94)*, pp. 180–186, Kyoto, Japan, 1994.

[34] Katrin Kirchhoff, Mei Yang, and Kevin Duh. Statistical Machine Translation of Parliamentary Proceedings Using Morpho-Syntactic Knowledge. In *TC-STAR Workshop on Speech-to-Speech Translation*, pp. 57–62, Barcelona, Spain, 2006.

[35] Kevin Knight. A Statistical MT Tutorial Workbook, April 30 1999. http://www.clsp.jhu.edu/ws99/projects/mt/mt-workbook.htm.

[36] Philipp Koehn. Pharaoh: a Beam Search Decoder for Phrase-based Statistical Machine Translation Models. In *Proceedings of the Association for Machine Translation in the Americas*, pp. 115–124, 2004.

[37] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical Phrase-based Translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, pp. 127–133, Edmonton, Canada, 2003.

[38] Kimmo Koskenniemi. Two-Level Model for Morphological Analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pp. 683–685, 1983.

[39] Young-Suk Lee. Morphological Analysis for Statistical Machine Translation. In *Proceedings of the 5th Meeting of the North American Chapter of the*

Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04), pp. 57–60, Boston, MA, 2004.

[40] Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. Language Model Based Arabic Word Segmentation. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL'03)*, pp. 399–406, Sapporo, Japan, 2003.

[41] Mohamed Maamouri, Ann Bies, and Tim Buckwalter. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt, 2004.

[42] Guido Minnen, John Carroll, and Darren Pearce. Robust, Applied Morphological Generation. In *Proceedings of the 1st International Conference on Natural Language Generation (INLG 2000)*, pp. 201–208, Mitzpe Ramon, Israel, 2000.

[43] Sonja Nieıen and Hermann Ney. Statistical Machine Translation with Scarce Resources Using Morpho-syntactic Information. *Computational Linguistics*, 30(2), 2004.

[44] Franz Josef Och. Google System Description for the 2005 NIST MT Evaluation. In *MT Eval Workshop (unpublished talk)*, 2005.

[45] Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. A Smorgasbord of Features for Statistical Machine Translation. In *Proceedings of the Human Language Technology / North American Association of Computational Linguistics Conference*, pp. 161–168, Boston, Massachusetts, 2004.

[46] Franz Josef Och and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–52, 2003.

[47] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, PA, 2002.

[48] Aaron Phillips and Violetta Cavalli-Sforza. Arabic-to-English Example Based Machine Translation Using Context-Insensitive Morphological Analysis. In *Journées dŠEtudes sur le Traitement Automatique de la Langue Arabe (JETALA)*, Rabat, Morocco, 2006.

[49] Maja Popović and Hermann Ney. Towards the Use of Word Stems and Suffixes for Statistical Machine Translation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pp. 1585–1588, Lisbon, Portugal, May 2004.

[50] Chris Quirk, Arul Menezes, and Colin Cherry. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 271–279, Ann Arbor, Michigan, 2005.

[51] Jason Riesa and David Yarowsky. Minimally Supervised Morphological Segmentation with Applications to Machine Translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA06)*, pp. 185–192, Cambridge, MA, 2006.

[52] Fatiha Sadat and Nizar Habash. Combination of Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 1–8, Sydney, Australia, July 2006. Association for Computational Linguistics.

[53] Mohammed Sharaf. Implications of the Agreement Features in (English to Arabic) Machine Translation. Master's thesis, Al-Azhar University, 2002.

[54] Noah Smith, David Smith, and Roy Tromble. Context-Based Morphological Disambiguation with Random Fields. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP05)*, pp. 475–482, Vancouver, Canada, 2005.

[55] Harold Somers. Review Article: Example-based Machine Translation. *Machine Translation*, 14(2):113–157, 1999.

[56] Abdelhadi Soudi. Challenges in the Generation of Arabic from Interlingua. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN-04)*, pp. 343–350, 2004. Fez, Morocco.

[57] Abdelhadi Soudi, Violetta Cavalli-Sforza, and Abderrahim Jamari. A Computational Lexeme-Based Treatment of Arabic Morphology. In *Proceedings of the Arabic Natural Language Processing Workshop, Conference of the Association for Computational Linguistics (ACL 2001)*, pp. 50–57, Toulouse, France, 2001.

[58] Abdelhadi Soudi, Violetta Cavalli-Sforza, and Abderrahim Jamari. A Prototype English-to-Arabic Interlingua-based MT system. In *Proceedings of the Third International Conference on Language Resources and Evaluation: Workshop on Arabic language resources and evaluation*, Las Palmas, Spain, 2002.

[59] Andreas Zollmann, Ashish Venugopal, and Stephan Vogel. Bridging the inflection morphology gap for arabic statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pp. 201–204, New York City, USA, 2006. Association for Computational Linguistics.

15

Arabic Morphological Generation and its Impact on the Quality of Machine Translation to Arabic

Ahmed Guessoum[1] and Rached Zantout[2]

[1] *Department of Computer Science, The University of Sharjah, P.O. Box 27272, Sharjah, UAE guessoum@sharjah.ac.ae*

[2] *Faculty of Engineering, Department of Computer & Communications Eng., Faculty of Engineering, Hariri Canadian University, Mechref, P.O.Box:10 – Damour, Chouf 2010, Lebanon, rached@cyberia.net.lb*

Abstract: The aim of this chapter is to highlight the complexity and importance of Arabic morphological information in an Arabic Machine Translation (AMT) system, i.e. a system that translates to or from Arabic. We summarize Arabic morphology and introduce the main morphological information that we have found relevant to machine translation to Arabic and categorize it into various types of features. In order to show the impact of these morphological features on machine translation quality, we have adopted an approach whereby we relate each of them to the quality of the translation. This leads us, through a statistical analysis of the test data, to a characterization of which features are more important in terms of their impact on the quality of the translation of a given AMT system (AMTS). The approach has been implemented and applied to evaluating an English-to-Arabic web-based MT system. The results of the evaluation of this system are presented, conclusions are drawn, and recommendations for improving their outputs made

15.1 Introduction

Translating between different languages is a very important discipline. The estimated value of the world market for translation was U.S. $20 billion according to the Gartner Group (Stamford, CT, USA) with an annual growth rate of 14.6% (Van der Meer, 2003). In 2004, the human translation market was estimated to be $1 billion (Oren, 2004), while the machine translation market was forecast to be in the $100 million range. MT software was reported to be responsible for the completion of between 30 and 50 percent of a Machine translation task automatically (ECL, 1996) and (Hedberg, 1995). MT software was also estimated to cut the cost of translation by two thirds.

Evaluation of Natural Language Processing (NLP) systems is currently a field of research on its own. Various researchers have stressed the importance of component-based evaluation and detailed error analyses (Arnold et al., 1993; Hedberg, 1995; Nyberg et al., 1994). Since MT systems combine lexical analyzers, morphological analyzers, parsers, semantic disambiguation modules, generators, and pragmatic analysis modules, it is important to be able to evaluate these various components individually as well as to evaluate the overall system. The main difficulty here is that, in most of the cases, evaluators do not have access to the individual components of the system under evaluation and are therefore forced into black-box evaluation. This means that an error in the output of the system cannot be attributed to one of the components since it can be due to one or many errors in one or more of the components of an NLP system.

In (Van Slype, 1979) evaluation is subdivided into two main categories: macro-evaluation and micro-evaluation. Among the macro-evaluation assessment components that are affected by morphological errors are the cognitive level components such as intelligibility, fidelity, coherence, usability, and acceptability of a translation. Among the micro-evaluation methods affected by morphological errors are the grammatical symptomatic components such as the analysis of grammatical errors found in the target output. (Chaumier et al., 1977) suggest an even finer scrutiny of the grammatical (sub-) constructs in the source and target texts, e.g., noun phrases, adjectival and verb phrases, object complements, adverbial complements, etc.

In general, there seems to be an agreement as to the following aspects that should be evaluated in any MT system: *adequacy*, which is the extent to which the meaning of the source text is rendered in the translated text; *fluency*, which is the extent to which the target text appeals to a native speaker in terms of well-formedness of the target text, grammatical correctness, absence of misspellings, adherence to common language usage of terms, and meaningfulness within the context (White et al., 1994); *informativeness*, which assesses the extent to which the translated text conveys enough information from the source text as to enable evaluators to answer various questions about the latter based on the translated text; and *intelligibility* (Arnold et al., 1993), which is strongly related to informativeness, though directly affected by grammatical errors and mistranslated or missing words. In (Nyberg et al., 1994) the authors from the KANT (Nyberg et al., 1992) team introduce a methodology based on evaluation metrics for knowledge-based MT. The evaluation criteria they consider are: *completeness,* which measures the ability of a system to produce an output for every input; *correctness,* which measures the ability of a system to produce a correct output for every input; and *stylistics,* which measures the appropriateness of the lexical, grammatical, and other choices made during the translation process. Based on the completeness, correctness, and stylistics criteria, the authors then defined four evaluation criteria, which test, as percentages, the Analysis Coverage, Analysis Correctness, Generation Coverage, and Generation Correctness. These four percentages then get multiplied, yielding the *Translation Correctness*, which measures the overall quality of the system. (Guessoum & Zantout, 2001) and (Guessoum & Zantout, 2005) introduce a semi-automatic methodology for component evaluation of Arabic MT systems (AMTSs) using a black-box approach. The methodology tests the correctness of each component of an MT system by analyzing carefully selected sentences translated by an MT system. Weighted averages are then computed and scores are derived for each component of a system under evaluation. An overall score for the system is also calculated. The weighted averages were shown to be indicators of what components of the system are the faultiest and therefore would need immediate attention by the developers. The difficulty in this approach is the ability to come up with a large number of test sentences that would test each component of the MT system.

The aim of this chapter is to highlight the complexity and importance of Arabic morphological information in an Arabic Machine Translation (AMT) system. In Section 15.2 we summarize Arabic morphology and in Section 15.3 we introduce the main morphological information that we have found relevant to machine translation to Arabic. We show how various aspects of Arabic morphology reflect important lexical, syntactic, semantic, and pragmatic aspects of a sentence in translation. We also categorize this information into various types of features. In Section 15.4 we show how, through a statistical analysis of a bilingual corpus consisting of English source text and machine-produced Arabic target text, we could suggest a characterization of which morphological features are more important in terms of their impact on the quality of the translation of a given AMT system (AMTS). In Section 15.5 conclusions are drawn, and recommendations for improving the outputs of AMT systems made.

15.2 Basics of Arabic Morphology

The information in this section was derived through readings in (Dahdah, 1995), (Hamalawy, 1996), and (Rajhi, 1979). Arabic words are grouped into three main categories: Nouns (الأسماء), Verbs (الأفعال) and Prepositions (الأحرف). The Noun and Verb categories consist of subcategories that affect how the word is used in the sentence and how it changes within the context of the sentence and the other words in the same sentence. Each subcategory obeys certain rules of morphology that detail whether a word can be used in a certain context and how its form changes in that context. The difference between words in subcategories can be as subtle as the presence (or absence) of a vowel or as clear as the addition (or removal) of letters to the word when it moves from one subcategory to another. Although the Preposition category contains subcategories, prepositions in Arabic do not, in general, change forms.

Morphology for Arabic is a tool that enables the language to grow and develop. Morphology, in general, is defined as producing a word from another by changing it so that it fits a certain new meaning. In Arabic, morphology is divided into four categories of derivations, the small (الإشتقاق الأصغر), the large (الإشتقاق الكبير), the larger (الإشتقاق الأكبر) and the largest (الإشتقاق الكبار). The small morphology produces one word from another but keeps similarities between the two words in their pronunciation and meaning (e.g. [1] علم (ςilm, science)→ عالم (ςAlim, scientist)). The large morphology produces a word from another by exchanging the letters in the roots of the words (علم (ςilm, science)→ عمل (ςamal, work)). The larger morphology produces a word from another by changing one (or more letters) and keeping the same meaning (عنوان (ςnwAn, address)→ علوان (ςulwAn, address)). The largest morphology produces a word from a group of words such as the contraction بسملة (basmalħ) from بسم الله الرحمن الرحيم (bismi All~Ah Alr~aHmAn Ar~aHym), "In the name of Allah, The Compassionate, The Merciful"). By far, the mostly used type of morphology in Arabic is the small morphology.

Small morphology can act on a Noun or a Verb. Any Noun or Verb in Arabic consists of a root and added letters. The roots for Arabic words have traditionally been considered to consist of three letters (the mostly used type of roots in Arabic) or four letters. Like English and French, in Arabic, letters can be added to the beginning and/or to the end of the root. However, unlike English and French, in Arabic, letters can be added inside (between the letters of) a root. This is one of the complexities that make Arabic a harder language to analyze or generate morphologically.

Fortunately, for computational linguists interested in developing Arabic language tools on computers, Arabic is a structured language. Basically, verbs and nouns cannot accept additions of all letters in the Arabic language at all places inside, at the beginning or the end of a root. In this chapter we will explain some of the rules pertaining to verbs in the Arabic language. The reader should bear in mind that Nouns obey similar rules. The reader should also bear in mind that the rules described below will not be exhaustive even for Arabic verbs as the purpose behind the explanation is to give the reader an appreciation of the complexity of Arabic morphology rather than to enumerate all the rules governing Arabic morphology.

In Arabic, there are certain letters that can come at the beginning of the root (prefixes); these letters are grouped in the Arabic word سألتني (s, A, l, t, n, y). The letters that can be added to the end of the root (suffixes) are grouped in the Arabic word أوهمتني (A, w, h, m, t, n, y). There is an upper limit on the number of prefixes (four letters) that can be added to a root. In some cases, a letter can be repeated as a prefix or a suffix. Letters that can be added to the inside of a root (infixes) have a more complicated set of rules. First, a group of letters ا (A), و (w), and ي (y)

[1] In the rest of the paper, any Arabic sentence will be followed by its transliteration using the scheme followed throughout the book .

(called أحرف العلة), while being part of the root, can disappear from an Arabic verb if the verb is in the imperative form. Second, only one or two letters can be added inside the root. Third, infixes are grouped in the Arabic word اتوني (A, t, w, n, y). Fourth, certain verb forms can be produced by repeating the same letter of the root.

Arabic morphology is a very structured process. For example, a verb can undergo morphology based on moulds that take any root and transform it into the corresponding verb by adding prefixes, suffixes or infixes in order to convey the meaning of the verb. For example, in order to specify that more than two people are writing to each other, the root كتب (kataba, (he) wrote) is used. The suffix ون (wn) is added to it to indicate that the verb is being done by more than two people[2]; the prefix ي (y) is added to indicate that the verb is in the present tense; and the infix ا (A) is added to indicate that they are writing to each other. Thus the verb obtained is يكاتبون (yukAtibwn). If the same meaning is to be conveyed but, now, instead of the group of people writing to each other we want to say that they play with each other, it is only necessary to replace the letters of the verb root write كتب (kataba) with those of the verb root play لعب (laçiba) to obtain the verb يلاعبون (yulAçibwn). In a similar manner, if the group consisted of two instead of more than two members then the only change needed is to use the suffix ان (An) instead of the suffix ون (wn). The different forms that can be used with a root to produce an Arabic verb have been classified differently in the literature. One such classification (Fowzan et al., 2000) enumerates 129 Morphological Patterns which can be used to generate verbs from roots.

In Arabic, the Noun or Verb will have different forms if the subject or object is masculine or feminine. Also differences in the forms can exist if the sentence refers to one person or a group of two or a group of more than two.

15.3 Arabic Morphological Generation as a Repository of Morphological, Syntactic, Semantic, and Pragmatic Information (in an AMTS)

It is well-known that Machine Translation is a complex process. It is ideally the result of analyzing the source text morphologically, lexically, syntactically, semantically — and even pragmatically and stylistically if needed, and producing its equivalent target text using all of these linguistic dimensions. The target language and the complexity of its morphology and grammar can make this machine translation process even more complex. This is indeed the case for Arabic, where, due to the complexity of the morphology, the generation of correct Arabic words must

[2] Recall that Arabic has two forms for the plural: the dual and the non-dual plural (more than two people).

take into account and reproduce all the linguistic information acquired from the analysis of the source text, be the source language English or any other.[3]

Consider for instance a simple sentence like "The *girls wrote the beautiful essays*". To translate it to Arabic, the morphological generator needs to

1. add the prefix ال (Al, the) to the word بنات (banAt, girls) which itself is the result of adding the infix ا (A) to obtain the plural of بنت (bint, girl);
2. add the suffix ت (t) to the basic verb form كتب (kataba, he wrote) to produce كتبت (katabat, she wrote) for past tense, feminine form;
3. add the prefix ال (Al, the) and suffix ات (At, feminine plural) to the word مقال (maqAl, an article/essay) because, in Arabic, the masculine word for article/essay has a feminine plural form;
4. add the prefix ال (Al, the) and suffix "ة" (ħ, for feminine) to the adjective جميل (jamyl, beautiful (masculine singular)) to obtain the feminine form of the adjective, for gender concordance with the plural word for articles/essays.

The generated Arabic sentence would therefore be

كتبت البنات المقالات الجميلة.
katabat AlbnAtu AlmqAlAt Aljmylħ
Wrote (fem.) the-girls (fem. plural) the-articles (fem. plural) the-beautiful (fem.).

In the above example, the verb and the subject have to match in gender; the noun and the adjective match in gender but also with respect to the definite case. It is clear that the morphological generator needs to take into account lexical and syntactic information in addition to the fact that a word re-ordering needs to be introduced by the "transfer" module.[4] As such, by reading the Arabic sentence, we can immediately tell (i.e. from the output of the morphological generator) whether the words are lexically and syntactically correct. In fact, even the meaning can be affected, as will be explained in the coming sections, if the morphological generator does not receive this information from other modules (such as the parser) in the machine translation system or does not correctly reproduce it.

In more complex examples, pragmatic knowledge could be used, such as reference resolution, so that the proper form of a word is generated. Consider, for example, the following sentence, its translation, and gloss:

This is your room - it's rather small هذه هي حجرتك– هي صغيرة إلى حدّ ما
haðihi hiya Hujratuka – hiya Saγyraħũ Ăilý Had~ĩ mA
This it (is[5]) your-room - it (feminine) (is) small (feminine) to limit some

[3] Despite the fact that the discussion in this chapter is about Arabic morphological generation, independently of the source language in any machine translation process, all the examples and the implementation will be about machine translation from English to Arabic.

[4] We call it "transfer" module no matter what actual approach is adopted in the machine translation system.

[5] The auxiliary "*is*" is implicit in Arabic.

In this case, the demonstrative pronoun "*this*" should be translated as هذه (haðihi, (feminine) *this*) and not هذا (haðA, (masculine) *this*) since حجرة (Hujraħ, room) is feminine. The gender needs also to be conveyed in the second هي (hiya, (feminine) *it*) and صغيرة (Saγyraħ, (feminine) *small*). Again, an error in the resolution of the references would be confusing or misleading. For instance, if the previous source sentence gets incorrectly translated as

(*)هذه هي حجرتك – هو صغير إلى حدّ ما

haðihi hiya Hujratuka – huwa Saγyrũ Ăilý Had~ĩ mA

This it (is) your-room - it (masc.) (is) small (masc.) to limit some

This translation would convey a completely different meaning. Indeed, the reader would believe that the speaker mentions the room of the listener but goes on/back to talking about some other person describing *him* as small to some extent. If a male person happens to be mentioned in the context of the sentence, the confusion would become complete!

In the context of a black-box evaluation, it is not possible to find out the faulty component(s) of a machine translation system that produces an output like the one in the last example. We cannot be sure whether the morphological generator received all the needed information to produce the correct words, and hence it would be the faulty component, or if it did not receive enough information from the other components in the MT system (parser, transfer, etc.) as to generate the correct words. In all cases, what we argue is that the presence or absence of morphological information can affect quite seriously the quality of an MT system.

From an analysis of a large number of sentences, as will be explained in the next section, we have singled out and categorized various types of morphological features that are important in Machine Translation and which can affect its quality. In fact, this is exactly the criterion for singling them out: we selected a feature if its improper handling affects the sentence quality.

These features are now presented, with clarifying examples, and will be further analyzed in the coming sections on an actual Arabic MT system.

1. Definite / Indefinite Nouns (A)

As explained earlier, an indefinite Arabic noun can be made definite by adding the article ال to it. From our analysis of the outputs of various AMT systems, this is quite frequently not done correctly. The result can be objectionable or even unclear.

Monkeys are very agile climbers.	القرود متسلّقون رشيقون جدّاً.
… afflicts women more than men.	... يبتلي النّساء أكثر من رجال.(*)

In the first example the prefix definite article ال (Al) is correctly added to the noun قرود (quruwd, monkeys). In the second example, both words for men and women should be definite in Arabic. The translation correctly places the definite article for "(the) women" (النساء , Aln~isA') but not for "men" (رجال , rijAl).

2. Case Ending (B)
One of the complexities of Arabic grammar is that words get inflected depending on the case (nominative, accusative, or genitive), the number (singular, dual, or plural), and the gender (masculine or feminine). The generation of the correct form of the word depending on these features must obviously be done very carefully and is in fact a common mistake among native speakers nowadays! The improper handling of the case ending results in unpleasant sentences and sometimes it modifies the sentence meaning entirely, especially when the word order is changed.[6]

… the massive aerial bombardment of military targets continued unabated.	القصف الجوّيّ الضّخم للأهداف العسكريّة استمرّ <u>قويّا</u>.
He abided in the wilderness for forty days.	أقام في الصّحراء لمدّة **<u>أربعون</u>** يوم. (*)

In the first sentence, the Arabic adjective قوّيا (qawiy~Aã , massive/strong) correctly appears in the accusative form. However, the numeral أربعون (ÂrbaÇwn, forty) should be in the genitive form while it appears in the nominative.

3. Imperative Mood (C)
A common error found in Arabic MT systems is the incorrect translation of verbs in the imperative mood into verbs in the present tense of the indicative mode, often with the wrong pronoun (he) being used. This is the case with the system we have evaluated for the purposes of this work.

Please contact…	(*) من فضلك **<u>يتصل</u>**

Here the verb contact is incorrectly translated as يّتصل (yat~aSil, he contacts). It should rather be the imperative form of the verb (i.e., Ăt~aSil, contact).

4. Verb Tenses (D)
This is another error commonly found in AMT systems. It is often coupled with the incorrect use of the pronoun (he). Obviously, enough morphological information needs to be made available to the morphological generator not to fall into this trap. Sometimes, this mistake shows up when the present and past tenses have the same form for a given verb in the source language (English in our case). However, the error would probably reflect an improper syntactic parsing of the source sentence.

[6] Word order modification is a fairly common thing to do in Arabic; it is usually used to give a different emphasis in the sentence.

Why don't we put the bed …	لماذا لا نضع السّرير ...
The anti-war agitation has begun ...	(*) الثورة المعارضة للحرب تبدأ...

In the first example, "we put" is correctly translated as نضع (naDaçu, we put) in the present tense whereas, in the second example, "has begun" somehow gets translated to تبدأ (tabdaÂu, (it) begins).

5. Expressions (E)

Handling common expressions often requires the application of prepositions, the definite article, pronouns, etc., to various categories of words. If this is not carefully done, the result will be incorrect words (in the context) or even morphologically ill-constructed words as explained in item 8 below. Of course, most of the time a word-to-word translation of these expressions gives appalling results morphologically, syntactically, and semantically.

For a man of 80…	بالنسبة لرجل في الـ80
… in the 25 to 40 age group.	**(*) في 25 إلى 40 جيل ...**

The first sentence is correctly translated with the proper Arabic preposition في (fy, in). The second sentence is badly translated, which results in the meaning "in 25 to 40 generations"!

6. Pronouns (F)

Another problem is the proper handling of pronouns. Pronouns can appear either suffixed to a word or separated from it, based on various morphological and syntactic rules. The pronouns may take one of several forms depending on the features mentioned earlier, namely the case, gender, and number.

6.1. Pronoun-Related Concordance (Case Ending)

The morphology of a word can get affected by a pronoun in a sentence. For instance, in the first example below, "as an afterthought" is translated as مستدركة (mustadrikaħã, as an afterthought of hers) the last letter of this word reflecting the fact that the subject is feminine. If the subject was a plural one like "They", the word مستدركة would become مستدركات (mustadrikAt, for the feminine plural) or مستدركين (mustadrikyn, for the non-feminine plural). Similar changes would occur if the subject was dual (two people), etc. In the second example, طافيا (TAfiyAã, afloat) appears incorrectly in the masculine singular form; it should be طافية (TAfiyaħã, afloat (feminine singular)).

All such morphological information needs to be available and generated for the sentence to be correct.

She only asked me … as an afterthought.	دعتني ... مستدركة .
She spent seven days afloat on a raft.	(*) قضت سبعة أيّامًا **طافيًا** على قارب.

6.2. Gender Concordance (and Pronoun Resolution)

Also a common source of error is gender concordance, where a pronoun needs to reflect the gender, or other more general aspects of pronoun resolution such as number. In the example below, حجرتك (Hujratuk, your room) is feminine but is incorrectly translated using the pronoun هو (huwa, it (masculine)) and صغير (Saγyr, small (masculine)).

This is your room - it's rather small...	(*) هذه هي حجرتك - **<u>هو صغير</u>** إلى حدّ ما...

6.3. Unnecessary Generation of Pronouns

From our experience with AMT systems, this error is quite a common one. A pronoun is very frequently generated when it is not needed. Indeed, in Arabic a verb implicitly indicates the person (singular, dual or plural; 1st, 2nd, or 3rd; etc.). As such the addition of a pronoun before the verb is often a redundancy (unless there is an emphasis to be conveyed) and sounds heavy. In the example below, سيُجْمَعُونَ (sayujmaςwun, (they) will be reunited) has the suffix ون which indicates that the subject is they (non-feminine plural taken by default here). Adding the pronoun to such a verb would convey a meaning like "It is they who will be reunited in the afterlife", which in the case of this particular sentence and its religious connotation, is probably quite unacceptable.

They'll be reunited in the afterlife.	<u>(هم)</u> سيُجْمَعُونَ في الآخرة .

6.4. Incorrect Pronoun

Sometimes, the pronoun is not obvious to guess from the sentence; it is understood from the context or by commonsense. In the example below, "It's advisable to book seats…" would probably mean "It's advisable for you to book seats…" or "It's advisable for one to book his/her seats…". A common mistake in AMT systems is to simply translate the verb "to book" as يحجز (yaHjizu, he books) conveying the following meaning for the sentence "It's advisable for him to book his seats…". This level of morphological information detail is indeed needed to render the semantics of the sentence. A human translator would most probably translate the sentence into:

من المستحسن حجز المقاعد على الأقل أسبوع مقدّماً

It's advisable to book seats at least a week in advance.	(*) هو مستحسن أن <u>يحجز</u> المقاعد على الأقلّ أسبوع مقدّمًا.

7. Number Concordance (G)

This feature should be clear from the above explanations. Nouns, verbs, adjectives, and pronouns match with respect to number. In the examples below, لحضارتين (liHaDAratayni, of two civilizations, genitive form) is in the dual form which is different from لحضارة (liHaDArħ, of one civilization). As such, the adjective must be in the dual genitive form عظيمتين (ςĎymatayni, (two) great). In the second example, متلهّف (mutalah~if, agog) incorrectly appears in the (default) masculine singular form, which would reflect that the subject is "he" (instead of we).

… of two great civilizations.	... لحضارتين عظيمتين.
We waited agog for news.	(*) انتظرنا **متلهّف** للأخبار .

8. Constructed Words (H)

In a number of cases that we have come across, ill-constructed words were generated. This clearly reflects errors that are intrinsic to the morphological generator. For instance, the output in the first example below, an invalid word كبعيدًا (kabaçydAã, as far afield) is generated. This word has probably been constructed by adding the prefix ك (k, as) to the word بعيدًا (baçydAã, far) giving a form which is not correct in Arabic. A similar process was followed in the second example where the suffix ني (ny, me) instead of ي (y, me/my) was incorrectly combined with the word إغضاب (ĂγDAb, aggravating).

… as far afield as Japan ...	... كبعيدًا كاليابان ...
Stop aggravating me…	توقّف عن إغضابني...

9. Inadequate Prepositions (I)

A preposition can prefix a word (e.g. prepositions ب , ك , and ل) or can be separated from it (e.g. prepositions من , عن, and على). If one is not careful, prepositions may be incorrectly translated. For example, with the AMT system we have evaluated, "at" in the first example below, was translated as في (fy, in) instead of ب (bi, with/at). Likewise, the incorrect preposition ل (li, for) was selected instead of عن (çn, of) to prefix the word حقوق (Huquwq, rights).

... at affordable prices.	(*) ... **في** الأسعار المتاحة.
She is renowned for her advocacy of human rights.	(*) هي مشهورة بدفاعها **لحقوق** الإنسان .

10. Use of the Improper Grammatical Category (J)

This type of error, as we will see below, does occur fairly commonly with AMT systems. It confirms what we have concluded in previous work (Guessoum & Zantout, 2001) and (Guessoum & Zantout, 2005) that the AMT systems we have evaluated follow an improved form of direct MT, although some of them claim that they use transfer-based MT. In the first example below, the AMT system we have evaluated, translates "forty-three" textually as ثلاثة وأربعون (θalAθaħũ wa Ârbaçwn) instead of producing the correct form في الثالثة و الأربعين (fy AlθAliθaħi wa AlÂrbaçyn, in the forty third (year)).

… and at forty-three, somehow ageless.	(*) ... و في ثلاثة و أربعون من عمرها، بطريقة ما دائمة الشّباب.
She asked the question expecting an affirmative.	(*) سألت السّؤال تنتظر موافقة .

The above features are what we have singled out as features to be looked at when evaluating the Arabic morphological generation module of an AMT system. The approach adopted in our work is now presented.

15.4 Analysis of Arabic Morphological Generation Features in an Arabic MT System

In order to study the impact of Arabic morphological generation features on the quality of MT to Arabic in an AMTS, we have collected in Phase 1 English sentences by looking up 1056 English words online using the Cambridge Dictionaries site at http://dictionary.cambridge.org. Out of the 1056 words, 781 were found to have sample sentences in the online dictionary. Out of these, 756 could be translated to Arabic using the web-based AMTS *Ajeeb* (http://www.ajeeb.com). These Arabic sentences have then gone through Phase 2. In Phase 2, we analyzed all the pairs of English and Arabic sentences looking for the various types of morphological features that we could single out as important in AMT. These are the various types of features that were presented in Section 15.3. Having a classification of the morphological information that is relevant to AMT, we needed to see how frequent the types are and how much they affected the quality of the output of a given AMTS.

One aspect we have mentioned earlier is that the errors found in the translation may be due to the morphological generator, syntactic parser, or any other module of the AMT system. However, what is relevant to our work is that whatever the source of the error, it is reflected at the morphological generation level as explained in the previous section. As we wanted to find a correlation between the type of the error and the quality of the translation, we had to be very careful in our evaluation approach. In fact, we have followed a number of steps.

First, we kept only the sentences or sentence chunks which contained at least one error related to the morphological features mentioned in Section 15.3. Errors like wrong word order have not been considered as they are purely syntactic (and therefore most probably not related to the morphological generator).

In the second step, we discarded all the sentences that contained more than two errors of the morphological types of interest. This is to avoid confusion as to which type affects affect the quality of the translation most, and for how much. We ended up with 177 sentences that contained one or two errors related to the categorized morphological features. We then tagged each sentence with A, B, or …, J, where A stands for the feature type "Definite/Indefinite", B for "Case Ending", etc., up to J for "Use of Improper Grammatical Category", as defined in Section 15.3. As some of the sentences may have two errors, not just one, we decided to assign only one tag/letter considering that error which most affects the sentence and, for simplicity, (mentally) correcting the other one.[7] As a result, each sentence received exactly one tag.

At this point we were ready for evaluating the selected pairs of sentences for adequacy and correctness of the machine translation. This has been done by human experts who were asked to assign a value between 0 and 5, where 0 means completely unacceptable and 5 perfectly clear at first reading while being faithful to the source sentence and sounding correct.

Once a translation quality value was assigned to each tagged sentence, we computed statistics giving the number of sentences afflicted with each type of error (having been assigned a specific tag) as well as the average quality measure (value between 0 and 5, inclusive) for each one of these types. This measure tells us how much an error for that morphological feature affects the quality of the translation. The closer the value to 0 for a particular type of morphological information, the more serious an impact the type has on the quality of the translation. Obviously, the closer the value to 5, the less impact the type has on the quality of the translation.

15.5 Results and Data Analysis

Table 15.1 shows the results of the analysis of the 177 pairs of (tagged) sentences translated using the AMT system mentioned in Section 15.4.

The frequencies of the sentences of types A, B, …to J, are given in the second row. We clearly see that the largest number of errors is about the handling of pronouns (20.34% of the cases), followed by using improper grammatical category (16.95% of the cases), and then by using inadequate prepositions (15.82% of the

Table 15.1. Quantitative breakdown of morphological errors affecting AMT system quality

Type of Error	A	B	C	D	E	F	G	H	I	J
Number of Cases	20	22	7	9	8	36	8	9	28	30
% of cases	11.3	12.43	3.9	5	4.5	20.34	4.5	5	15.82	16.95
Average (Out of 5)	4.25	4.41	1	2.56	2.5	2.9	3.12	2.22	2.61	1.43

[7] Note that we could refine our evaluation by considering combinations of errors as further affecting the meaning/intelligibility of the target sentence.

cases). On the other hand, the least frequent error type is the handling of the imperative mood (3.9% of the cases).

The frequencies just presented are not meaningful enough on their own if we do not know how serious each of the error types is and how much it affects the quality of the translation. This is what the last row of Table 15.1 tells us. In particular, it shows that the type of morphological error which most affects the quality of the translation (in the case of the AMTS under evaluation) is the handling of the imperative mood. The results tell us that despite the fact that this error occurs in only 3.9% of the cases, whenever it occurs, it drastically affects the meaning of the sentence with an average score of 1 out of 5. The rest of the table can be read in the same way. In particular, only two types of errors are "mild" enough as to produce sentences that are still reasonably adequately translated, with a quality score of more than 4 out of 5. These types of errors are the incorrect handling of Definite/Indefinite nouns and the incorrect case endings. Both of these errors are fairly frequent (>11% of the cases for each one) but do not seriously affect the meaning and correctness of the target sentence.

Table 15.1 is concise enough and, in our opinion, quite useful for a better evaluation of an AMT system and, hopefully, for its improvement. Indeed, the averages we have computed also tell us how much an improvement of the quality of the sentence we would get if we correct that particular type of morphological information error. For instance, correcting the error for the case of the handling of the imperative mood should make the sentence noticeably more comprehensible.

15.6 Conclusion and Recommendations

In this chapter, the importance of Morphological Generation to the clarity of the output of an MT system was emphasized. The complexity of morphology in Arabic was presented through the description of some of the rules that govern Arabic morphology. An analysis of the common types of errors related to Arabic morphological generation was then made and several important types of errors were detailed. An existing commercial AMTS was then evaluated to determine which type(s) of error affected the translation to Arabic using that AMTS. The approach used in this chapter for evaluation identified several types of errors as affecting the output of the AMTS most drastically.

It is expected that the developers of the AMTS under evaluation would benefit from this evaluation by concentrating their research on treating the most common errors and those which affect the output of the AMTS most drastically. The categories of errors that were identified in this chapter can also help developers of AMT systems look for such errors in their output and treat them inside the AMTS. This will lead to a better AMT system that will produce outputs of better quality.

The evaluation of one AMT system in this contribution has revealed important information about output errors and their types. It is recommended that other AMT systems be evaluated in the same manner. This will lead to determining whether the types of errors that were identified in this chapter are indeed general for many AMT systems and therefore being general features of Arabic to be paid attention to in machine translation to Arabic.

As we happen to have collected a much larger corpus of pairs of translated sentences, we intend to do the evaluation for a much larger part of this corpus so as to reach statistically more conclusive results. Research could also be done on how to automate the evaluation above by using language tools that would be able to analyze the Arabic sentences and identify the types of errors automatically. For example, using an Arabic morphological analyzer, the output words could be automatically analyzed into a set of roots and associated morphological information. Then, a module could be developed that would check the types of errors by analyzing the information for all words in a sentence. Automating parts of the evaluation would allow the treatment of large corpora in shorter times.

Acknowledgements

We would like to thank M. A. Hussein (BSc student at the University of Sharjah) for having collected the sample English sentences from the web site http://dictionary.cambridge.org as well as their translations to Arabic using the web-based Machine Translation system Ajeeb (http://www.ajeeb.com).

References

Arnold, D.J., Sadler, L.G. & Humphreys, R.L. (1993). Evaluation: An Assessment. In *Machine Translation. Special Issue on Evaluation*, *8*(1–2), 1–24.

Chaumier, J., Mallen, M.C. & Van Slype, G. (1977). Evaluation du Système de Traduction Automatique SYSTRAN; Evaluation de la Qualité de la Traduction. *CEC Report Nr. 4*. Luxembourg.

Dahdah, A. (1995). معجم تصريف الأفعال العربية [Dictionary of the Conjugation of Arabic Verbs]. Beirut, Lebanon: Librairie du Liban.

ECL (Equipe Consortium Limited) (1996). *Survey of Machine Translation: Products and Services*. Summary of a report to the European Commission. Retrieved from http://www2.echo.lu/langeng/reps/mtsurvey/mtsurvey.html

Fowzan, M., Al-Harbi, S., Kazdar, S. & Al-Qahtani, H. (2000). المحلل الصرفي للأفعال [Morphological Analysis of Verbs]. B.Sc. project report, King Saoud University, College of Computer and Information Sciences, Computer Sciences Department, Riyadh, Saudi Arabia.

Guessoum, A. & Zantout, R. (2001). A Methodology for a Semi-Automatic Evaluation of the Language Coverage of Machine Translation System Lexicons. *Machine Translation, 16*(2), 127–149.

Guessoum, A. & Zantout, R. (2005). A Methodology for Evaluating Arabic Machine Translation Systems. *Machine Translation, 18*(4), 299–335.

Hamalawy, A. (1996). شذا العرف في فن الصرف [The Art of Morphology]. Beirut, Lebanon: مؤسسة الكتب الثقافية.

Hedberg, S. (1995). Machine Translation Comes of Age. *Computer Select*, September.

Van der Meer, J. (2003). At Last Translation Automation Becomes a Reality: An Anthology of the Translation Market. In *Proceedings of International Workshop of the European Association for Machine Translation/Controlled Language Applications Workshop* (pp. 180–184), Dublin, Ireland.

Nyberg, E. H. III and Mitamura, T. (1994). The KANT System: Fast, Accurate, High-Quality Translation in Practical Domains. In *Proceedings of the 14th International Conference on Computational Linguistics, COLING-92*, pp. 1069-1073.

Nyberg, E. H. III., Mitamura, T. & Carbonell, J.G. (1994). Evaluation Metrics for Knowledge-Based Machine Translation. In *Proceedings of COLING -94*.

Oren, T. (December 2004). *Machine Translation and the Global Blogosphere*. Retrieved from http://windsofchange.net.

Rajhi, A. (1979). التطبيق الصرفي [Applied Morphology]. Lebanon: دار النهضة العربية للطباعة و النشر.

Van Slype, G. (1979). *Critical Study of Methods for Evaluating the Quality of Machine Translation* (Final Report). Prepared for the Commission of the European Communities, Directorate for General Scientific and Technical Information Management (DG XIII), BR 19142, Brussels.

White, J., O'Connell, T. & O'Mara, F. (1994). *Machine Translation Program: 3Q94 Evaluation.*. In *Proceedings of the November 1994 Meeting of the Advanced Research Projects Agency*. Retrieved from http://ursula.georgetown.edu/mt_web/3Q94FR.htm

Index