ACL-05

Computational Approaches to Semitic Languages

Workshop Proceedings

29 June 2005
University of Michigan
Ann Arbor, Michigan, USA

Production and Manufacturing by
Omnipress Inc.
Post Office Box 7214
Madison, WI 53707-7214

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
75 Paterson Street, Suite 9
New Brunswick, NJ 08901
USA
Tel: +1-732-342-9100
Fax: +1-732-342-9339
acl@aclweb.org

Preface

Semitic Languages

The Semitic family includes many languages and dialects spoken by a large number of native speakers (around 300 Million). However, Semitic languages are still understudied. The most prominent members of this family are Arabic and its dialects, Hebrew, Amharic, Aramaic, Maltese and Syriac. Beyond their shared ancestry which is apparent through pervasive cognate sharing, a common characteristic of these languages is the rich and productive pattern-based morphology and similar syntactic constructions.

Previous Efforts

An increasing body of computational linguistics work is starting to appear for both Arabic and Hebrew. Arabic alone, as the largest member of the Semitic family, has been receiving a lot of attention lately in terms of dedicated workshops and conferences. These include, but are not limited to, the workshop on Arabic Language Resources and Evaluation (LREC 2002), a special session on Arabic processing in Traitement Automatique du Langage Naturel (TALN 2004), the Workshop on Computational Approaches to Arabic Script-based Languages (COLING 2004), and the NEMLAR Arabic Language Resources and Tools Conference in Cairo, Egypt (2004). This phenomenon has been coupled with a relative surge in resources for Arabic due to concerted efforts by the LDC and ELDA/ELRA. However, there is an apparent lag in the development of resources and tools for other Semitic languages. Often, work on individual Semitic languages, unfortunately, still tends to be done with limited awareness of ongoing research in other Semitic languages. Within the last four years, only three workshops addressed Semitic languages: an ACL 2002 Workshop on Computational Approaches to Semitic Languages and an MT Summit IX Workshop on Machine Translation for Semitic Languages in 2003, and the EAMT 2004, held in Malta, had a special session on Semitic languages.

Current Workshop

Welcome to the ACL 2005 Workshop on Computational Approaches to Semitic Languages. This workshop is a sequel to the ACL 2002 workshop and shares its goals of: (i) heightening awareness amongst Semitic-language researchers of shared breakthroughs and challenges, (ii) highlighting issues common to all Semitic languages as much as possible, (iii) encouraging the potential for developing coordinated approaches; and (iv) in addition, leveraging resource and tool creation for less prominent members of the Semitic language family.

We received 21 submissions, we accepted 12. The accepted papers cover several languages: Modern Standard Arabic, Dialectal Arabic, Hebrew, and Amharic. They cover a span of topics in computational linguistics, from morphological analysis and disambiguation and diacritization to information retrieval and document classification using both symbolic and statistical approaches.

We hope you enjoy reading this volume as much as we did.

The workshop organizers,

Kareem Darwish, Mona Diab, Nizar Habash

Organizers:

Kareem Darwish, German University in Cairo
Mona Diab, Columbia University Center for Computational Learning Systems
Nizar Habash, Columbia University Center for Computational Learning Systems

Program Committee:

Ibrahim A. Alkharashi (King Abdulaziz City for Science and Technology, Saudi Arabia)
Tim Buckwalter (Linguistic Data Consortium, USA)
Violetta Cavalli-Sforza (Carnegie Mellon University, USA)
Yaacov Choueka (Bar-Ilan University, Israel)
Joseph Dichy (Lyon University, France)
Martha Evens (Illinois Institute of Technology, USA)
Ali Farghaly (SYSTRAN Software, Inc.)
Alexander Fraser (USC/ISI)
Andrew Freeman (Mitre)
Alon Itai, (Technion, Israel)
George Kiraz (Beth Mardutho: The Syriac Institute, USA)
Katrin Kirchhoff (University of Washington, USA)
Alon Lavie (Carnegie Mellon University, USA)
Mohamed Maamouri (Linguistic Data Consortium, USA)
Uzzi Ornan (Technion, Israel)
Anne De Roeck (Open University, UK)
Michael Rosner (University of Malta, Malta)
Salim Roukos (IBM, USA)
Khalil Sima'an (University of Amsterdam, Netherlands)
Abdelhadi Soudi (ENIM, Rabat, Morocco)
Shuly Wintner (University of Haifa, Israel)
Remi Zajac (SYSTRAN Software, USA)

Invited Speaker:

Salim Roukos, IBM T. J. Watson Research Center

Table of Contents

Conference Program

Wednesday, June 29, 2005

9:00–9:15 Opening Remarks by Organizers

Session 1: Morphology

9:15–9:40 *Memory-based morphological analysis generation and part-of-speech tagging of Arabic*
Erwin Marsi, Antal van den Bosch and Abdelhadi Soudi

9:40–10:05 *A finite-state morphological grammar of Hebrew*
Shlomo Yona and Shuly Wintner

10:05–10:30 *Morphological Analysis and Generation for Arabic Dialects*
Nizar Habash, Owen Rambow and George Kiraz

10:30–11:00 Coffee Break

11:00–11:40 Invited Talk by Salim Roukos

Session 2: Applications I

11:40–12:05 *Examining the Effect of Improved Context Sensitive Morphology on Arabic Information Retrieval*
Kareem Darwish, Hany Hassan and Ossama Emam

12:05–12:30 *Modifying a Natural Language Processing System for European Languages to Treat Arabic in Information Processing and Information Retrieval Applications*
Gregory Grefenstette, Nasredine Semmar and Faiza Elkateb-Gara

12:30-2:15 Lunch Break

Wednesday, June 29, 2005 (continued)

Session 3: Part of Speech Tagging

2:15–2:40 *Choosing an Optimal Architecture for Segmentation and POS-Tagging of Modern Hebrew*
Roy Bar-Haim, Khalil Sima'an and Yoad Winter

2:40–3:05 *Part of Speech tagging for Amharic using Conditional Random Fields*
Sisay Fissaha Adafre

3:05–3:30 *POS Tagging of Dialectal Arabic: A Minimally Supervised Approach*
Kevin Duh and Katrin Kirchhoff

3:30–4:00 Coffee Break

Session 4: Applications II

4:00–4:25 *The Impact of Morphological Stemming on Arabic Mention Detection and Coreference Resolution*
Imed Zitouni, Jeffrey Sorensen, Xiaoqiang Luo and Radu Florian

4:25–4:50 *Classifying Amharic News Text Using Self-Organizing Maps*
Samuel Eyassu and Björn Gambäck

4:50–5:15 *Arabic Diacritization Using Weighted Finite-State Transducers*
Rani Nelken and Stuart M. Shieber

5:15–5:40 *An Integrated Approach for Arabic-English Named Entity Translation*
Hany Hassan and Jeffrey Sorensen

5:40–6:00 Parting Words & Discussion

Memory-based morphological analysis generation and part-of-speech tagging of Arabic

Erwin Marsi, Antal van den Bosch
ILK, Tilburg University
P.O. Box 90153
NL-5000 LE Tilburg
The Netherlands
{E.C.Marsi,Antal.vdnBosch}@uvt.nl

Abdelhadi Soudi
Center for Computational Linguistics
Ecole Nationale de L'Industrie Minérale
Rabat,
Morocco,
asoudi@gmail.com/asoudi@enim.ac.ma

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 – the construction of all possible analyses of isolated unvoweled wordforms – is performed as a letter-by-letter operation prediction task, where the operation encodes segmentation, part-of-speech, character changes, and vocalization. Part-of-speech tagging is carried out by a bi-modular tagger that has a subtagger for known words and one for unknown words. We report on the performance of the morphological analyzer and part-of-speech tagger. We observe that the tagger, which has an accuracy of 91.9% on new data, can be used to select the appropriate morphological analysis of words in context at a precision of 64.0 and a recall of 89.7.

1 Introduction

Memory-based learning has been successfully applied to morphological analysis and part-of-speech tagging in Western and Eastern-European languages (van den Bosch and Daelemans, 1999; Daelemans et al., 1996). With the release of the Arabic Treebank by the Linguistic Data Consortium (current version: 3), 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, as exemplified by Diab et al. (2004).

However, Arabic appears to be a special challenge for data-driven approaches. It is a Semitic language with a non-concatenative morphology. 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. An Arabic word may be composed of a stem consisting of a consonantal root and a pattern, affixes, and clitics. The affixes include inflectional markers for tense, gender, and number. The clitics may be either attached to the beginning of stems (proclitics) or to the end of stems (enclitics) and include possessive pronouns, pronouns, some prepositions, conjunctions and determiners.

Arabic verbs, for example, can be conjugated according to one of the traditionally recognized patterns. There are 15 triliteral forms, of which at least 9 are in 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. The broken plural system, for example, 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

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 1–8,
Ann Arbor, June 2005.

statistically minor plural patterns are also possible.

Various ways of accounting for Arabic morphology have been proposed. The type of account of Arabic morphology that is generally accepted by (computational) linguists is that proposed by (McCarthy, 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. For example, the Arabic stem *katab* ("he wrote") is composed of the morpheme *ktb* ("the notion of writing") and the vowel melody morpheme 'a-a'. The two are integrated according to the pattern CVCVC (C=consonant, V=vowel). This means that word structure in this morphology is not built linearly as is the case in concatenative morphological systems.

The attempts to model Arabic morphology in a two-level system (Kay's (1987) Finite State Model, Beesley's (1990; 1998) Two-Level Model and Kiraz's (1994) Multi-tape Two-Level Model) reflect McCarthy's separation of levels. It is beyond the scope of this paper to provide a detailed description of these models, but see (Soudi, 2002).

In this paper, 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. The following three sections describe our exploratory work on memory-based morphological analysis and PoS tagging, and integration of the two tasks. The final two sections contain a short discussion of related work and an overall conclusion.

2 Memory-based learning

Memory-based learning, also known as instance-based, example-based, or lazy learning (Aha et al., 1991; Daelemans et al., 1999), extensions of 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 allows it to deal with productive but low-frequency exceptions (Daelemans et al., 1999).

An instance consists of a fixed-length vector of n feature-value pairs, and 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 kernel function. Classification in memory-based learning is performed by the k-NN algorithm that searches for the k 'nearest neighbours' according to the $\Delta(X, Y)$ kernel function[1].

The distance function and the classifier can be refined by several kernel plug-ins, such as feature weighting (assigning larger distance to mismatches on important features), and distance weighting (assigning a smaller vote in the classification to more distant nearest neighbors). Details can be found in (Daelemans et al., 2004).

3 Morphological analysis

We focus first on morphological analysis . Training on data extracted from the Arabic Treebank, we induce a morphological analysis generator which we control for undergeneralization (recall errors) and overgeneralization (precision errors).

3.1 Data

3.1.1 Arabic Treebank

Our point of departure is the Arabic Treebank 1 (ATB1), version 3.0, distributed by LDC in 2005, more specifically the "after treebank" PoS-tagged data. Unvoweled tokens as they appear in the original news paper are accompanied in the treebank by vocalized versions; all of their morphological analyses are generated by means of Tim Buckwalter's Arabic Morphological Analyzer (Buckwalter, 2002), and the appropriate morphological analysis is singled out. An example is given in Figure 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 analysis is rule-based, and basically consists of three steps. First, all possible segmentations of the input string

[1] All experiments with memory-based learning were performed with TiMBL, version 5.1 (Daelemans et al., 2004), available from `http://ilk.uvt.nl`.

```
INPUT STRING: \331\203\330\252\330\250
LOOK-UP WORD: ktb
     Comment:
       INDEX: P2W38
  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
```

Figure 1: Example token from ATB1

in terms of prefixes (0 to 4 characters long), stems (at least one character), and suffixes (0 to 6 characters long) are generated. Next, dictionary lookup is used to determine if these segments are existing morphological units. Finally, the numbers of analyses is further reduced by checking for the mutual compatibility of prefix+stem, stem+suffix, and prefix+stem in three compatibility tables. The resulting analyses have to a certain extent been manually checked. Most importantly, a star (*) preceding a solution indicates that this is the correct analysis in the given context.

3.1.2 Preprocessing

We grouped the 734 files from the treebank into eleven parts of approximately equal size. Ten parts were used for training and testing our morphological analyzer, while the final part was used as held-out material for testing the morphological analyzer in combination with the PoS tagger (described in Section 4).

In the corpus the number of analyses per word is not entirely constant, either due to the automatic generation method or to annotator edits. As our initial goal is to predict all possible analyses for a given word, regardless of contextual constraints, we first created a *lexicon* that maps every word to all analyses encountered and their respective frequencies From the 185,061 tokens in the corpus, we extracted 16,626 unique word types – skipping punctuation tokens – and 129,655 analyses, which amounts to 7.8 analyses per type on average.

```
= = = = = k t b = = = ka/PREP+;ka;k;ku
= = = = k t b = = = = a/PREP+t;uti;ata;t;utu
= = = k t b = = = = = ab/PV+a/PVSUFF_SUBJ:3MS+;
                      b/NOUN_PROP+;ub/NOUN+i/CASE_DEF_GEN+;
                      ub/NOUN+a/CASE_DEF_ACC+;
                      ub/NOUN+K/CASE_INDEF_GEN+;
                      ib/PV_PASS+a/PVSUFF_SUBJ:3MS+;
                      ub/NOUN+N/CASE_INDEF_NOM+;ub/NOUN+;
                      ub/NOUN+u/CASE_DEF_NOM+
```

Figure 2: Instances for the analyses of the word *ktb* in Figure 1.

3.1.3 Creating instances

These separate lexicons were created for training and testing material. The lexical entries in a lexicon were converted to *instances* suitable to memory-based learning of the mapping from words to their analyses (van den Bosch and Daelemans, 1999). Instances consist of a sequence of feature values and a corresponding class, representing a potentially complex morphological operation.

The features are created by sliding a window over the unvoweled look-up word, resulting in one instance for each character. Using a 5-1-5 window yields 11 features, i.e. the input character in focus, plus the five preceding and five following characters. The equal sign (=) is used as a filler symbol.

The instance classes represent the morphological analyses. The classes corresponding to a word's characters should enable us to derive all associated analyses. This implies that the classes need to encode several aspects simultaneously: vocalization, morphological segmentation and tagging. The following template describes the format of classes:

```
class = subanalysis; subanalysis; ...

subanalysis = preceding vowels & tags +
              input character +
              following vowels & tags
```

For example, the classes of the instances in Figure 2 encode the ten solutions for the word *ktb* in Figure 1. The ratio behind this encoding is that it allows for a simple derivation of the solution, akin to the way that the pieces of a jigsaw puzzle can be combined. We can exhaustively try all combinations of the subanalyses of the classes, and check if the right side of one subanalysis matches the left side of a subsequent subanalysis. This reconstruction process is illustrated in Figure 3 (only two reconstructions are depicted, corresponding to SOLUTION 1 and SOLUTION 4). For example, the subanalysis `ka` from the first class in Figure 2 matches the subanalysis `ata` from the sec-

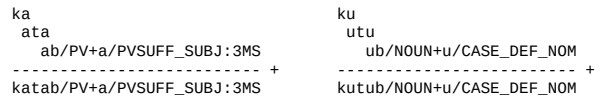

Figure 3: Illustration of how two morphological analyses are reconstructed from the classes in Figure 2.

ond class, which in turn matches the subanalysis `ab/PV+a/PVSUFF_SUBJ:3MS` from the third class; together these constitute the complete analysis `katab/PV+a/PVSUFF_SUBJ:3MS`.

3.2 Initial Experiments

To test the feasibility of our approach, we first train and test on the full data set. Timbl is used with its default settings (overlap distance function, gain-ratio feature weighting, $k = 1$). Rather than evaluating on the accuracy of predicting the complex classes, we evaluate on the complete correctness of all reconstructed analyses, in terms of precision, recall, and F-score (van Rijsbergen, 1979). As expected, this results in a near perfect recall (97.5). The precision, however, is much lower (52.5), indicating a substantial amount of analysis overgeneration; almost one in two generated analyses is actually not valid. With an F-score of only 68.1, we are clearly not able to reproduce the training data perfectly.

Next we split the data in 9 parts for training and 1 part for testing. The k-NN classifier is again used with its default settings. Table 1 shows the results broken down into known and unknown words. As known words can be looked up in the lexicon derived from the training material, the first row presents the results with lookup and the second row without lookup (that is, with prediction). The fact that even with lookup the performance is not perfect shows that the upper bound for this task is not 100%. The reason is that apparantly some words in the test material have received analyses that never occur in the training material and vice versa. For known words without lookup, the recall is still good, but the precision is low. This is consistent with the initial results mentioned above. For unknown words, both recall and precison are much worse, indicating rather poor generalization.

To sum up, there appear to be problems with both the precision and the recall. The precision is low for known words and even worse for unknown words.

	#Wrds	Prec	Rec	F
Known with lookup	3220	92.6	98.1	95.3
Known without lookup	3220	49.9	95.0	65.5
Unknown	847	22.8	26.8	24.7

Table 1: Results of initial experiments split into known and unknown words, and with and without lookup of known words.

	#Wrds	Prec	Rec	F
Known	3220	15.6	99.0	26.9
Unknown	847	3.9	66.8	7.5

Table 2: Results of experiments for improving the recall, split into known and unknown words.

Analysis overgeneration seems to be a side effect of the way we encode and reconstruct the analyses. The recall is low for unknown words only. There appear to be at least two reasons for this undergeneration problem. First, if just one of the predicted classes is incorrect (one of the pieces of the jigsaw puzzle is of the wrong shape) then many, or even all of the reconstructions fail. Second, some generalizations cannot be made, because infrequent classes are overshadowed by more frequent ones with the same features. Consider, for example, the instance for the third character (l) of the word *jEl*:

```
= = = j E l = = = = =
```

Its real class in the test data is:

```
al/VERB_PERFECT+;ol/NOUN+
```

When the k-NN classifier is looking for its nearest neighbors, it finds three; two with a "verb imperfect" tag, and one with a "noun" tag.

```
{ al/VERB_IMPERFECT+ 2, ol/NOUN+ 1}
```

Therefore, the class predicted by the classifier is `al/VERB_IMPERFECT+`, because this is the majority class in the NN-set. So, although a part of the correct solution is present in the NN-set, simple majority voting prevents it from surfacing in the output.

3.3 Improving recall

In an attempt to address the low recall, we revised our experimental setup to take advantage of the complete NN-set. As before, the k-NN classifier is used,

	Prec	Rec	F
Known	58.6 (0.4)	66.6 (0.5)	62.4 (0.3)
Unknown	28.7 (3.7)	37.2 (1.2)	32.2 (2.5)
All	53.4 (1.2)	62.2 (0.6)	57.5 (0.8)

Table 3: Average results and SD of the 10-fold CV experiment, split into known and unknown words

but rather than relying on the classifier to do the majority voting over the (possibly weighted) classes in the k-NN set and to output a *single* class, we perform a reconstruction of analyses combining *all* classes in the k-NN set. To allow for more classes in k-NN's output, we increase k to 3 while keeping the other settings as before. As expected, this approach increases the number of analyses. This, in turn, increases the recall dramatically, up to nearly perfect for known words; see Table 2. However, this gain in recall is at the expense of the precision, which drops dramatically. So, although our revised approach solves the issues above, it introduces massive overgeneration.

3.4 Improving precision

We try to tackle the overgeneration problem by filtering the analyses in two ways. First, by ranking the analyses and limiting output to the n-best. The ranking mechanism relies on the distribution of the classes in the NN-set. Normally, some classes occur more frequently than others in the NN-set. During the reconstruction of a particular analysis, we sum the frequencies of the classes involved. The resulting score is then used to rank the analyses in decreasing order, which we filter by taking the n-best.

The second filter employs the fact that only certain sequences of morphological tags are valid. Tag bigrams are already implicit in the way that the classes are constructed, because a class contains the tags preceding and following the input character. However, cooccurrence restrictions on tags may stretch over longer distances; tag trigram information is not available at all. We therefore derive a frequency list of all tag trigrams occurring in the training data. This information is then used to filter analyses containing tag trigrams occurring below a certain frequency threshold in the training data.

Both filters were optimized on the fold that was used for testing so far, maximizing the overall F-score. This yieled an n-best value of 40 and tag frequency treshold of 250. Next, we ran a 10-fold cross-validation experiment on all data (except the held out data) using the method described in the previous section in combination with the filters. Average scores of the 10 folds are given in Table 3. In comparison with the initial results, both precision and recall on unknown words has improved, indicating that overgeneration and undergeneration can be midly counteracted.

3.5 Discussion

Admittedly, the performance is not very impressive. We have to keep in mind, however, that the task is not an easy one. It includes vowel insertion in ambiguous root forms, which – in contrast to vowel insertion in prefixes and suffixes – is probably irregular and unpredictable, unless the appropriate stem would be known. As far as the evaluation is concerned, we are unsure whether the analyses found in the treebank for a particular word are exhaustive. If not, some of the predictions that are currently counted as precision errors (overgeneration) may in fact be correct alternatives.

Since instances are generated for each type rather than for each token in the data, the effect of token frequency on classification is lost. For example, instances from frequent tokens are more likely to occur in the k-NN set, and therefore their (partial) analyses will show up more frequently. This is an issue to explore in future work. Depending on the application, it may also make sense to optimize on the correct prediction of unkown words, or on increasing only the recall.

4 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[2]. 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. We conclude this

[2]In our experiments we used the MBT software package, version 2 (Daelemans et al., 2003), available from `http://ilk.uvt.nl/`.

```
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
...
```

Figure 4: Part of an ATB1 sentence with unvoweled words (left) and their respective PoS tags (right).

section by describing the effect of using the output of the morphological analyzer as extra input to the tagger.

4.1 Data preparation

While the morphological analyzer attempts to generate all possible analyses for a given unvoweled 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 using the * marker. We developed a PoS tagger that is trained to predict an unvoweled word in context, a concatenation of the PoS tags of its morphemes. Essentially this is the task of the morphological analyzer without segmentation and vocalization. Figure 4 shows part of a sentence where for each word the respective tag is given in the second column. Concatenation is marked by the delimiter +.

We trained on the full ten folds used in the previous sections, and tested on the eleventh fold. The training set thus contains 150,966 words in 4,601 sentences; the test set contains 15,102 words in 469 sentences. 358 unique tags occur in the corpus. In the test set 947 words occur that do not occur in the training set.

4.2 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 in (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. 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 a held-out validation set (taken from the training data).

The unknown-word tagger 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 characters and the four suffix characters 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 validation material.

The accuracy of the tagger on the held-out corpus is 91.9% correctly assigned tags. On the 14155 known words in the test set the tagger attains an accuracy of 93.1%; on the 947 unknown words the accuracy is considerably lower: 73.6%.

5 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

Part-of-speech source	All words Precision	All words Recall	Known words Precision	Known words Recall	Unknown words Precision	Unknown words Recall
Gold standard	70.1	97.8	75.8	99.5	30.2	73.4
Predicted	64.0	89.7	69.8	92.0	23.9	59.0

Table 4: Precision and recall of the identification of the contextually appropriate morphological analysis, measured on all test words and split on known words and unknown words. The top line represents the upper-bound experiment with gold-standard PoS tags; the bottom line represents the experiment with predicted PoS tags.

amounts to predicting the solution that is preceded by "*" 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. We employed the following two rules to implement this: (1) If the input word occurs in the training data, then look up the morphological analyses of the word in the training-based lexicon, and return all morphological analyses with a PoS content matching the tag predicted by the tagger. (2) Otherwise, let the memory-based morphological analyzer produce analyses, and return all analyses with a PoS content matching the predicted tag.

We first carried out an experiment integrating the output of the morphological analyzer and the PoS tagger, faking perfect tagger predictions, in order to determine the upper bound of this approach. Rather than predicting the PoS tag with MBT, we directly derived the PoS tag from the annotations in the treebank. The upper result line in Table 4 displays the precision and recall scores on the held-out data of identifying the appropriate morphological analysis, i.e. the solution marked by *. Unsurprisingly, the recall on known words is 99.5%, since we are using the gold-standard PoS tag which is guaranteed to be among the training-based lexicon, except for some annotation discrepancies. More interestingly, about one in four analyses of known words matching on PoS tags actually mismatches on vowel or consonant changes, e.g. because it represents a different stem – which is unpredictable by our method.

About one out of four unknown words has morphological analyses that do not match the gold-standard PoS (a recall of 73.4); at the same time, a considerable amount of overgeneration of analyses accounts for the low amount of analyses that matches (a precision of 30.2).

Next, the experiment was repeated with *predicted* PoS tags and morphological analyses. The results are presented in the bottom result line of Table 4. The precision and recall of identifying correct analyses of known words degrades as compared to the upper-bounds results due to incorrect PoS tag predictions. On unknown words the combination of heavy overgeneration by the morphological analyzer and the 73.6% accuracy of the tagger leads to a low precision of 23.9 and a fair recall of 59.0. On both known and unknown words the integration of the morphological analyzer and the tagger is able to narrow down the analyses by the analyzer to a subset of matching analyses that in about nine out of ten cases contains the "* SOLUTION" word.

6 Related work

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 (Kay, 1987; Beesley, 1990; Kiraz, 1994; Beesley, 1998; Cavalli-Sfora et al., 2000; Soudi, 2002).

We are not aware of any machine-learning approach to Arabic morphology, but find related issues treated in (Daya et al., 2004), who propose a machine-learning method augmented with linguistic constraints to identifying roots in Hebrew words – a related but reverse task to ours. Arabic PoS tagging seems to have attracted some more attention. 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. Diab et al. (2004) describe a part-of-speech tagger based on support vector machines that is trained on tokenized data (clitics are separate tokens), reporting a tagging accuracy of 95.5%.

7 Conclusions

We investigated the application of memory-based learning (k-nearest neighbor classification) to morphological analysis and PoS tagging of unvoweled written Arabic, using the ATB1 corpus as training and testing material. The morphological analyzer was shown to attain F-scores of 0.32 on *unknown* words when predicting all aspects of the analysis, including vocalization (a partly unpredictable task, certainly if no context is available). The PoS tagger attains an accuracy of about 74% on unknown words, and 92% on all words (including known words). A combination of the two which selects from the set of generated analyses a subset of analyses with the PoS predicted by the tagger, yielded a recall of the contextually appropriate analysis of 0.90 on test words, yet a low precision of 0.64 largely caused by overgeneration of invalid analyses.

We make two final remarks. First, memory-based morphological analysis of Arabic words appears feasible, but its main limitation is its inevitable inability to recognize the appropriate stem of unknown words on the basis of the ambiguous root form input; our current method simply overgenerates vocalizations, keeping high recall at the cost of low precision. Second, memory-based PoS tagging of written Arabic text also appears to be feasible; the observed performances are roughly comparable to those observed for other languages. The PoS tagging task as we define it is deliberately separated from the problem of vocalization, 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.

References

D. W. Aha, D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

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

K. Beesley. 1998. Consonant spreading in Arabic stems. In *Proceedings of COLING-98*.

T. Buckwalter. 2002. Buckwalter Arabic morphological analyzer version 1.0. Technical Report LDC2002L49, Linguistic Data Consortium. available from `http://www.ldc.upenn.edu/`.

V. Cavalli-Sfora, A. Soudi, and M. Teruko. 2000. Arabic morphology generation using a concatenative strategy. In *Proceedings of the First Conference of the North-American Chapter of the Association for Computational Linguistics*, Seattle, WA, USA.

T. M. Cover and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

W. Daelemans, 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 Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.

W. Daelemans, 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.

W. Daelemans, J. Zavrel, A. van den Bosch, and K. van der Sloot. 2003. MBT: Memory based tagger, version 2.0, reference guide. ILK Technical Report 03-13, Tilburg University.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2004. TiMBL: Tilburg memory based learner, version 5.1, reference guide. ILK Technical Report 04-02, Tilburg University.

E. Daya, D. Roth, and S. Wintner. 2004. Learning Hebrew roots: Machine learning with linguistic constraints. In *Proceedings of EMNLP'04*, Barcelona, Spain.

M. Diab, K. Hacioglu, and D. Jurafsky. 2004. Automatic tagging of arabic text: From raw text to base phrase chunks. In *Proceedings of HLT/NAACL-2004*.

A. Freeman. 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.

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

S. Khoja. 2001. APT: Arabic part-of-speech tagger. In *Proceedings of the Student Workshop at NAACL-2001*.

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

J. McCarthy. 1981. A prosodic theory of non-concatenative morphology. *Linguistic Inquiry*, 12:373–418.

A. Soudi. 2002. *A Computational Lexeme-based Treatment of Arabic Morphology*. Ph.D. thesis, Mohamed V University (Morocco) and Carnegie Mellon University (USA).

A. van den Bosch and W. Daelemans. 1999. Memory-based morphological analysis. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 285–292, San Francisco, CA. Morgan Kaufmann.

C.J. van Rijsbergen. 1979. *Information Retrieval*. Buttersworth, London.

A finite-state morphological grammar of Hebrew

Shlomo Yona
Department of Computer Science
University of Haifa
31905 Haifa, Israel
shlomo@cs.haifa.ac.il

Shuly Wintner
Department of Computer Science
University of Haifa
31905 Haifa, Israel
shuly@cs.haifa.ac.il

Abstract

Morphological analysis is a crucial component of several natural language processing tasks, especially for languages with a highly productive morphology, where stipulating a full lexicon of surface forms is not feasible. We describe HAMSAH (HAifa Morphological System for Analyzing Hebrew), a morphological processor for Modern Hebrew, based on finite-state linguistically motivated rules and a broad coverage lexicon. The set of rules comprehensively covers the morphological, morpho-phonological and orthographic phenomena that are observable in contemporary Hebrew texts. Reliance on finite-state technology facilitates the construction of a highly efficient, completely bidirectional system for analysis and generation. HAMSAH is currently the broadest-coverage and most accurate freely-available system for Hebrew.

1 Hebrew morphology: the challenge

Hebrew, like other Semitic languages, has a rich and complex morphology. The major word formation machinery is root-and-pattern, where roots are sequences of three (typically) or more consonants, called *radicals*, and patterns are sequences of vowels and, sometimes, also consonants, with "slots" into which the root's consonants are being inserted (interdigitation). Inflectional morphology is highly productive and consists mostly of suffixes, but sometimes of prefixes or circumfixes.

As an example of root-and-pattern morphology, consider the Hebrew[1] roots *g.d.l* and *r.e.m* and the patterns *hCCCh* and *CiCwC*, where the 'C's indicate the slots. When the roots combine with these patterns the resulting lexemes are *hgdlh*, *gidwl*, *hremh*, *riewm*, respectively. After the root combines with the pattern, some morpho-phonological alternations take place, which may be non-trivial: for example, the *htCCCwt* pattern triggers assimilation when the first consonant of the root is *t* or *d*: thus, *d.r.e+htCCCwt* yields *hdrewt*. The same pattern triggers metathesis when the first radical is *s* or *e*: *s.d.r+htCCCwt* yields *hstdrwt* rather than the expected *htsdrwt*. Frequently, root consonants such as *w* or *i* are altogether missing from the resulting form. Other *weak* paradigms include roots whose first radical is *n* and roots whose second and third radicals are identical. Thus, the roots *q.w.m*, *g.n.n*, *n.p.l* and *i.c.g*, when combining with the *hCCCh* pattern, yield the seemingly similar lexemes *hqmh*, *hgnh*, *hplh* and *hcgh*, respectively.

The combination of a root with a pattern produces a *base* (or a *lexeme*), which can then be inflected in various forms. Nouns, adjectives and numerals inflect for number (singular, plural and, in rare cases, also dual) and gender (masculine or feminine). In addition, all these three types of nominals have two phonologically distinct forms, known as the *absolute* and *construct* states. Unfortunately, in the standard orthography approximately half of the nomi-

[1] To facilitate readability we sometimes use a transliteration of Hebrew using ASCII characters:

a	b	g	d	h	w	z	x	v	i	k
א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ
l	m	n	s	y	p	c	q	r	e	t
ל	מ	נ	ס	ע	פ	צ	ק	ר	ש	ת

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 9–16,
Ann Arbor, June 2005.

nals appear to have identical forms in both states, a fact which substantially increases the ambiguity. In addition, nominals take pronominal suffixes which are interpreted as possessives. These inflect for number, gender and person: *spr+h→sprh* "her book", *spr+km→sprkm* "your book", etc. As expected, these processes involve certain morphological alternations, as in *mlkh+h→mlkth* "her queen", *mlkh+km→mlktkm* "your queen". Verbs inflect for number, gender and person (first, second and third) and also for a combination of tense and aspect, which is traditionally analyzed as having the values past, present, future, imperative and infinite. Verbs can also take pronominal suffixes, which in this case are interpreted as direct objects, but such constructions are rare in contemporary Hebrew of the registers we are interested in.

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]*, does not distinguish between *[o]* and *[u]* and leaves many of the *[i]* vowels unspecified. Furthermore, the single letter וֹ *w* is used both for the vowels *[o]* and *[u]* and for the consonant *[v]*, whereas יִ *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 (*b* "in", *k* "as", *l* "to" and *m* "from"), the definite article *h* "the", the coordinating conjunction *w* "and" and some subordinating conjunctions (such as *e* "that" and *ke* "when"), all attach to the words which immediately follow them. Thus, a form such as *ebth* can be read as a lexeme (the verb "capture", third person singular feminine past), as *e+bth* "that+field", *e+b+th* "that+in+tea", *ebt+h* "her sitting" or even as *e+bt+h* "that her daughter". When a definite nominal is prefixed by one of the prepositions *b*, *k* or *l*, the definite article *h* is assimilated with the preposition and the resulting form becomes ambiguous as to whether or not it is definite: *bth* can be read either as *b+th* "in tea" or as *b+h+th* "in the tea".

An added complexity stems from the fact that there exist two main standards for the Hebrew script: one in which vocalization diacritics, known as *niqqud* "dots", decorate the words, and another in which the dots are missing, and other characters represent some, but not all of the vowels. Most of the texts in Hebrew are of the latter kind; unfortunately, different authors use different conventions for the undotted script. Thus, the same word can be written in more than one way, sometimes even within the same document, again adding to the ambiguity.

In light of the above, morphological analysis of Hebrew forms is a non-trivial task. Observe that simply stipulating a list of surface forms is not a viable option, both because of the huge number of potential forms and because of the complete inability of such an approach to handle out-of-lexicon items; the number of such items in Hebrew is significantly larger than in European languages due to the combination of prefix particles with open-class words such as proper names. The solution must be a dedicated morphological analyzer, implementing the morphological and orthographic rules of the language.

Several morphological processors of Hebrew have been proposed, including works by Choueka (1980; 1990), Ornan and Kazatski (1986), Bentur et al. (1992) and Segal (1997); see a survey in Wintner (2004). Most of them are proprietary and hence cannot be fully evaluated. However, the main limitation of existing approaches is that they are ad-hoc: the rules that govern word formation and inflection are only implicit in such systems, usually intertwined with control structures and general code. This makes the maintenance of such systems difficult: corrections, modifications and extensions of the lexicon are nearly impossible. An additional drawback is that all existing systems can be used for analysis but not for generation. Finally, the efficiency of such systems depends on the quality of the code, and is sometimes sub-optimal.

2 Finite-state technology

Finite-state technology (Beesley and Karttunen, 2003) solves the three problems elegantly. It provides a language of extended regular expressions which can be used to define very natural linguistically motivated grammar rules. Such expressions can then be compiled into finite-state networks (automata and transducers), on which efficient algorithms can be applied to implement both analysis and generation. Using this methodology, a computational linguist can design rules which closely follow standard linguistic notation, and automatically ob-

tain a highly efficient morphological processor.

While the original Two-Level formulation (Koskenniemi, 1983) of finite-state technology for morphology was not particularly well suited to Semitic languages (Lavie et al., 1988), modifications of the Two-Level paradigm and more advanced finite-state implementations have been applied successfully to a variety of Semitic languages, including Ancient Akkadian (Kataja and Koskenniemi, 1988), Syriac (Kiraz, 2000) and Arabic. In a number of works, Beesley (1996; 1998; 2001) describes a finite-state morphological analyzer of Modern Standard Arabic which handles both inflectional and derivational morphology, including interdigitation. In the following section we focus on a particular finite-state toolbox which was successfully used for Arabic.

In this work we use XFST (Beesley and Karttunen, 2003), an extended regular expression language augmented by a sophisticated implementation of several finite-state algorithms, which can be used to compactly store and process very large-scale networks. XFST grammars define a binary relation (a *transduction*) on sets of strings: a grammar maps each member of a (possibly infinite) set of strings, known as the *surface*, or *lower* language, to a set of strings (the *lexical*, or *upper* language). The idea is that the surface language defines all and only the grammatical words in the language; and each grammatical word is associated with a set of lexical strings which constitutes its *analyses*. As an example, the surface string *ebth* may be associated by the grammar with the set of lexical strings, or analyses, depicted in figure 1.

XFST enables the definition of *variables*, whose values, or *denotations*, are sets of strings, or languages. Grammars can set and use those variables by applying a variety of *operators*. For example, the *concatenation* operator (unfortunately indicated by a space) can be used to concatenate two languages: the expression ‘`A B`’ denotes the set of strings obtained by concatenating the strings in `A` with the strings in `B`. Similarly, the operator ‘`|`’ denotes set union, ‘`&`’ denotes intersection, ‘`~`’ set complement, ‘`-`’ set difference and ‘`*`’ Kleene closure; ‘`$A`’ denotes the set of strings containing at least one instance of a string from `A` as a substring. The empty string is denoted by ‘`0`’ and ‘`?`’ stands for any alphabet symbol. Square brackets are used for bracketing.

In addition to sets of strings, XFST enables the definition of binary relations over such sets. By default, every set is interpreted as the identity relation, whereby each string is mapped to itself. But relations can be explicitly defined using a variety of operators. The ‘`.x.`’ operator denotes cross product: the expression ‘`A.x.B`’ denotes the relation in which each string in `A` is mapped to each string in `B`. An extremely useful operation is composition: denoted by ‘`.o.`’, it takes two *relations*, `A` and `B`, and produces a new relation of pairs (a, c) such that there exists some b that (a, b) is a member of `A` and (b, c) is a member of `B`.

Finally, XFST provides also several *replace* rules. Expressions of the form ‘`A->B || L _ R`’ denote the relation obtained by replacing strings from `A` by strings from `B`, whenever the former occur in the context of strings from `L` on the left and `R` on the right. Each of the context markers can be replaced by the special symbol ‘`.#.`’, indicating a word boundary. For example, the expression ‘`[h]->[t] || ? _ .#.`’ replaces occurrences of ‘`h`’ by ‘`t`’ whenever the former occurs before the end of a word. Composing this example rule on an (identity) relation whose strings are various words results in replacing final `h` with final `t` in all the words, not affecting the other strings in the relation.

XFST supports diverse alphabets. In particular, it supports UTF-8 encoding, which we use for Hebrew (although subsequent examples use a transliteration to facilitate readability). Also, the alphabet can include *multi-character symbols*; in other words, one can define alphabet symbols which consist of several (print) characters, e.g., ‘`number`’ or ‘`tense`’. This comes in handy when *tags* are defined, see below. Characters with special meaning (such as ‘`+`’ or ‘`[`’) can be escaped using the symbol ‘`%`’. For example, the symbol ‘`%+`’ is a literal plus sign.

Programming in XFST is different from programming in high level languages. While XFST rules are very expressive, and enable a true implementation of some linguistic phenomena, it is frequently necessary to specify, within the rules, information that is used mainly for “book-keeping”. Due to the limited memory of finite-state networks, such information is encoded in *tags*, which are multi-character symbols attached to strings. These tags

```
[+verb][+id]9430[+base]ebt[+root]ebt[+binyan]+Pa'al[+agr]+3p/F/Sg[+tense]+past
[+verb][+id]1541[+base]ebh[+root]ebh[+binyan]+Pa'al[+agr]+3p/F/Sg[+tense]+past
[+conj]e[+prep]b[+noun][+id]19804[+base]th[+gender]+M[+number]+Sg[+construct]+true
[+conj]e[+prep]b[+noun][+id]19804[+base]th[+gender]+M[+number]+Sg[+construct]+false
[+conj]e[+prep]b[+defArt][+noun][+id]19804[+base]th[+gender]+M[+number]+Sg[+construct]+false
[+conj]e[+noun][+id]19130[+base]bth[+gender]+F[+number]+Sg[+construct]+false
[+conj]e[+noun][+id]1379[+base]bt[+gender]+F[+number]+Sg[+construct]+false[+poss]+3p/F/Sg
[+noun][+id]17280[+base]ebt[+gender]+F[+number]+Sg[+construct]+false[+poss]+3p/F/Sg
```

Figure 1: The analyses of the surface string שבתה *ebth*

can be manipulated by the rules and thus propagate information among rules. For example, nouns are specified for *number*, and the number feature is expressed as a concatenation of the tag `number` with the multi-character symbol `+singular` or `+plural`. Rules which apply to plural nouns only can use this information: if `nouns` is an XFST variable denoting the set of all nouns, then the expression `$[number %+plural] .o. nouns` denotes only the plural nouns. Once all linguistic processing is complete, "book-keeping" tags are erased.

3 A morphological grammar of Hebrew

The importance of morphological analysis as a preliminary phase in a variety of natural language processing applications cannot be over-estimated. The lack of good morphological analysis and disambiguation systems for Hebrew is reported as one of the main bottlenecks of a Hebrew to English machine translation system (Lavie et al. (2004)). The contribution of our system is manyfold:

- HAMSAH is the broadest-coverage and most accurate publicly available morphological analyzer of Modern Hebrew. It is based on a lexicon of over 20,000 entries, which is constantly being updated and expanded, and its set of rules cover all the morphological, morphophonological and orthographic phenomena observed in contemporary Hebrew texts. Compared to Segal (1997), our rules are probably similar in coverage but our lexicon is significantly larger. HAMSAH also supports non-standard spellings which are excluded from the work of Segal (1997).
- The system is fully reversible: it can be used both for analysis and for generation.
- Due to the use of finite-state technology, the system is highly efficient. While the network has close to 2 million states and over 2 million arcs, its compiled size is approximately 4Mb and analysis is extremely fast (between 50 and 100 words per second).
- Morphological knowledge is expressed through linguistically motivated rules. To the best of our knowledge, this is the first formal grammar for the morphology of Modern Hebrew.

The system consists of two main components: a *lexicon* represented in Extensible Markup Language (XML), and a set of finite-state rules, implemented in XFST. The use of XML supports standardization, allows a format that is both human and machine readable, and supports interoperability with other applications. For compatibility with the rules, the lexicon is automatically converted to XFST by dedicated programs. We briefly describe the lexicon in section 3.1 and the rules in section 3.2.

3.1 The lexicon

The lexicon is a list of lexical entries, each with a *base* (citation) form and a unique *id*. The base form of nouns and adjectives is the absolute singular masculine, and for verbs it is the third person singular masculine, past tense. It is listed in dotted and undotted script as well as using a one-to-one Latin transliteration. Figure 2 depicts the lexical entry of the word *bli* "without". In subsequent examples we retain only the transliteration forms and suppress the Hebrew ones.

```
<item dotted="בְּלִי" id="4917"
   translit="bli" undotted="בלי">
  <conjunction type="coord"/>
</item>
```

Figure 2: The lexical entry of *bli* "without"

The lexicon specifies morpho-syntactic features (such as gender or number), which can later be used by parsers and other applications. It also lists several lexical proerties which are specifically targeted at morphological analysis. A typical example is the feminine suffix of adjectives, which can be one of *h*, *it* or *t*, and cannot be predicted from the base form. The lexicon lists information pertaining to non-default behavior with idiosyncratic entries.

Adjectives inflect regularly, with few exceptions. Their citation form is the absolute singular masculine, which is used to generate the feminine form, the masculine plural and the feminine plural. An additional dimension is status, which can be absolute or construct. Figure 3 lists the lexicon entry of the adjective *yilai* "supreme": its feminine form is obtained by adding the *t* suffix (hence `feminine="t"`). Other features are determined by default. This lexicon entry yields *yilai*, *yilait*, *yilaiim*, *yilaiwt* etc.

```
<item id="13852" translit="yilai">
 <adjective feminine="t" />
</item>
```

Figure 3: A lexicon item for *yilai* "supreme"

Similarly, the citation form of nouns is the absolute singular masculine form. Hebrew has grammatical gender, and the gender of nouns that denote animate entities coincides with their natural gender. The lexicon specifies the feminine suffix via the *feminine* attribute. Nouns regularly inflect for number, but some nouns have only a plural or only a singular form. The plural suffix (*im* for masculine, *wt* for feminine by default) is specified through the *plural* attribute. Figure 4 demonstrates a masculine noun with an irregular plural suffix, *wt*.

```
<item id="5044" translit="ewlxn">
 <noun gender="masculine"
       number="singular"
       plural="wt" /></item>
```

Figure 4: A lexicon item for the noun *ewlxn* "table"

Closed-class words are listed in the lexicon in a similar manner, where the specific category determines which attributes are associated with the citation form. For example, some adverbs inflect for person, number and gender (e.g., *lav* "slowly"), so this is indicated in the lexicon. The lexicon also specifies the person, number and gender of pronouns, the type of proper names (location, person, organization), etc. The lexical representation of verbs is more involved and is suppressed for lack of space.

Irregularities are expressed directly in the lexicon, in the form of additional or alternative lexical entries. This is facilitated through the use of three optional elements in lexicon items: *add*, *replace* and *remove*. For example, the noun *chriim* "noon" is also commonly spelled *chrim*, so the additional spelling is specified in the lexicon, along with the standard spelling, using *add*. As another example, consider Segolate nouns such as *bwqr* "morning". Its plural form is *bqrim* rather than the default *bwqrim*; such stem changing behavior is specified in the lexicon using *replace*. Finally, the verb *ykwl* "can" does not have imperative inflections, which are generated by default for all verbs. To prevent the default behavior, the superfluous forms are *remove*d.

The processing of irregular lexicon entries requires some explanation. Lexicon items containing *add*, *remove* and *replace* elements are included in the general lexicon without the *add*, *remove* and *replace* elements, which are listed in special lexicons. The general lexicon is used to build a basic morphological finite-state network. Additional networks are built using the same set of rules for the *add*, *remove* and *replace* lexicons. The final network is obtained by subtracting the *remove* network from the general one (using the set difference operator), adding the *add* network (using the set union operator), and finally applying *priority union* with the *replace* network. This final finite-state network contains only and all the valid inflected forms.

The lexicon is represented in XML, while the morphological analyzer is implemented in XFST, so the former has to be converted to the latter. In XFST, a lexical entry is a relation which holds between the surface form of the lemma and a set of lexical strings. As a surface lemma is processed by the rules, its associated lexical strings are manipulated to reflect the impact of inflectional morphology. The surface string of XFST lexical entries is the citation form specified in the XML lexicon. Figure 5

lists the XFST representation of the lexical entry of the word *bli*, whose XML representation was listed in figure 2.

```
[+negation][+id]21542[+undotted]
בלי[+translit]bli
```

Figure 5: The lexicon item of *bli* in XFST

3.2 Morphological and orthographic rules

In this section we discuss the set of rules which constitute the morphological grammar, i.e., the implementation of linguistic structures in XFST. The grammar includes hundreds of rules; we present a small sample, exemplifying the principles that govern the overall organization of the grammar. The linguistic information was collected from several sources (Barkali, 1962; Zdaqa, 1974; Alon, 1995; Cohen, 1996; Schwarzwald, 2001; Schwarzwald, 2002; Ornan, 2003).

The grammar consists of specific rules for every part of speech category, which are applied to the appropriate lexicons. For each category, a variable is defined whose denotation is the set of all lexical entries of that category. Combined with the category-specific rules, we obtain morphological grammars for every category (not including idiosyncrasies). These grammars are too verbose on the lexical side, as they contain all the information that was listed in the lexicon. Filters are therefore applied to the lexical side to remove the unneeded information.

Our rules support surface forms that are made of zero or more prefix particles, followed by a (possibly inflected) lexicon item. Figure 6 depicts the high-level organization of the grammar (recall from section 2 that '`.o.`' denotes composition). The variable `inflectedWord` denotes a union of all the possible inflections of the entire lexicon. Similarly, `prefixes` is the set of all the possible sequences of prefixes. When the two are concatenated, they yield a language of all possible surface forms, vastly over-generating. On the upper side of this language a prefix particle filter is composed, which enforces linguistically motivated constraints on the possible combinations of prefixes with words. On top of this another filter is composed, which handles "cosmetic" changes, such as removing "book-keeping" tags. A similar filter is applied to the the lower side of the network.

```
tagAffixesFilter
.o.
prefixesFilters
.o.
[ prefixes inflectedWord ]
.o.
removeTagsFilter
```

Figure 6: A high level view of the analyzer

As an example, consider the feminine singular form of adjectives, which is generated from the masculine singular by adding a suffix, either *h*, *it* or *t*. Some idiosyncratic forms have no masculine singular form, but do have a feminine singular form, for example *hrh* "pregnant". Therefore, as figure 7 shows, singular feminine adjectives are either extracted verbatim from the lexicon or generated from the singular masculine form by suffixation. The rule `[ %+feminine <- ? || %+gender _ ]` changes the gender attribute to `feminine` for the inflected feminine forms. This is a special form of a replace rule which replaces any symbol ('`?`') by the multi-character symbol '`+feminine`', in the context of occurring after '`+gender`'. The right context is empty, meaning *anything*.

```
define feminineSingularAdjective [
 [$[%+gender [%+feminine]]
  .o. adjective ] |
 [ %+feminine <- ? || %+gender _ ]
 .o. [ sufH | sufT | sufIT ]
];
```

Figure 7: Feminine adjectives

Figure 8 shows how the suffix *h* (the value of the variable `HE`) is used in the inflection. The default is not to add an additional *h* if the masculine adjective already terminates with it, as in *mwrh* "male teacher" → *mwrh* "female teacher". This means that exceptions to this default, such as *gbwh* "tall, m" → *gbwhh* "tall, f", are being improperly treated. Such forms are explicitly listed in the lexicon as idiosyncrasies (using the add/replace/remove mechanism), and will be corrected at a later stage. The suffixes *t*

and *it* are handled in a similar way.

```
define sufH [
 [ [ $[%+feminine %+h] .o.
     masculineSingularAdjective ]
   [ 0 .x. addedHE ] ]
 .o. [ addedHE -> 0 || HE _ .#. ]
 .o. [ addedHE -> HE ]
];
```

Figure 8: Adding the suffix *h*

Figure 9 shows how plural nouns with the *wt* suffix are processed. On the lower side some conditional alternations are performed before the suffix is added. The first alternation rule replaces *iih* with *ih* at the end of a word, ensuring that nouns wrttent with a spurious *i* such as *eniih* "second" are properly inflected as *eniwt* "seconds" rather than *eniiwt*. The second alternation rule removes final *t* to ensure that a singular noun such as *meait* "truck" is properly inflected to its plural form *meaiwt*. The third ensures that nouns ending in *wt* such as *smkwt* "authority" are properly inflected as *smkwiwt*. Of course, irregular nouns such as *xnit* "spear", whose plural is *xnitwt* rather than *xniwt*, are lexically specified and handled separately. Finally, a final *h* is removed by the fourth rule, and subsequently the plural suffix is concatenated.

```
define pluralWTNoun [
 [
  [ %+plural <- %+singular || %+number _ ]
  .o. $[%+number %+singular]
  .o. $[%+plural %+wt]
  .o. noun
  .o. [ YOD YOD HE -> YOD HE || _ .#. ]
  .o. [ ALEF YOD TAV -> ALEF YOD || _ .#. ]
  .o. [ VAV TAV -> VAV YOD || _ .#. ]
  .o. [ [HE|TAV] -> 0 || _ .#. ]
 ] [ 0 .x. [VAV TAV] ]
];
```

Figure 9: Plural nouns with *wt* suffix

The above rules only superficially demonstrate the capabilities of our grammar. The bulk of the grammar consists of rules for inflecting verbs, including a complete coverage of the weak paradigms. The grammar also contains rules which govern the possible combinations of prefix particles and the words they combine with.

4 Conclusion

We described a broad-coverage finite-state grammar of Modern Hebrew, consisting of two main components: a lexicon and a set of rules. The current underlying lexicon includes over 20,000 items. The average number of inflected forms for a lexicon item is 33 (not including prefix sequences). Due to the use of finite-state technology, the grammar can be used for generation or for analysis. It induces a very efficient morphological analyzer: in practice, over eighty words per second can be analyzed on a contemporary workstation.

For lack of space we cannot fully demonstrate the output of the analyzer; refer back to figure 1 for an example. HAMSAH is now used for a number of projects, including as a front end for a Hebrew to English machine translation system (Lavie et al., 2004). It is routinely tested on a variety of texts, and tokens with zero analyses are being inspected manually. A systematic evaluation of the quality of the analyzer is difficult due to the lack of available alternative resources. Nevertheless, we conducted a small-scale evaluation experiment by asking two annotators to review the output produced by the analyzer for a randomly chosen set of newspaper articles comprising of approximately 1000 word tokens. The following table summarizes the results of this experiment.

	number	%
tokens	959	100.00%
no analysis	37	3.86%
no correct analysis	41	4.28%
correct analysis produced	881	91.86%

The majority of the missing analyses are due to out-of-lexicon items, particularly proper names.

In addition to maintenance and expansion of the lexicon, we intend to extend this work in two main directions. First, we are interested in automatic methods for expanding the lexicon, especially for named entities. Second, we are currently working on a disambiguation module which will rank the analyses produced by the grammar according to context-dependent criteria. Existing works on part-of-speech tagging and morphological disambiguation in Hebrew (Segal, 1999; Adler, 2004; Bar-Haim, 2005) leave much room for further research. Incorporating state-of-the-art machine learning techniques

for morphological disambiguation to the output produced by the analyzer will generate an optimal system which is broad-coverage, effective and accurate.

Acknowledgments

This work was funded by the Israeli Ministry of Science and Technology, under the auspices of the Knowledge Center for Processing Hebrew. We are grateful to Yael Cohen-Sygal, Shira Schwartz and Alon Itai for their help.

References

Meni Adler. 2004. Word-based statistical language modeling: Two-dimensional approach. Thesis proposal, Ben Gurion University, Beer Sheva, April.

Emmanuel Alon. 1995. *Unvocalized Hebrew Writing: The Structure of Hebrew Words in Syntactic Context*. Ben-Gurion University of the Negev Press. In Hebrew.

Roy Bar-Haim. 2005. Part-of-speech tagging for Hebrew and other Semitic languages. Master's thesis, Computer Science Department, Technion, Haifa, Israel.

Shaul Barkali. 1962. *Lux HaP'alim HaShalem (The Complete Verbs Table)*. Reuven Mass, Jerusalem. In Hebrew.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite-State Morphology: Xerox Tools and Techniques*. CSLI, Stanford.

Kenneth R. Beesley. 1996. Arabic finite-state morphological analysis and generation. In *Proceedings of COLING-96, the 16th International Conference on Computational Linguistics*, Copenhagen.

Kenneth R. Beesley. 1998. 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.

Kenneth R. Beesley. 2001. Finite-state morphological analysis and generation of Arabic at Xerox Research: Status and plans in 2001. In *ACL Workshop on Arabic Language Processing: Status and Perspective*, pages 1–8, Toulouse, France, July.

Esther Bentur, Aviella Angel, Danit Segev, and Alon Lavie. 1992. Analysis and generation of the nouns inflection in Hebrew. In Uzzi Ornan, Gideon Arieli, and Edit Doron, editors, *Hebrew Computational Linguistics*, chapter 3, pages 36–38. Ministry of Science and Technology. In Hebrew.

Yaacov Choueka. 1980. Computerized full-text retrieval systems and research in the humanities: The Responsa project. *Computers and the Humanities*, 14:153–169.

Yaacov Choueka. 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*, page 63, Tel Aviv, April. In Hebrew.

Haim A. Cohen. 1996. klalei ha-ktiv xasar ha-niqqud. *leshonenu la&am*, special edition, May. In Hebrew.

Laura Kataja and Kimmo Koskenniemi. 1988. Finite-state description of Semitic morphology: A case study of Ancient Akkadian. In *COLING*, pages 313–315.

George Anton Kiraz. 2000. Multitiered nonlinear morphology using multitape finite automata: a case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105, March.

Kimmo Koskenniemi. 1983. *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production*. The Department of General Linguistics, University of Helsinki.

Alon Lavie, Alon Itai, Uzzi Ornan, and Mori Rimon. 1988. On the applicability of two-level morphology to the inflection of Hebrew verbs. In *Proceedings of the International Conference of the ALLC*, Jerusalem, Israel.

Alon Lavie, Shuly Wintner, Yaniv Eytani, Erik Peterson, and Katharina Probst. 2004. Rapid prototyping of a transfer-based Hebrew-to-English machine translation system. In *Proceedings of TMI-2004: The 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, MD, October.

Uzzi Ornan and Wadim Kazatski. 1986. Analysis and synthesis processes in Hebrew morphology. In *Proceedings of the 21 National Data Processing Conference*. In Hebrew.

Uzzi Ornan. 2003. *The Final Word*. University of Haifa Press, Haifa, Israel. In Hebrew.

Ora Schwarzwald. 2001. *Moden Hebrew*, volume 127 of *Languages of the World/Materials*. LINCOM EUROPA.

Ora Schwarzwald. 2002. *Studies in Hebrew Morphology*. The Open University of Israel.

Erel Segal. 1997. Morphological analyzer for unvocalized hebrew words. Unpublished work, available from `http://www.cs.technion.ac.il/~erelsgl/hmntx.zip`.

Erel Segal. 1999. Hebrew morphological analyzer for Hebrew undotted texts. Master's thesis, Technion, Israel Institute of Technology, Haifa, October. In Hebrew.

Shuly Wintner. 2004. Hebrew computational linguistics: Past and future. *Artificial Intelligence Review*, 21(2):113–138.

Yizxaq Zdaqa. 1974. *Luxot HaPoal (The Verb Tables)*. Kiryath Sepher, Jerusalem. In Hebrew.

Morphological Analysis and Generation for Arabic Dialects

Nizar Habash and Owen Rambow and George Kiraz
Center for Computational Learning Systems
Columbia University
New York, NY 10115, USA
{habash,rambow}@cs.columbia.edu, gkiraz@GorgiasPress.com

Abstract

We present MAGEAD, a morphological analyzer and generator for the Arabic language family. Our work is novel in that it explicitly addresses the need for processing the morphology of the dialects. MAGEAD provides an analysis to a root+pattern representation, it has separate phonological and orthographic representations, and it allows for combining morphemes from different dialects.

1 Introduction

In this paper we present initial work on MAGEAD, a morphological analyzer and generator for the Arabic language family, by which we mean both Modern Standard Arabic (MSA) and the spoken dialects.[1] There has been much work on Arabic morphology (for an overview, see (Al-Sughaiyer and Al-Kharashi, 2004)). Our work is novel in that it explicitly addresses the need for processing the morphology of the dialects. There are several important consequences:

- First, we want to be able to exploit the existing regularities among the dialects and between the dialects and MSA, in particular systematic sound changes which operate at the level of the root consonants, and pattern changes. This requires an **explicit analysis into root and pattern**.
- Second, the dialects are mainly used in spoken communication and in the rare cases when they are written they do not have standard orthographies, and different (inconsistent) orthographies may be used even within a single written text. We thus need a representation of morphology that incorporates **models of both phonology and orthography**.
- Third, in certain contexts, speakers often create words with morphemes from more than one dialect, or from a dialect and MSA. For example, the verb stem may be from MSA while the dialectal present progressive prefix is used. This means that our analyzer needs to be able to have access to **morphological data from more than one member of the language family**.

In addition, we add two general requirements for morphological analyzers. First, we want both a morphological analyzer and a morphological generator. Second, we want to use a representation that is defined in terms of a lexeme and attribute-value pairs for morphological features such as aspect or person. This is because we want our component to be usable in natural language processing (NLP) applications such as natural language generation and machine translation, and the lexeme provides a usable lexicographic abstraction.

We tackle these requirements by implementing the multitape approach of Kiraz (2000), which we

[1] We would like to thank two anonymous reviewers for helpful comments, and Amittai Aviram for his feedback and help with the implementation. The work reported in this paper was supported by NSF Award 0329163.

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 17–24,
Ann Arbor, June 2005.

extend by adding an additional tape for independently modeling phonology and orthography. This is the first large-scale implementation of (Kiraz, 2000). We use the AT&T finite-state toolkit (Mohri et al., 1998) for the implementation. The use of finite state technology makes MAGEAD usable as a generator as well as an analyzer, unlike some morphological analyzers which cannot be converted to generators in a straightforward manner (Buckwalter, 2004; Habash, 2004).

This paper is organized as follows. In Section 2, we discuss the linguistic situation of the Arabic-speaking world. In Section 3, we present the relevant facts about morphology in the Arabic language family. We then present our approach to morphological analysis in Section 4, and its implementation in Section 5. We conclude by sketching the planned evaluation.

2 The Arabic Dialects

The Arabic-speaking world is characterized by diglossia (Ferguson, 1959). Modern Standard Arabic (MSA) is the shared written language from Morocco to the Gulf, but it is not a native language of anyone. It is spoken only in formal, scripted contexts (news, speeches). In addition, there is a continuum of spoken dialects (varying geographically, but also by social class, gender, etc.) which are native languages, but rarely written (except in very informal contexts: blogs, email, etc). Dialects differ phonologically, lexically, morphologically, and syntactically from one another; many pairs of dialects are mutually unintelligible. In unscripted situations where spoken MSA would normally be required (such as talk shows on TV), speakers usually resort to repeated code-switching between their dialect and MSA, as nearly all native speakers of Arabic are unable to produce sustained spontaneous discourse in MSA.

3 Arabic Dialect Morphology

3.1 Types of Arabic Morphemes

Arabic morphemes fall into three categories: templatic morphemes, affixational morphemes, and non-templatic word stems (NTWSs). Affixational morphemes are concatenated to form words, while templatic morphemes are interleaved. Templatic morphemes come in three types that are equally needed to create a word stem: roots, patterns and vocalisms. Affixes can be classified into prefixes, suffixes and circumfixes, which precede, follow or surround the word stem, respectively. Finally NTWSs are word stems that are not constructed from a root/pattern/vocalism combination. The following three subsections discuss each of the morpheme categories. This is followed by a brief discussion of some morphological adjustment phenomena.

3.1.1 Roots, Patterns and Vocalism

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[2] كتب *katab* 'to write', كاتب *kaAtib* 'writer', and مكتوب *maktuwb* 'written' all share the root morpheme *ktb* (ك ت ب) 'writing-related'.

The pattern morpheme is an abstract template in which roots and vocalisms are inserted. We will represent the pattern as a string of letters including special symbols to mark where root radicals and vocalisms are inserted. We use numbers (i.e. 1, 2, 3, 4, or 5) to indicate radical position[3] and the symbol *V* is used to indicate the position of the vocalism. For example, the pattern *1V22V3* indicates that the second root radical is to be doubled. A pattern can include letters for additional consonants and vowels, e.g., the verbal pattern *V1tV2V3*.

The vocalism morpheme specifies which short vowels to use with a pattern.[4] A word stem is constructed by interleaving the three types of templatic morphemes. For example, the word stem كتب *katab* 'to write' is constructed from the root *ktb* (ك ت ب), the pattern *1V2V3* and the vocalism *aa*.

[2] In this paper, we use the following conventions for representing examples. All orthographic word forms are provided in undiacritized Arabic script followed by a diacritized version in the Buckwalter transliteration scheme, which is a 1-to-1 transliteration of MSA orthographic symbols using ASCII characters (Buckwalter, 2004). All morphemes are shown diacritized in the Buckwalter transliteration of a plausible standard orthographic representation, though we sometimes include an undiacritized version in Arabic script in parentheses for clarity. All phonemic sequences are written between the usual slashes, but we use the Buckwalter scheme (with obvious adjustments) rather than IPA to represent phonemes.

[3] Often in the literature, radical position is indicated with *C*.

[4] Traditional accounts of Arabic morphology collapse vocalism and pattern.

3.1.2 Affixational Morphemes

Arabic affixes can be prefixes such as *sa+* (+سـ) 'will/[future]', suffixes such as *+uwna* (+ون) '[masculine plural]' or circumfixes such as *ta++na* (تـ++ن) '[subject 2nd person feminine plural]'. Multiple affixes can appear in a word. For example, the word وسيكتبونها *wasayaktubuwnahA* 'and they will write it' has two prefixes, one circumfix and one suffixes:[5]

(1) wasayaktubuwnahA
wa+ sa+ y+ aktub +uwna +hA
and will 3person write masculine-plural it

Some of the affixes can be thought of as orthographic clitics, such as *w+* (+و) 'and' prepositions (*l+* (+ل) 'to/for', *b+* (+بـ) 'in/with' and *k+* (+كـ) 'as') or the pronominal object clitics (e.g., *+hA* (+ها) in the example above). Others are bound morphemes.

3.1.3 Non-Templatic Word Stem

NTWS are word stems that are not derivable from templatic morphemes. They tend to be foreign names and borrowed terms. For example, واشنطن *waA$inTun* 'Washington'. Word stems can still take affixational morphemes, e.g., والواشنطنيون *waAl-waA$inTuniy˜uwn* 'and the Washingtonians'.

3.1.4 Morphological Rewrite Rules

An Arabic word is constructed by first creating a word stem from templatic morphemes or by using a NTWS. Affixational morphemes are then added to this stem. The process of combining morphemes involves a number of phonological, morphemic and orthographic rules that modify the form of the created word so it is not a simple interleaving or concatenation of its morphemic components.

An example of a phonological rewrite rule is the voicing of the /t/ of the verbal pattern *V1tV2V3* (Form VIII) when the first root radical is /z/, /d/, or /*/ (ز, د, or ذ): the verbal stem *zhr+V1tV2V3+iaa* is realized phonologically as /izdahar/ (orthographically: ازدهر) 'flourish' not /iztahar/ (orthographically: ازتهر). An example of a morphemic rewrite rule is the feminine morpheme, *+p* (+ة). Phonologically, it is realized as /t/ word-internally, but it is silent at the end of a word. Orthographically, it is realized as ت *t* in word-internal position (i.e., when followed by a letter), but as ة+ *+p* word-finally. For example, *>amiyrap+nA* (أميرة+نا) is realized as أميرتنا *>amiyratnA* 'our princess' (phonologically: /'amiyratnA/)[6]. Finally, an example of an orthographic rewrite rule is the deletion of the Alif (ا) of the definite article morpheme *Al+* (+ال) in nouns when preceded by the preposition *l+* (+ل) (in both of the following examples, the Alif is silent):

(2) a. للبيت *lilbayti* /lilbayti/ 'to the house'
li+ Al+ bayt +i
to+ the+ house +[genitive]

b. بالبيت *biAlbayti* /bilbayti/ 'in the house'
bi+ Al+ bayt +i
in+ the+ house +[genitive]

3.2 Morpheme Type and Function and the Lexeme

The type of morpheme is independent of the morphological function it is used for (derivational or inflectional). Although affixational morphemes tend to be inflectional and templatic morphemes derivational, there are many exceptions. For example, the plural of كتاب *kitAb* 'book' is not formed through affixation of the inflectional plural morphemes *+At* (+ات) or *+uwn* (+ون), but rather through the use of a different pattern, resulting in كتب *kutub* 'books'. This form of plural construction is called "broken plural" in Arabic to distinguish it from the strictly affixational "sound plural". Conversely, the adjective كتبي *kutubiy˜* 'book-related' is derived from the noun كتب *kutub* 'books' using affixational morphemes. Note that approaches for Arabic stemming that are limited to handling affixational morphology will both miss related terms that are inflected templatically and conflate derived forms generated affixationally.

A common misconception about Arabic morphology concerns the regularity of derivational morphology. However, the meaning of a word cannot be predicted from the root and the pattern+vocalism pair. For example, the masculine noun مكتب *maktab* 'office/bureau/agency' and the feminine noun

[5] We analyze the imperfective word stem as including an initial short vowel, and leave a discussion of this analysis to future publications.

[6] The case markers are ignored in this example for the sake of simplicity.

مكتبة *maktabap* ‘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 gender can account for the semantic difference. It is this unpredictability of derivational meaning that makes us prefer lexemes as deepest units of morphological analysis, rather than root+pattern pairs. We use the root+pattern analysis only to relate different dialects, and since it has proven useful for certain natural language processing tasks, such as IR (Abu-Salem et al., 1999). We use the lexemic representation to represent the lexicon for applications such as machine translation, including translation between dialects. We return to the definition of “lexeme” in Section 4.2.

3.3 Dialect Morphology

Arabic dialect morphology shares with MSA morphology the root-and-pattern system. Additionally, each dialect morphology shares with MSA morphology some of the morphology lexicon (inventory of morphemes), and the morphological rules. Consider the following forms by way of example:

(3) Egyptian: مبنئلهلكش mabin}ulhalak$ =
ma+ b+ n+ [’wl + V12V3 + iu] +ha +lak +$
MSA: لا نقولها لك lA naquwluha laka =
lA / n+ [qwl + V12V3 + au] +u +ha / la +ka

Here, the Egyptian stem is formed from the same pattern as the MSA stem, but the initial radical, *q* in MSA, has become ’ in Egyptian through regular sound change. The vocalism in Egyptian also differs from that in MSA. Then, we add the first person plural subject agreement marker, the prefix *n+* (which in MSA is the circumfix *n++u*) and the third person feminine singular object clitic *+ha* (same in MSA). In Egyptian, we add a second person masculine singular indirect object clitic *+lak*, the present progressive prefix *b+*, and the negation circumfix *ma++$*. None of these exist in MSA: their meaning is represented with separate words, or as a zero morpheme in the case of the present tense marker. Note that Egyptian orthography is not standardized, so that the form above could be plausibly written in any of the following orthographies, among others: مابنؤلهلكش mAbin&ulhalak$, ما بنئلهلكش mA bin}ulhAlak$, ما بنقلها لكش mA binqulhalak$, مبنقلهلكش mabinqulhalak$, ما بنقولها لكش mA binquwlhA lak$.

Within a word form, all morphemes need not be from the same dialect. Consider the following example.[7] The speaker, who is a journalist conducting an interview, switches from MSA to Egyptian (between square brackets) for a complementizer (اللي *Ailliy*) that introduces a relative clause. He then continues in Egyptian with the prefix *b+* (+بـ) ‘[present progressive]’, and then, inside the word, returns to MSA, using an MSA verb in which the passive voice is formed with MSA morphology, *-tuwaj˜ah* (-توجه) ‘be directed’.

(4) هل كانت إسرائيل هي الأولى [اللي بـ+] -توجه لها القوات المصرية أو كانت توجه ضد قواتعربية أخرى؟

hal kaAnat <isra}iyl AilmafruwD hiya Aal>uwlaY [Ailliy bi+] tuwaj˜ah laha Ailquw˜aAt AilmaSriy˜ap >aw kaAnat tuwaj˜ah Did quw˜aAt Earabiy˜ap >uxraY?
Should it have been Israel first [that] Egyptian armies were directed towards, or were they to be directed against other Arab armies?

4 Morphological Analysis of Arabic

4.1 Previous Work

Despite the complexity of Semitic root-and-pattern morphology, computational morphologists have taken up the challenge of devising tractable systems for computing it both under finite-state methods and non-finite-state methods. Kataja and Koskenniemi (1988) presented a system for handling Akkadian root-and-pattern morphology by adding a additional lexicon component to Koskenniemi’s two-level morphology (1983). The first large scale implementation of Arabic morphology within the constraints of finite-state methods was that of Beesley et al. (1989) with a ‘detouring’ mechanism for access to multiple lexica, which later gave rise to other works by Beesley (Beesley, 1998) and, independently, by Buckwalter (2004).

The now ubiquitous linguistic approach of McCarthy (1981) to describe root-and-pattern morphol-

[7] This example is a transcript of a broadcast originally taken from the Al-Jazeera web site. It can now be found at http://web.archive.org/web/20030210100557/www.aljazeera.net/programs/century_witness/articles/2003/1/1-24-1.htm .

ogy under the framework of autosegmental phonology gave rise to a number of computational proposals. Kay (1987) devised a framework with which each of the autosegmental tiers is assigned a tape in a multi-tape finite state machine, with an additional tape for the surface form. Kiraz (2000,2001) extended Kay's approach and implemented a working multi-tape system with pilot grammars for Arabic and Syriac. Other autosegmental approaches (described in more details in Kiraz 2001 (Chapter 4)) include those of Kornai (1995), Bird and Ellison (1994), Pulman and Hepple (1993), whose formalism Kiraz adopted, and others. In this work we follow the multi-tape approach, and specifically that of (Kiraz, 2000). This is the first large-scale implementation of that approach.

4.2 Our Approach: Outline

In our approach, there are three levels of representation:

Lexeme Level. Words are represented in terms of a lexeme and features. Example:

(5) Aizdaharat: Aizdahar$_1$ POS:V PER:3 GEN:F NUM:SG ASPECT:PERF

The list of features is dialect-independent. The lexeme itself can be thought of as a triple consisting of a root (or an NTWS), a meaning index, and a morphological behavior class (MBC). The MBC maps the features to morphemes. For example, [+FEM] for كاتب *kaAtib* 'writer$_{\text{MASC}}$ ' yields كاتبة *kaAtibap* 'writer$_{\text{FEM}}$ ' which is different from [+FEM] for ابيض *AabyaD* 'white$_{\text{MASC}}$ ' which yields بيضاء *bayDaA'* 'white$_{\text{FEM}}$ '. The MBCs are of course specific to the dialect in question or MSA (though conceivably some can be shared between dialects). For convenience (as in the example above), lexemes are often represented using a citation form.

Morpheme Level. Words are represented in terms of morphemes. (5) is now represented as follows:

(6) Aizdaharat: [zhr + V1tV2V3 + iaa] + at

Surface Level. Words are a string of characters. Using standard MSA orthography, our example becomes:

(7) ازدهرت Aizdaharat

Phonologically, we get:

(8) /izdaharat/

This paper focuses on the morpheme layer (morphology) and the transition between the morpheme and the surface levels. This transition draws on the following resources:

- a unified context-free grammar for morphemes (for all dialects together) which specifies the ordering of affixival morphemes.
- Morphophonemic and phonological rules that map from the morphemic representation to the phonological representation.
- Orthographic rules that map from phonology and morphology to an orthographic representation.

We will next discuss the formal representational and computational framework for these resources.

4.3 Multitape Automata

We follow (Kiraz, 2000) in using a multitape analysis. We extend that analysis by introducing a fifth tier. The five tiers are used as follows:

- Tier 1: pattern and affixival morphemes.
- Tier 2: root.
- Tier 3: vocalism.
- Tier 4: phonological representation.
- Tier 5: orthographic representation.

Tiers 1 through 3 are always input tiers. Tier 4 is first an output tier, and subsequently an input tier. Tier 5 is always an output tier. All tiers are read or written at the same time, so that the rules of the multi-tier automaton are rules which scan the input tiers and, depending on the state, write to the output tier. The introduction of two surface-like tiers is due to the fact that many dialects do not have a standard orthography, as discussed above in Section 3.3.

5 Implementing Multitape Automata

We have implemented multi-tape finite state automata as a layer on top of the AT&T two-tape finite state transducers. Conversion from this higher layer (the new **Morphtools format**) to the Lextools format (an NLP-oriented extension of the AT&T toolkit

for finite-state machines, (Sproat, 1995)) is done for different types of Lextools files such as rule files or context-free grammar files. A central concept here is that of the **multitape string** (MTS), a special representation of multiple tiers in Morphtools that gets converted to a sequence of **multi-tier tokens** (MTT) compatible with Lextools. In the next section, we discuss the conversion of MTS into MTT. Then, we discuss an example rule conversion.

5.1 The Multitape String

A multitape string (MTS) is represented as `<T,R,V,P,O>`. where:

- *T* is the template or basic pattern. The template is represented as a string indicating the position of root consonant (1,2,3,4,5 or C), vowel (V), and any consonant or vowel deemed to be part of the template but not a separate morpheme. For example, Arabic verb form II pattern is represented as `1V22V3` and form VIII is represented as `V1tV2V3`.
- *R* is the root radicals (consonants).
- *V* is the vocalism vowels.
- *P* is the phonological level.
- *O* is the orthographic level.

There are two special symbols: (1) `%` is a wild card symbol that can match anything (appropriate for that tier) and (2) `@<Letter>` (e.g., `@X`) is a variable whose type can be defined explicitly. Both symbols can appear in any tier (except that in our current implementation, `%` cannot appear in tier *T*).

The first (or template) tier (*T*) is always required. The additional tiers can be left underspecified. For example, the full MTS specification for the root `zhr` with form VIII with active vocalism is:

(9) `<V1tV2V3,zhr,iaa>`

When converting an MTS to Lextools format, the *T* tier is used to create a basic default sequence of multi tier tokens (MTTs). For our example (9), `V1tV2V3` leads to this initial MTT sequence:

(10) `[V0%00] [1%000] [t0000] [V0%00] [2%000] [V0%00] [3%000]`

When the symbol *V* appears in the template, a `0` is inserted in the radical position (since no radical can be inserted here) and a wild card is inserted in the vocalism position. The opposite is true for when radical symbol (*C,1,2,3,4,5*) appears in the template, a `0` is inserted in the vocalism tier (as no vowel from the vocalism can be inserted here) and a wild card in the radical tier. all other characters appearing in the template tier (e.g., `t` in the example above), are paired with `0`s in all other tiers.

Additional information from other tiers are then written on top of the default MTT sequence created from the template tier. The representation in (10) is transformed into (12), using the information from the root and vocalism tiers in (9):

(11) `[V0i00] [1z000] [t0000] [V0a00] [2h000] [V0a00] [3r000]`

This sequence corresponds to the form /iztahar/. After applying phonological rules, which will be discussed in the next section, the MTT sequence is as follows. Note that the fourth tier has been filled in.

(12) `[V0ii0] [1z0z0] [t00d0] [V0aa0] [2h0h0] [V0aa0] [3r0r0]`

In this fourth tier, this represents the phonological form /izdahar/. Applying orthographic rules for diacritized orthography, we write symbols into the fifth tier, which corresponds to the orthographic form ازدهر *Aizdahar*.

(13) `[0000A] [V0iii] [1z0zz] [t00dd] [V0aaa] [2h0hh] [V0aaa] [3r0rr]`

Note that the fourth tier provides the (phonemic) pronunciation for the orthography in the fifth tier.

5.2 Representing the Structure of the Word

The basic structure of the word is represented using a context-free grammar (CFG). The CFG covers all dialects and MSA, and only when they differ in terms of the morpheme sequencing does the CFG express dialect-specific rules. How exactly to write this CFG is an empirical question: for example, if frequently speakers mix MSA verb stems with ECA subject agreement suffixes, then the following grammar fragment would not be sufficient. We intend to develop probabilistic models of intra-word code switching in order to guide the morphological analysis in the presence of code switching.

The following rule is the top-level rule which

states that a word is a verb, a noun, or a particle, and it can be preceded by an optional conjunction (for example, *w+*). It holds in all dialects and MSA.

(14) `[WORD] -> [CONJ]? ([VERB]|[NOUN]|[PART])`

The following rule expands verbs to three inflectional types and adds an optional object clitic. For Egyptian (ECA) only, an indirect object clitic can also be added.

(15) `[VERB] -> ([PV_VERB]|[IV_VERB]) [OBJ_PRON]? [ECA:IOBJ_PRON]?`

The next level of expansion then introduces specific morphemes for the two classes of perfective verbs and imperfective verbs. Here, we split into separate forms for each dialect and MSA; we give examples for MSA and Egyptian.

(16) a. `[PV_VERB] -> [MSA:PV_VERB_STEM] [MSA:SUF:PVSUBJ_1S]`
b. `[PV_VERB] -> [ECA:PV_VERB_STEM] [ECA:SUF:PVSUBJ_1S]`

This list is continued (for all dialects and MSA) for all combinations of person, number, and gender. In the case of the imperfective, we get additional prefixes, and circumfixes for the subject clitics. Note that here we allow a combination of the MSA imperfective verb stem with the Egyptian prefixes, but we do not allow the MSA prefixes with the Egyptian verb stem.

(17) a. `[IV_VERB] -> ([MSA:FUT]| [MSA:RESULT]| [MSA:SUBJUNC]| [MSA:EMPHATIC]| [ECA:PRESENT] | [ECA:FUT])? [MSA:IV_VERB_CONJUG]`
b. `[IV_VERB] -> ([ECA:FUT]| [ECA:PRESENT])? [ECA:IV_VERB_CONJUG]`

We then give the verbal stem morphology for MSA (the Egyptian case is similar).

(18) `[MSA:IV_VERB_CONJUG] -> [MSA:PRE:IVSUBJ_1S] [MSA:IV_VERB_STEM] [MSA:SUF:IVSUBJ_1S]`

Again, this list is continued for all valid combinations of person, number, and gender. The verbal stems are expanded to possible forms (combination of pattern and vocalism, not specified for root), or NTWSs. Since the forms are specific to perfective or imperfective aspect, they are listed separately.

(19) `[MSA:PV_VERB_STEM] -> ([MSA:FORM_I_PV]| [MSA:FORM_II_PV]| [MSA:FORM_III_PV]| [MSA:FORM_IV_PV]| ...)`

Each form is expanded separately:

(20) a. `[MSA:FORM_I_PV] -> (<1V2V3,%,aa>| <1V2V3,%,ai>| <1V2V3,%,au>)`
b. `[MSA:FORM_II_PV] -> <1V22V3,%,aa>`

Separate rules introduce the morphemes which are represented by nonterminals such as `[MSA:PRE:IVSUBJ_1S]` or `[ECA:PRESENT]`. Such a context-free specification using MTS is then compiled into MTT sequences in the same manner as described above. The resulting specification is a valid input to Lextools, which generates the finite state machines.

5.3 Representing Rules

We now discuss the representation of rules. We start out with three default rules which are the same for all Arabic dialects and MSA (and possibly for all languages that use templatic morphology). Rule (21a) writes a letter which is in the pattern tier but which is not specified as either root or vocalism to the fourth (phonological) tier, while Rule (21b) and (21c) write a radical and a pattern vowel, respectively.

(21) a. `<@X,,,0> -> @X, @X=[LETTER]`
b. `<C,@X,,0> -> @X`
c. `<V,,@X,0> -> @X`

Phonological and morphemic rules have the same format, as they write to the fourth tier, usually overwriting a symbol placed there by the default rules. Rule (22) implements the rule mentioned in Section 3.1.4 (in Form VIII, the /t/ of the pattern changes to a /d/ if the first radical is /z/, /d/, or /*/). Rule (22) accounts for the surface phonological form in (8); without Rule (22), we would have *iztahar* instead of *izdahar*.

(22) `<t,,,t> -> d / <1,@M,,> _ , @M=[zd*]`

For the orthography we use the fifth tier. As in the case of phonology, we have default rules, which yield a simple phonemic orthography.

(23) a. `<@Y,,,@X,0> -> @X, @Y=[LETTER], @X=[LETTER]`
b. `<V,,@V,@X,0> -> @X, @X=[LETTER]`
c. `<C,@C,,@X,0> -> @X, @X=[LETTER]`
d. `<+,,,+,+> -> 0`

These default rules cover much of MSA orthography, but in addition, there are some special orthographic rules, for example:

(24) `<0V,,@X,@X,0> -> A@X, # _, @X=[LETTER]`

This rule inserts an Alif at the beginning of a word which starts with a pattern vowel.

6 Outlook

This paper describes work in progress. We are currently in the process of populating MAGEAD with morphological data and rules for MSA and Egyptian, with smaller efforts for Yemeni and Levantine. We intend to evaluate MAGEAD using a double strategy: a test suite of selected surface word/analysis pairs which tests the breadth of phenomena covered, and a test corpus, which tests the adequacy on real text. The test suite can be assembled by hand over time from individual examples and is used for regression testing during development, as well as for qualitative assessment of the analyzer or generator. The only test corpus we currently have is the Penn Arabic Treebank for MSA.

In the next phase of the development work, we will link the list of morphemes obtained during analysis to the lexeme level of representation. This will be done using a dialect-specific lexicon, but we will also develop tools to exploit the lexical similarity between the dialects and MSA (and among the dialects) by hypothesizing lexemes based on regular sound change rules.

References

Hani Abu-Salem, Mahmoud Al-Omari, and Martha W. Evens. 1999. Stemming methodologies over individual query words for an arabic information retrieval system. *J. Am. Soc. Inf. Sci.*, 50(6):524–529.

Imad A. Al-Sughaiyer and Ibrahim A. Al-Kharashi. 2004. Arabic morphological analysis techniques: A comprehensive survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213.

K. Beesley, T. Buckwalter, and S. Newton. 1989. Two-level finite-state analysis of Arabic morphology. In *Proceedings of the Seminar on Bilingual Computing in Arabic and English*, page n.p.

K. Beesley. 1998. Arabic morphology using only finite-state operations. In M. Rosner, editor, *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 50–7, Montereal.

S. Bird and T. Ellison. 1994. One-level phonology. *Computational Linguistics*, 20(1):55–90.

Tim Buckwalter. 2004. Buckwalter Arabic morphological analyzer version 2.0.

Charles F Ferguson. 1959. Diglossia. *Word*, 15(2):325–340.

Nizar Habash. 2004. Large scale lexeme based arabic morphological generation. In *Proceedings of Traitement Automatique du Langage Naturel (TALN-04)*. Fez, Morocco.

L. Kataja and K. Koskenniemi. 1988. Finite state description of Semitic morphology. In *COLING-88: Papers Presented to the 12th International Conference on Computational Linguistics*, volume 1, pages 313–15.

M. Kay. 1987. Nonconcatenative finite-state morphology. In *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics*, pages 2–10.

George Anton Kiraz. 2000. Multi-tiered nonlinear morphology using multi-tape finite automata: A case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105.

George Kiraz. 2001. *Computational Nonlinear Morphology: With Emphasis on Semitic Languages*. Cambridge University Press.

A. Kornai. 1995. *Formal Phonology*. Garland Publishing.

K. Koskenniemi. 1983. *Two-Level Morphology*. Ph.D. thesis, University of Helsinki.

J. McCarthy. 1981. A prosodic theory of nonconcatenative morphology. *Linguistic Inquiry*, 12(3):373–418.

M. Mohri, F. Pereira, and M. Riley. 1998. A rational design for a weighted finite-state transducer library. In D. Wood and S. Yu, editors, *Automata Implementation*, Lecture Notes in Computer Science 1436, pages 144–58. Springer.

S. Pulman and M. Hepple. 1993. A feature-based formalism for two-level phonology: a description and implementation. *Computer Speech and Language*, 7:333–58.

R. Sproat. 1995. Lextools: Tools for finite-state linguistic analysis. Technical Report 11522-951108-10TM, Bell Laboratories.

Examining the Effect of Improved Context Sensitive Morphology on Arabic Information Retrieval

Kareem Darwish
Dept. of Information Engineering & Technology
German University in Cairo
5th District, New Cairo, Cairo, Egypt
and
IBM Technology Development Center
P.O. Box 166, El-Ahram, Giza, Egypt
kareem@darwish.org

Hany Hassan and Ossama Emam
IBM Technology Development Center
P.O. Box 166
El-Ahram, Giza, Egypt
{hanyh,emam}@eg.ibm.com

Abstract

This paper explores the effect of improved morphological analysis, particularly context sensitive morphology, on monolingual Arabic Information Retrieval (IR). It also compares the effect of context sensitive morphology to non-context sensitive morphology. The results show that better coverage and improved correctness have a dramatic effect on IR effectiveness and that context sensitive morphology further improves retrieval effectiveness, but the improvement is not statistically significant. Furthermore, the improvement obtained by the use of context sensitive morphology over the use of light stemming was not significantly significant.

1 Introduction

Due to the morphological complexity of the Arabic language, much research has focused on the effect of morphology on Arabic Information Retrieval (IR). The goal of morphology in IR is to conflate words of similar or related meanings. Several early studies suggested that indexing Arabic text using roots significantly increases retrieval effectiveness over the use of words or stems [1, 3, 11]. However, all 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 [4, 12, 13], 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 the number of words that the analyzer can analyze, to the cases programmed into the transducers. Darwish attempted to solve this problem by developing a statistical morphological analyzer for Arabic called Sebawai that attempts to rank possible analyses to pick the most likely one [7]. He concluded that even with ranked analysis, morphological analysis did not yield statistically significant improvement over words in IR. A later study by Aljlayl et al. on a large Arabic collection of 383,872 documents suggested that lightly stemmed words, where only common prefixes and suffixes are stripped from them, were perhaps better index term for Arabic [2]. Similar studies by Darwish [8] and Larkey [14] also suggested that light stemming is indeed superior to morphological analysis in the context of IR.

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 25–30,
Ann Arbor, June 2005.

However, the shortcomings of morphology might be attributed to issues of coverage and correctness. Concerning coverage, analyzers typically fail to analyze Arabized or transliterated words, which may have prefixes and suffixes attached to them and are typically valuable in IR. As for correctness, the presence (or absence) of a prefix or suffix may significantly alter the analysis of a word. For example, for the word "Alksyr" is unambiguously analyzed to the root "ksr" and stem "ksyr." However, removing the prefix "Al" introduces an additional analysis, namely to the root "syr" and the stem "syr." Perhaps such ambiguity can be reduced by using the context in which the word is mentioned. For example, for the word "ksyr" in the sentence "sAr ksyr" (and he walked like), the letter "k" is likely to be a prefix.

The problem of coverage is practically eliminated by light stemming. However, light stemming yields greater consistency without regard to correctness. Although consistency is more important for IR applications than linguistic correctness, perhaps improved correctness would naturally yield great consistency. Lee et al. [15] adopted a trigram language model (LM) trained on a portion of the manually segmented LDC Arabic Treebank in developing an Arabic morphology system, which attempts to improve the coverage and linguistic correctness over existing statistical analyzers such as Sebawai [15]. The analyzer of Lee et al. will be henceforth referred to as the IBM-LM analyzer. IBM-LM's analyzer combined the trigram LM (to analyze a word within its context in the sentence) with a prefix-suffix filter (to eliminate illegal prefix suffix combinations, hence improving correctness) and unsupervised stem acquisition (to improve coverage). Lee et al. report a 2.9% error rate in analysis compared to 7.3% error reported by Darwish for Sebawai [7]. This paper evaluates the IBM-LM analyzer in the context of a monolingual Arabic IR application to determine if in-context morphology leads to improved retrieval effectiveness compared to out-of-context analysis. To determine the effect of improved analysis, particularly the use of in-context morphology, the analyzer is used to produce analyses of words in isolation (with no context) and in-context. Since IBM-LM only produces stems, Sebawai was used to produce the roots corresponding to the stems produced by IBM-LM. Both are compared to Sebawai and light stemming.

The paper will be organized as follows: Section 2 surveys related work; Section 3 describes the IR experimental setup for testing the IBM-LM analyzer; Section 4 presents experimental results; and Section 5 concludes the paper.

2 Related Work

Most early studies of character-coded Arabic text retrieval relied on relatively small test collections [1, 3, 9, 11]. The early studies suggested that roots, followed by stems, were the best index terms for Arabic text. More recent studies are based on a single large collection (from TREC-2001/2002) [9, 10]. The studies examined indexing using words, word clusters [14], terms obtained through morphological analysis (e.g., stems and roots [9]), light stemming [2, 8, 14], and character n-grams of various lengths [9, 16]. The effects of normalizing alternative characters, removal of diacritics and stop-word removal have also been explored [6, 19]. These studies suggest that perhaps light stemming and character n-grams are the better index terms.

Concerning morphology, some attempts were made to use statistics in conjunction with rule-based morphology to pick the most likely analysis for a particular word or context. In most of these approaches an Arabic word is assumed to be of the form prefix-stem-suffix and the stem part may or may not be derived from a linguistic root. Since Arabic morphology is ambiguous, possible segmentations (i.e. possible prefix-stem-suffix tuples) are generated and ranked based on the probability of occurrence of prefixes, suffixes, stems, and stem template. Such systems that use this methodology include RDI's MORPHO3 [5] and Sebawai [7]. The number of manually crafted rules differs from system to system. Further MORPHO3 uses a word trigram model to improve in-context morphology, but uses an extensive set of manually crafted rules. The IBM-LM analyzer uses a trigram language model with a minimal set of manually crafted rules [15]. Like other statistical morphology systems, the IBM-LM analyzer assumes that a word is constructed as prefix-stem-suffix. Given a word, the analyzer generates all possible segmentations by identifying all matching prefixes and suffixes from a table of

prefixes and suffixes. Then given the possible segmentations, the trigram language model score is computed and the most likely segmentation is chosen. The analyzer was trained on a manually segmented Arabic corpus from LDC.

3 Experimental Design

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 [17]. 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 [10]. Queries were formed from the TREC topics by combining the title and description fields. 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 with a search.

Experiments were performed for the queries with the following index terms:

- w: words.
- ls: lightly stemmed words, obtained using Al-Stem [17][1].
- SEB-s: stems obtained using Sebawai.
- SEB-r: roots obtained using Sebawai.
- cIBM-LMS: stems obtained using the IBM-LM analyzer in context. Basically, the entire TREC collection was processed by the analyzer and the prefixes and suffixes in the segmented output were removed.
- cIBM-SEB-r: roots obtained by analyzing the in-context stems produced by IBM-LM using Sebawai.
- IBM-LMS: stems obtained using the IBM-LM analyzer without any contextual information. Basically, all the unique words in the collection were analyzed one by one and the prefixes and suffixes in the segmented output were removed.
- IBM-SEB-r: roots obtained by analyzing the out-of-context stems produced by IBM-LM using Sebawai.

All retrieval experiments were performed using the Lemur language modeling toolkit, which was configured to use Okapi BM-25 term weighting with default parameters and with and without blind relevance feedback (the top 20 terms from the top 5 retrieved documents were used for blind relevance feedback). 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 0.05 to claim significance.

4 Results and Discussion

Figure 1 shows a summary of the results for different index terms. Tables 1 and 2 show statistical significance between different index terms using the p value of the Wilcoxon test. When comparing index terms obtained using IBM-LM and Sebawai, the results clearly show that using better morphological analysis produces better retrieval effectiveness. The dramatic difference in retrieval effectiveness between Sebawai and IBM-LM highlight the effect of errors in morphology that lead to inconsistency in analysis. When using contextual information in analysis (compared to analyzing words in isolation – out of context) resulted in only a 3% increase in mean average precision when using stems (IBM-LMS), which is a small difference compared to the

[1] A slightly modified version of Leah Larkey's Light-10 light stemmer [8] was also tried, but the stemmer produced very similar results to Al-Stem.

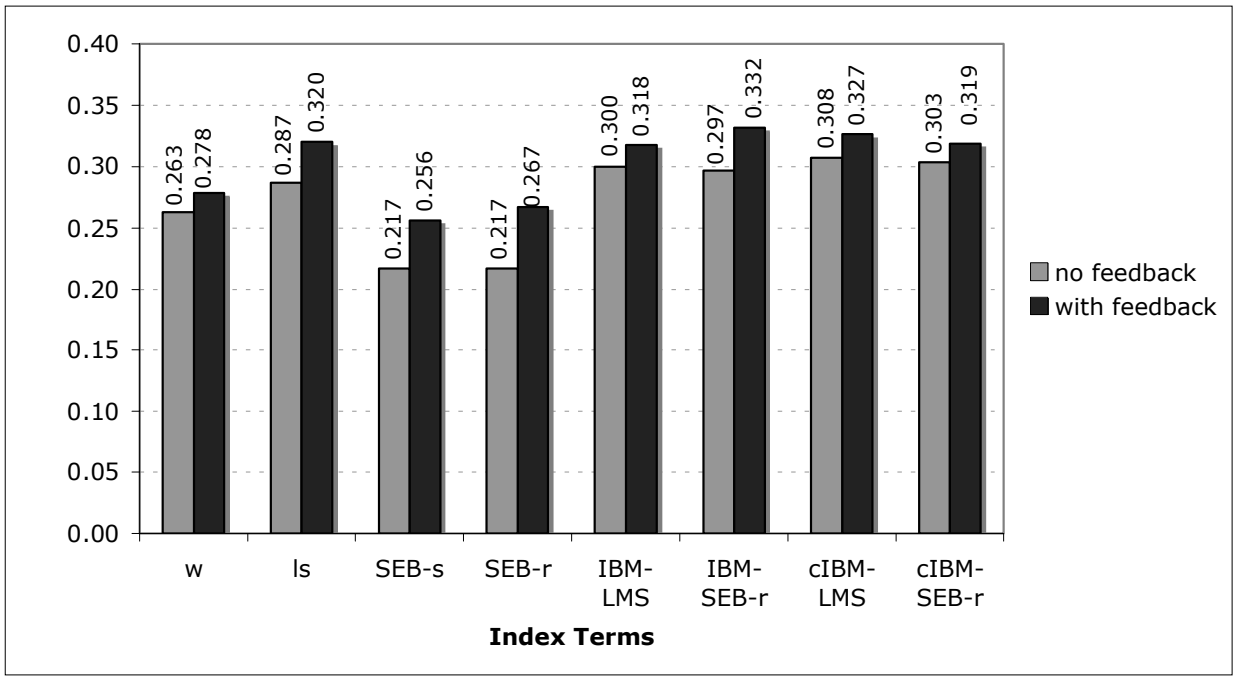

Figure 1. Comparing index term with and without blind relevance feedback using mean average precision

effect of blind relevance feedback (about 6% increase) and produced mixed results when using roots (IBM-SEB-r). Nonetheless, the improvement for stems was almost statistically significant with *p* values of 0.063 and 0.054 for the cases with and without blind relevance feedback. Also considering that improvement in retrieval effectiveness resulted from changing the analysis for only 0.12% of the words in the collection (from analyzing them out of context to analyzing them in context)[2] and that the authors of IBM-LM report about 2.9% error rate in morphology, perhaps further improvement in morphology may lead to further improvement in retrieval effectiveness. However, further improvements in morphology and retrieval effectiveness are likely to be difficult. One of difficulties associated with developing better morphology is the disagreement on what constitutes "better" morphology. For example, should "mktb" and "ktb" be conflated? "mktb" translates to office, while ktb translates to books. Both words share the common root "ktb," but they are not interchangeable in meaning or usage. One would expect that increasing conflation would improve recall at the expense of precision and decreasing conflation would have the exact opposite effect. It is known that IR is more tolerant of over-conflation than under-conflation [18]. This fact is apparent in the results when comparing roots and stems. Even though roots result in greater conflation than stems, the results for stems and roots are almost the same. Another property of IR is that IR is sensitive to consistency of analysis. In the case of light stemming, stemming often mistakenly removes prefixes and suffixes leading to over conflation, for which IR is tolerant, but the mistakes are done in a consistent manner. It is noteworthy that sense disambiguation has been reported to decrease retrieval effectiveness [18]. However, since improving the correctness of morphological analysis using contextual information is akin to sense disambiguation, the fact that retrieval results improved, though slightly, using context sensitive morphology is a significant result.

In comparing the IBM-LM analyzer (in context or out of context) to light stemming (using Al-Stem), although the difference in retrieval effectiveness is small and not statistically significant, using the IBM-LM analyzer, unlike using Al-Stem, leads to

[2] Approximately 7% of unique tokens had two or more different analysis in the collection when doing in-context morphology. In tokens with more than one analysis, one of the analyses was typically used more than 98% of the time.

ls	SEB-s	SEB-r	IBM-LMS	IBM-SEB-r	cIBM-LMS	cIBM-SEB-r	
0.055	0.475	0.671	0.038	0.027	0.019	0.049	w
	0.004	0.023	0.560	0.359	0.946	0.505	ls
		0.633	0.005	0.001	0.001	0.012	SEB-s
			0.039	0.007	0.020	0.064	SEB-r
				0.0968	0.063	0.758	IBM-LMS
					0.396	0.090	IBM-SEB-r
						0.001	cIBM-LMS

Table 1. Wilcoxon *p* values (shaded=significant), with blind relevance feedback.

ls	SEB-s	SEB-r	IBM-LMS	IBM-SEB-r	cIBM-LMS	cIBM-SEB-r	
0.261	0.035	0.065	0.047	0.135	0.011	0.016	w
	0.000	0.000	0.968	0.757	0.515	0.728	ls
		0.269	0.000	0.000	0.000	0.000	SEB-s
			0.000	0.000	0.000	0.000	SEB-r
				0.732	0.054	0.584	IBM-LMS
					0.284	0.512	IBM-SEB-r
						0.005	cIBM-LMS

Table 2. Wilcoxon *p* values (shaded=significant), without blind relevanc e feedback

statistically significant improvement over using words. Therefore there is some advantage, though only a small one, to using statistical analysis over using light stemming. The major drawback to morphological analysis (specially in-context analysis) is that it requires considerably more computing time than light stemming[3].

5 Conclusion

The paper investigated the effect of improved morphological analysis, especially context sensitive morphology, in Arabic IR applications compared to other statistical morphological analyzers and light stemming. The results show that improving morphology has a dramatic effect on IR effectiveness and that context sensitive morphology slightly improved Arabic IR over non-context sensitive morphology, increasing IR effectiveness by approximately 3%. The improvement is almost statistically significant. Developing better morphology could lead to greater retrieval effectiveness, but improving analyzers is likely to be difficult and would require careful determination of the proper level of conflation. In overcoming some of the difficulties associated with obtaining "better" morphology (or more fundamentally the proper level of word conflation), adaptive morphology done on a per query term basis or user feedback might prove valuable. Also, the scores that were used to rank the possible analyses in a statistical morphological analyzer may prove useful in further improving retrieval. Other IR techniques, such as improved blind relevance feedback or combination of evidence approaches, can also improve monolingual Arabic retrieval.

Perhaps improved morphology is particularly beneficial for other IR applications such as cross-language IR, in which ascertaining proper translation of words is particularly important, and

[3] The processing of the TREC collection using the in-context IBM-LM required 16 hours on a 2.4 GHz Pentium 4 machine with 1 Gigabyte of RAM compared to 10 minutes to perform light stemming.

in-document search term highlighting for display to a user.

References

1. Abu-Salem, H., M. Al-Omari, and M. Evens. Stemming Methodologies Over Individual Query Words for Arabic Information Retrieval. JASIS, 1999. 50(6): p. 524-529.

2. Aljlayl, M., S. Beitzel, E. Jensen, A. Chowdhury, D. Holmes, M. Lee, D. Grossman, and O. Frieder. IIT at TREC-10. In TREC. 2001. Gaithersburg, MD.

3. Al-Kharashi, I. and M Evens. Comparing Words, Stems, and Roots as Index Terms in an Arabic Information Retrieval System. JASIS, 1994. 45(8): p. 548 - 560.

4. Antworth, E. PC-KIMMO: a two-level processor for morphological analysis. In Occasional Publications in Academic Computing. 1990. Dallas, TX: Summer Institute of Linguistics.

5. Ahmed, Mohamed Attia. A Large-Scale Computational Processor of the Arabic Morphology, and Applications. A Master's Thesis, Faculty of Engineering, Cairo University, Cairo, Egypt, 2000.

6. Chen, A. and F. Gey. Translation Term Weighting and Combining Translation Resources in Cross-Language Retrieval. In TREC, 2001. Gaithersburg, MD.

7. Darwish, K. Building a Shallow Morphological Analyzer in One Day. ACL Workshop on Computational Approaches to Semitic Languages. 2002.

8. Darwish, K. and D. Oard. CLIR Experiments at Maryland for TREC 2002: Evidence Combination for Arabic-English Retrieval. In TREC. 2002. Gaithersburg, MD.

9. Darwish, K. and D. Oard. Term Selection for Searching Printed Arabic. SIGIR, 2002. Tampere, Finland. p. 261 - 268.

10. Gey, F. and D. Oard. The TREC-2001 Cross-Language Information Retrieval Track: Searching Arabic Using English, French or Arabic Queries. TREC, 2001. Gaithersburg, MD. p. 16-23.

11. Hmeidi, I., G. Kanaan, and M. Evens. Design and Implementation of Automatic Indexing for Information Retrieval with Arabic Documents. JASIS, 1997. 48(10): p. 867 - 881.

12. Kiraz, G. Arabic Computation Morphology in the West. In The 6th International Conference and Exhibition on Multi-lingual Computing. 1998. Cambridge.

13. Koskenniemi, K., Two Level Morphology: A General Computational Model for Word-form Recognition and Production. 1983, Department of General Linguistics, University of Helsinki.

14. Larkey, L., L. Ballesteros, and M. Connell. Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis. SIGIR 2002. p. 275-282, 2002.

15. Lee, Y., K. Papineni, S. Roukos, O. Emam, and H. Hassan. Language Model Based Arabic Word Segmentation. In the Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, July 2003, Sapporo, Japan. p. 399 - 406.

16. Mayfield, J., P. McNamee, C. Costello, C. Piatko, and A. Banerjee. JHU/APL at TREC 2001: Experiments in Filtering and in Arabic, Video, and Web Retrieval. In TREC 2001. Gaithersburg, MD. p. 322-329.

17. Oard, D. and F. Gey. The TREC 2002 Arabic/English CLIR Track. In TREC 2002. Gaithersburg, MD.

18. Sanderson, M. Word sense disambiguation and information retrieval. In Proceedings of the 17th ACM SIGIR Conference, p. 142-151, 1994

19. Xu, J., A. Fraser, and R. Weischedel. 2001 Cross-Lingual Retrieval at BBN. In TREC, 2001. Gaithersburg, MD. p. 68 - 75.

Modifying a Natural Language Processing System for European Languages to Treat Arabic in Information Processing and Information Retrieval Applications

Gregory Grefenstette, Nasredine Semmar, Faïza Elkateb-Gara
Multilingual Multimedia Knowledge Engineering Laboratory (LIC2M)
Commissariat à l'Energie Atomique, Laboratoire d'Intégration des Systèmes et des Technologies (CEA LIST)
B.P. 6, 92265 Fontenay-aux-Roses Cedex, France
{gregory.grefenstette,nasredine.semmar,faiza.gara}@cea.fr

Abstract

The goal of many natural language processing platforms is to be able to someday correctly treat all languages. Each new language, especially one from a new language family, provokes some modification and design changes. Here we present the changes that we had to introduce into our platform designed for European languages in order to handle a Semitic language. Treatment of Arabic was successfully integrated into our cross language information retrieval system, which is visible online.

1 Introduction

When a natural language processing (NLP) system is created in a modular fashion, it can be relatively easy to extend treatment to new languages (Maynard, *et al.* 2003) depending on the depth and completeness desired. We present here lessons learned from the extension of our NLP system that was originally implemented for Romance and Germanic European[1] languages to a member of the Semitic language family, Arabic. Though our system was designed modularly, this new language posed new problems. We present our answers to these problems encountered in the creation of an Arabic processing system, and illustrate its integration into an online cross language information retrieval (CLIR) system dealing with documents written in Arabic, English French and Spanish.

2 The LIMA natural language processor

Our NLP system (Besançon *et al*., 2003), called LIMA[2], was built using a traditional architecture involving separate modules for

1. Morphological analysis:
    a. Tokenization (separating the input stream into a graph of words).
    b. Simple word lookup (search for words in a full form lexicon).
    c. Orthographical alternative lookup (looking for differently accented forms, alternative hyphenisation, concatenated words, abbreviation recognition), which might alter the original non-cyclic word graph by adding alternative paths.
    d. Idiomatic expressions recognizer (detecting and considering them as single words in the word graph).
    e. Unknown word analysis.
2. Part-of-Speech and Syntactic analysis:
    a. After the morphological analysis, which has augmented the original graph with as many nodes as there

[1] European languages from non indo-European families (Basque, Finnish and Hungarian) pose some of the same problems that Arabic does.

[2] LIMA stands for the LIC2M Multilingual Analyzer.

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 31–38,
Ann Arbor, June 2005.

are interpretations for the tokens, part-of-speech analysis using language models from a hand-tagged corpus reduces the number of possible readings of the input.
   b. Named entity recognizer.
   c. Recognition of nominal and verbal chains in the graph.
   d. Dependency relation extraction.
3. Information retrieval application:
   a. Subgraph indexing.
   b. Query reformulation (monolingual reformulation for paraphrases and synonymy; multilingual for cross language information retrieval).
   c. Retrieval scoring comparing partial matches on subgraphs and entities.

Our LIMA NLP system (Besançon *et al*., 2003) was first implemented for English, French, German and Spanish, with all data coded in UTF8. When we extended the system to Arabic, we found that a number of modifications had to be introduced. We detail these modifications in the next sections.

3 Changes specific to Semitic languages

Two new problems posed by Arabic (and common to most Semitic languages) that forced us to alter our NLP system are the problem of incomplete vowelization of printed texts[3] and the problem of agglutinative clitics. We discuss how these new problems influenced our lexical resources and language processing steps.

Lexical Resources

The first task for introducing a new language is to create the lexical resources for this language. Since Arabic presents agglutination of articles, prepositions and conjunctions at the beginning of words as well as pronouns at the end of words, and these phenomena were not treated in our existing European languages[4], we had to decide how this feature would be handled in the lexicon. Solutions to this problem have been proposed, ranging from generation and storage of all agglutinated words forms (Debili and Zouari, 1985) to the compilation of valid sequences of proclitics, words and enclitics into finite-state machines (Beesley, 1996). Our system had already addressed the problem of compounds for German in the following way: if an input word is not present in the dictionary, a compound-searching module returns all complete sequences of dictionary words (a list of possible compound joining "fogemorphemes" is passed to this module) as valid decompositions of the input word. Though theoretically this method could be used to treat Arabic clitics, we decided against using this existing process for two reasons:

1. Contrary to German, in which any noun may theoretically be the first element of a compound, Arabic clitics belong to a small closed set of articles, conjunctions, prepositions and pronouns. Allowing any word to appear at the beginning or end of an agglutinated word would generate unnecessary noise.
2. Storing all words with all possible clitics would multiply the size of lexicon proportionally to the number of legal possible combinations. We decided that this would take up too much space, though others have adopted this approach as mentioned above.

We decided to create three lexicons: two additional (small) lists of proclitic and enclitic combinations, and one large lexicon of full form[5] voweled words (with no clitics), the creation of the large lexicon from a set of lemmas using classic conjugation rules did not require any modification of the existing dictionary building and compilation component. Since our NLP system already possessed a mechanism for mapping unaccented words to accented entries, and we decided to use this existing

[3] Since the headwords of our monolingual and cross-lingual reference dictionaries for Arabic possess voweled entries, we hope to attain greater precision by treating this problem. An alternative but noisy approach (Larkey et al. 2002) is to reduce to unvoweled text throughout the NLP application.

[4] Spanish, of course, possesses enclitic pronouns for some verb forms but these were not adequately treated until the solution for Arabic was implemented in our system.

[5] Our dictionary making process generates all full form versions of non compound and unagglutinated words. These are e then compiled into a finite-state automaton. Every node corresponding to a full word is flagged, and an index corresponding to the automaton path points to the lexical data for that word.

mechanism for later matching of voweled and unvoweled versions of Arabic words in applications. Thus the only changes for lexical resources involve adding two small clitic lexicons.

Processing Steps: Morphological analysis

Going back to the NLP processing steps listed in section 2, we now discuss new processing changes needed for treating Arabic. Tokenization (1a) and simple word lookup (2a) of the tokenized strings in the dictionary were unchanged as LIMA was coded for UTF8. If the word was not found, an existing orthographical alternative lookup (1c) was also used without change (except for the addition of the language specific correspondence table between accented and unaccented characters) in order to find lexical entries for unvoweled or partially voweled words. Using this existing mechanism for treating the vowelization problem does not allow us to exploit partial vowelization as we explain in a later section.

At this point in the processing, a word that contains clitics will not have been found in the dictionary since we had decided not to include word forms including clitics. We introduced, here, a new processing step for Arabic: a clitic stemmer. This stemmer uses the following linguistic resources:

- The full form dictionary, containing for each word form its possible part-of-speech tags and linguistic features (gender, number, etc.). We currently have 5.4 million entries in this dictionary[6].
- The proclitic dictionary and the enclitic dictionary, having the same structure of the full form dictionary with voweled and unvoweled versions of each valid combination of clitics. There are 77 and 65 entries respectively in each dictionary.

The clitic stemmer proceeds as follows on tokens unrecognized after step 1c:

- Several vowel form normalizations are performed (ً ٌ ٍ ّ ُ َ are removed, آ إ أ are replaced by ا and final ؤ ئ ي or ة are replaced by ء ىء ى or ه).
- All clitic possibilities are computed by using proclitics and enclitics dictionaries.
- A radical, computed by removing these clitics, is checked against the full form lexicon. If it does not exist in the full form lexicon, re-write rules (such as those described in Darwish (2002)) are applied, and the altered form is checked against the full form dictionary. For example, consider the token وهواهم and the included clitics (و, هم), the computed radical هوا does not exist in the full form lexicon but after applying one of the dozen re-write rules, the modified radical هوى is found the dictionary and the input token is segmented into root and clitics as: وهواهم = و + هوى + هم.
- The compatibility of the morpho-syntactic tags of the three components (proclitic, radical, enclitic) is then checked. Only valid segmentations are kept and added into the word graph. Table 1 gives some examples of segmentations[7] of words in the sentence من جانبها أكدت وزارة الداخلية العراقية

Agglutinated word	Segmentations of the agglutinated word
ومن	ومن = و + من
جانبها	جانبها = جانب + ها
الداخلية	الداخلية = ال + داخلية الداخلية = [ا + ل] + داخلية
العراقية	العراقية = ال + عراقية العراقية = [ا + ل] + عراقية
المحافظات	المحافظات = ال + محافظات المحافظات = [ا + ل] + محافظات
للخطف	للخطف = [ل + ال] + خطف
الوزير	الوزير = ال + وزير الوزير = [ا + ل] + وزير
نفسه	نفسه = نفس + ه

Table 1: Segmentations of some agglutinated words.

Producing this new clitic stemmer for Arabic allowed us to correctly treat a similar (but previously ignored) phenomenon in Spanish in which verb forms can possess pronominal enclitics. For example, the imperative form of "give to me" is written as "dame", which corresponds to the radical "da" followed the enclitic "me". Once we implemented this clitic stemmer for Arabic, we created an en-

[6] If we generated all forms including appended clitics, we would generate an estimated 60 billion forms (Attia, 1999).

[7] For example, the agglutinated word الداخلية has two segmentations but only the segmentation: الداخلية = ال + داخلية will remain after POS tagging in step 2a

clitic dictionary for Spanish and then successfully used the same stemmer for this European language. At this point, the treatment resumes as with European languages. The detection of idiomatic[8] expressions (step 1d) is performed after clitic separation using rules associated with trigger words for each expression. Once a trigger is found, its left and right lexical contexts in the rule are then tested. The trigger must be an entry in the full form lexicon, but can be represented as either a surface form or a lemma form combined with its morpho-syntactic tag. Here we came across another problem specific to Semitic languages. Since Arabic lexicon entries are voweled and since input texts may be partially voweled or unvoweled, we are forced to only use lemma forms to describe Arabic idiomatic expressions rules with the existing mechanism, or else enter all the possible partial vowelizations for each word in an idiomatic expression. Since, at this point after step 1c, each recognized word is represented with all its possible voweled lemmas in the analysis graph, we developed 482 contiguous idiomatic voweled expression rules. For example one of the developed rules recognizes in the text كانون الثاني (January) as a whole and tags the expression as a being a month.

After idiomatic expression recognition, any nodes not yet recognized are assigned (in step 1e) default linguistic values based on features recognized during tokenization (e.g. presence of uppercase or numbers or special characters). Nothing was changed for this step of default value assignment in order to treat Arabic, but since Semitic languages do not have the capitalization clues that English and French have for recognizing proper and since Arabic proper names can often be decomposed into simple words (much like Chinese names), the current implementation of this step with our current lexical resources poses some problems.

For example, consider the following sentence:
فرانك لامبارد يحتفل بالتسجيل لتشلسي وزميله إيدور غوديانسن يشاركه الفرحة *Frank Lampard celebrates the score by Chelsea and his team mate Eidur Gudjohnsen shares his elation*. The name فرانك (Frank) is identified as such because it is found in the lexicon; the name لامبارد (Lampard) is not in the lexicon and incorrectly stemmed as لا + مبارد (plural of the noun مبرد (grater)); the name إيدور (Eidur) is incorrectly tagged as a verb; and غوديانسن (Gudjohnsen), which is not in the dictionary and for which the clitic stemmer does not produce any solutions receives the default tags adjective, noun, proper noun and verb, to be decided by the part-of-speech tagger. To improve this performance, we plan to enrich the Arabic lexicon with more proper names, using either name recognition (Maloney and Niv, 1998) or a back translation approach after name recognition in English texts (Al-Onaizan and Knight, 2002).

Processing Steps: Part-of-speech analysis

For the succeeding steps involving part-of-speech tagging, named entity recognition, division into nominal and verbal chains, and dependency extraction no changes were necessary for treating Arabic. After morphological analysis, as input to step 2a, part-of-speech tagging, we have the same type of word graph for Arabic text as for European text: each node is annotated with the surface form, a lemma and a part-of-speech in the graph. If a word is ambiguous, then more than one node appears in the graph for that word. Our part-of-speech tagging involves using a language model (bigrams and trigrams of grammatical tags) derived from hand-tagged text to eliminate unattested or rare sub paths in the graph of words representing a sentence. For Arabic, we created a hand-tagged corpus, and where then able to exploit the existing mechanism.

One space problem that has arisen in applying the existing processing designed for European languages comes from the problem of vowelization. With our previous European languages, it was extremely rare to have more than one possible lemmatization for a given pair: (surface form, grammatical part-of-speech tag)[9]. But, in Arabic this can be very common since an unvoweled string can correspond to many different words, some with the same part-of-speech but different lemmas. The effect of this previously unseen type of ambiguity on our data structures was to greatly increase the word graph size before and after part-of-speech tagging. Since each combination of (sur-

[8] An *idiom* in our system is a (possibly non-contiguous sequence) of known words that act as a single unit. For example, *made up* in *He made up the story on the spot*. Once an idiomatic expression is recognized the individual words nodes are joined into one node in the word graph.

[9] One example from French is the pair (*étaient*, finite-verb) that can correspond to the two lemmas: *être* and *étayer*.

face-form, part-of-speech-tag, and lemma) gives rise to a new node, the graph becomes larger, increasing the number of paths that all processing steps must explore. The solution to this for Arabic and other Semitic languages is simple, though we have not yet implemented it. We plan to modify our internal data structure so that each node will correspond to the surface form, a part-of-speech tag, and a set of lemmas: (surface-form, part-of-speech-tag, {lemmas}). The inclusion of a set of possible lemmas, rather than just one lemma, in a node will greatly reduce the number of nodes in the graph and speed processing time.

The next step in our NLP system, after part-of-speech tagging, is named entity recognition (Abuleil and Evans, 2004) using name triggers (e.g., President, lake, corporation, etc.). Beyond the problem mentioned above of distinguishing possible proper nouns, here we had an additional problem since our recognizer extracted the entity in its surface form. Since in Arabic, as in other Semitic languages, the input text is usually only partially voweled, this gave rise to many different forms (corresponding to different surface forms) for the same entity. This minor problem was solved by storing the fully voweled forms of the entities (for application such as information retrieval as shown below) rather than the surface form.

After named entity recognition, our methods of verbal and nominal chain recognition and dependency extraction did not require any modifications for Arabic. But since the sentence graphs, as mentioned above, are currently large, we have restricted the chains recognized to simple noun and verb chunks (Abney, 1991) rather than the more complex chains (Marsh, 1984) we recognize for European languages. Likewise, the only dependency relations that we extract for the moment are relations between nominal elements. We expect that the reduction in sentence graph once lemmas are all collected in the same word node will allow us to treat more complex dependency relations.

4 Integration in a CLIR application

The results of the NLP steps produce, for all languages we treat, a set of normalized lemmas, a set of named entities and a set of nominal compounds (as well as other dependency relations for some languages). These results can be used for any natural language processing application. For example, we have integrated LIMA as a front-end for a cross language information retrieval system. The inclusion of our Arabic language results into the information retrieval system did not necessitate any modifications to this system.

This information retrieval (IR) application involves three linguistic steps, as shown in section 2. First, in step 3a, subgraphs (compounds and their components) of the original sentence graph are stored. For example, the NLP analysis will recognize an English phrase such as "management of water resources" as a compound that the IR system will index. This phrase and its sub-elements are normalized and indexed (as well as simple words) in the following head-first normalized forms:

- management_water_resource
- resource_water
- management_resource

Parallel head-first structures are created for different languages, for example, the French "gestion des ressource en eau" generates:

- gestion_ressource_eau
- ressource_eau
- gestion_ressource.

The corresponding Arabic phrase: إدارة موارد المياه is likewise indexed with voweled forms:

- مَاء_مَـوْرِد_ إدَارَة
- مَاء_مَـوْرِد
- مَـوْرِد_ إدَارَة

When a question is posed to our cross language IR (CLIR) system it undergoes the same NLP treatment as in steps 1a to 3a. Then the query is reformulated using synonym dictionaries and translation dictionaries in step 3b. For Arabic, we have not yet acquired any monolingual synonym dictionaries, but we have purchased and modified cross-lingual transfer dictionaries between Arabic and English, Arabic and French, and Arabic and Spanish[10]. When a compound is found in a query, it is normalized and its sub elements are extracted as shown above. Using the reformulation dictionaries, variant versions of the compound are generated (monolingual, then cross-lingual versions) and at-

[10] Lindén and Piitulainen (2004) propose a method for extracting monolingual synonym lists from bilingual resources.

tested variants are retained as synonyms to the original compound[11] (Besançon *et al*., 2003). To integrate the Arabic version into our CLIR system, no modifications were necessary beyond acquiring and formatting the cross language reformulation dictionaries.

The final NLP step (3c) involving in our CLIR system involves ranking relevant documents. Contrary to a bag of word system, which uses only term frequency in queries and documents, our system (Besançon *et al*., 2003) returns documents in ranked weighted classes[12] whose weightings involve the presence of named entities, the completeness of the syntactic subgraphs matched, and the database frequencies of the words and subgraphs matched.

Example

An online version of our cross language retrieval system involving our Arabic processing is visible online at a third party site: http://alma.oieau.fr. This base contains 50 non-parallel documents about sustainable development for each of the following languages: English, Spanish, French and Arabic. The user can enter a query in natural language and specify the language to be used. In the example of the Figure 1, the user entered the query "إدارة موارد المياه" and selected Arabic as the language of the query.

Relevant documents are grouped into classes characterized by the same set of concepts (i.e., reformulated subgraphs) as the query contains. Figure 2 shows some classes corresponding to the query " إدارة موارد المياه". The query term إدارة_موارد_مياه is a term composed of three words: مياه, موارد and إدارة. This compounds, its derived variants and their sub elements are reformulated into English, French, and Spanish and submitted to indexed versions of documents in each of these languages (as well as against Arabic documents). The highest ranking classes (as seen in Figure 2 for this example) match the following elements:

Class	Query terms	Number of retrieved documents
1	إدارة_موارد_مياه	14
2	إدارة_موارد، موارد_مياه	18
3	مياه، إدارة_موارد	9

Terms of the query or the expansion of these terms which are found in the retrieved documents are highlighted as illustrated in Figures 2 and 3.

5 Conclusion

We have presented here an overview of our natural language processing system and its use in a CLIR setting. This article describes the changes that we had to implement to extend this system, which was initially implemented for treating European languages to the Semitic language, Arabic. Every new language possesses new problems for NLP systems, but treating a language from a new language family can severely test the original design. We found that the major problems we encountered in dealing with a language from the Semitic language family involved the problems of dealing with partially voweled or unvoweled text (two different problems), and of dealing with clitics. To treat the problem of clitics, we introduced two new lexicons and added an additional clitic stemming step at an appropriate place in our morphological analysis. For treating the problem of vowelization, we simply used existing methods for dealing with unaccented text, but this solution is not totally satisfactory for two reasons: we do not adequately exploit partially voweled text, and our data structures are not efficient for associating many different lemma (differing only in vowelization) with a single surface form. We are currently working on both these aspects in order to improve our treatment of Arabic. But the changes, that we describe here, involved in adding Arabic were not very extensive, and we able to integrate Arabic language treatment into a cross language information retrieval platform using one man-year of work after having created the lexicon and training corpus. A version of our CLIR is available online and illustrated in this article. We plan to more fully evaluate the performance of the CLIR system using the TREC 2001 and TREC 2002 in the coming year.

[11] This technique will only work with translations which have at least one subelement that is has a parallel between languages, but this is often the case for technical terms.

[12] This return to a mixed Boolean approach is found in current research on Question Answering systems (Tellex *et al.*, 2003). Our CLIR system resembles such systems, which return the passage in which the answer is found, since we highlight the most significant passages of each retrieved document.

References

Steven Abney. Parsing by Chunks. 1991. In R. C. Berwick, S. P. Abney, and C. Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, Kluwer Academic Publishers, Boston.

Saleem Abuleil, Martha Evens. 2004. Named Entity Recognition and Classification for Text in Arabic. *IASSE 2004*, pp. 89-94

Mohamed Attia. 1999. A large-Scale Computational Processor of Arabic Morpholofy, and Applications. M.S. thesis in Computer Engineering, Cairo University, pp. 28-32.

Y. Al-Onaizan and K. Knight. 2002. Machine Transliteration of Names in Arabic Text. *Proc. of ACL Workshop on Computational Approaches to Semitic Languages*, pp. 400-408

Kenneth Beesley. 1996. Arabic Finite-State Morphological Analysis and Generation. *Proc. of COLING-96*, pp. 89-94.

Romaric Besançon, Gaël de Chalendar, Olivier Ferret, Christian Fluhr, Olivier Mesnard, and Hubert Naets. 2003. Concept-Based Searching and Merging for Multilingual Information Retrieval: First Experiments at CLEF 2003. *CLEF-2003*, pp. 174-184.

K. Darwish. 2002. Building a Shallow Arabic Morphological Analyzer in One Day. In *Proc. ACL-02*, pp. 47-54

Fathi Debili and Lotfi Zouari. 1985. Analyse morphologique de l'arabe écrit voyellé ou non fondée sur la construction automatique d'un dictionnaire arabe, *Cognitiva*, Paris.

Leah S. Larkey, Lisa Ballesteros, Margaret E. Connell. 2002. Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. *Proc. of SIGIR 2002*, pp. 275-282

Krister Lindén and Jussi Piitulainen. 2004. Discovering Synonyms and Other Related Words. *CompuTerm 2004*, Geneva, Switzerland, August 29.

John Maloney and Michael Niv. 1998. TAGARAB: A Fast, Accurate Arabic Name Recogniser Using High Precision Morphological Analysis. *Proc. of the Workshop on Computational Approaches to Semitic Languages*. Montreal, Canada. August.

Elain Marsh. 1984. A Computational Analysis of Complex Noun Phrases in Navy Messages. In *Proc. of COLING '84*, Stanford, pp. 505-508.

Diana Maynard, Valentin Tablan, Kalina Bontcheva, Hamish Cunningham. 2003. Rapid Customization of an Information Extraction System for a Surprise Language. *ACM Trans. Asian Lang. Inf. Process.* 2(3) pp. 295-300.

Stefanie Tellex, Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes. 2003. Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. *Proc. Of SIGIR 2003*, pp. 41-47

Figure 1: User interface for querying the database. The user can choose between English, French, Spanish and Arabic as input language. For best results, the query should be syntactically correct and not in telegraphic form.

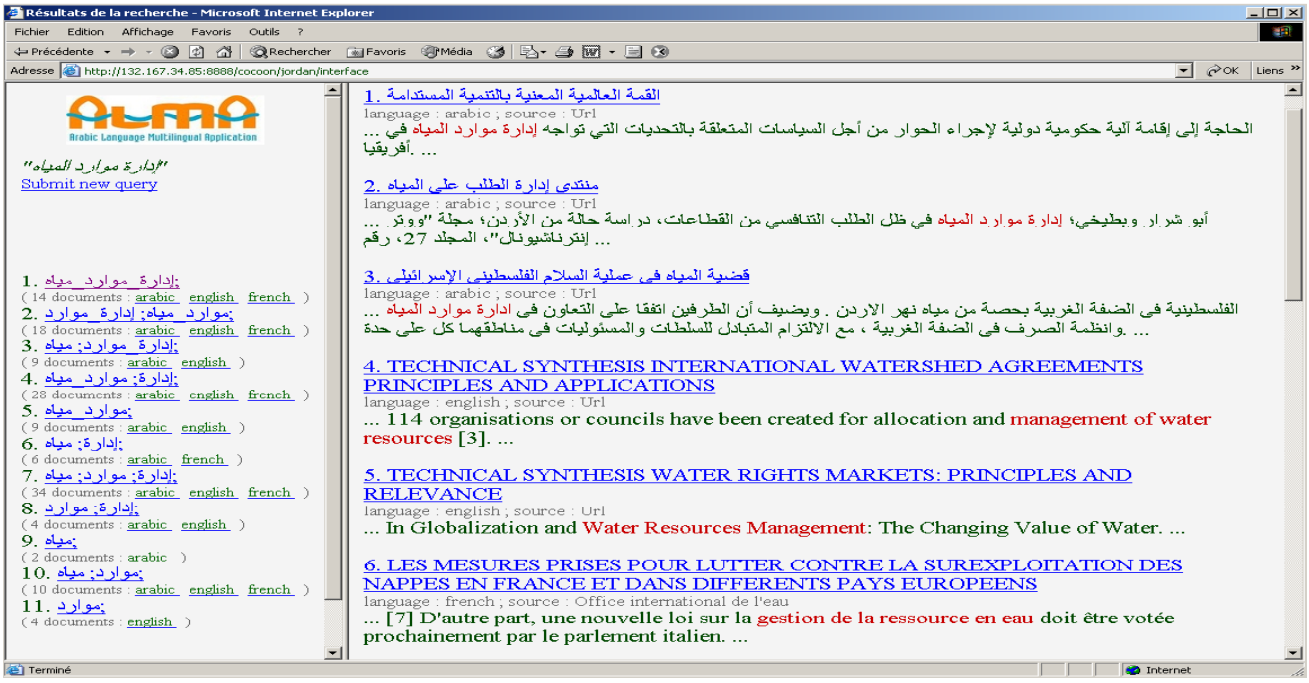

Figure 2: Search results user interface. Results can appear in many languages.

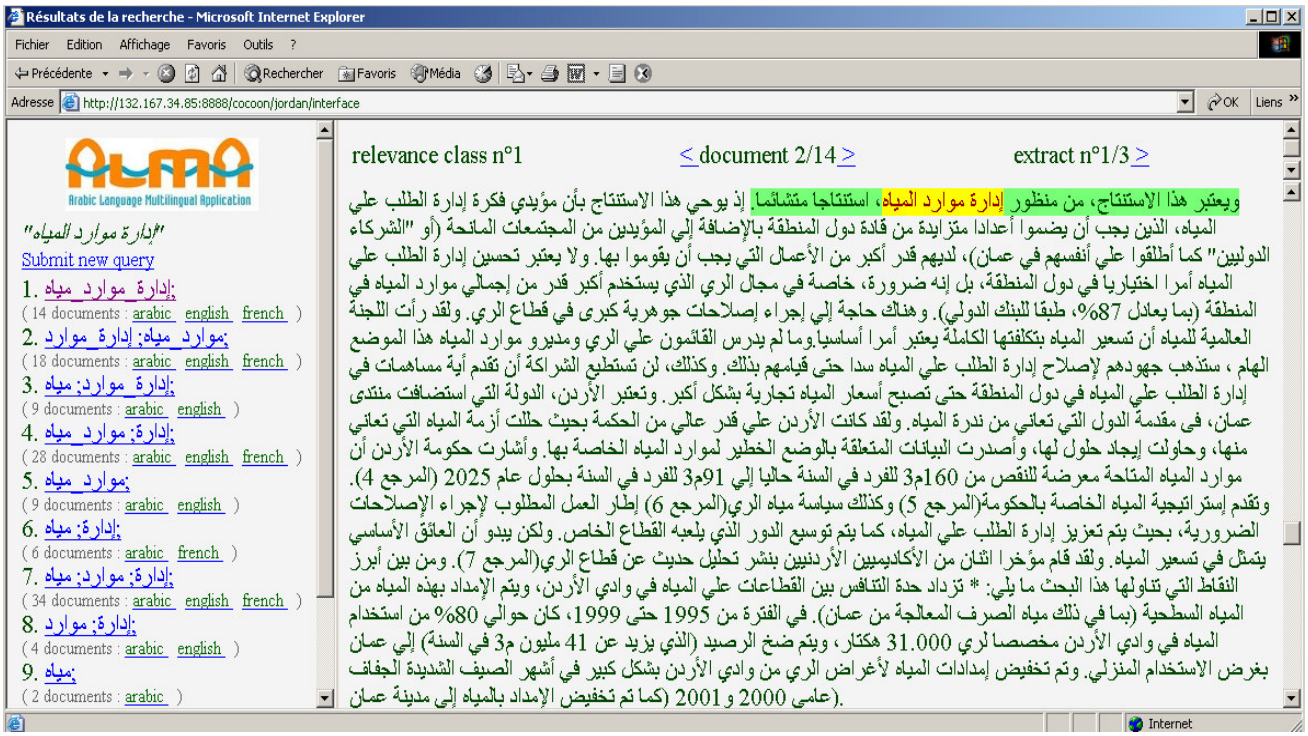

Figure 3: Highlighting query terms in retrieved documents.

Choosing an Optimal Architecture for Segmentation and POS-Tagging of Modern Hebrew

Roy Bar-Haim
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
barhair@cs.biu.ac.il

Khalil Sima'an
ILLC
Universiteit van Amsterdam
Amsterdam, The Netherlands
simaan@science.uva.nl

Yoad Winter
Dept. of Computer Science
Technion
Haifa 32000, Israel
winter@cs.technion.ac.il

Abstract

A major architectural decision in designing a disambiguation model for segmentation and Part-of-Speech (POS) tagging in Semitic languages concerns the choice of the input-output terminal symbols over which the probability distributions are defined. In this paper we develop a segmenter and a tagger for Hebrew based on Hidden Markov Models (HMMs). We start out from a morphological analyzer and a very small morphologically annotated corpus. We show that a model whose terminal symbols are word segments (=morphemes), is advantageous over a word-level model for the task of POS tagging. However, for segmentation alone, the morpheme-level model has no significant advantage over the word-level model. Error analysis shows that both models are not adequate for resolving a common type of segmentation ambiguity in Hebrew – whether or not a word in a written text is prefixed by a definiteness marker. Hence, we propose a morpheme-level model where the definiteness morpheme is treated as a possible feature of morpheme terminals. This model exhibits the best overall performance, both in POS tagging and in segmentation. Despite the small size of the annotated corpus available for Hebrew, the results achieved using our best model are on par with recent results on Modern Standard Arabic.

1 Introduction

Texts in Semitic languages like Modern Hebrew (henceforth *Hebrew*) and Modern Standard Arabic (henceforth *Arabic*), are based on writing systems that allow the concatenation of different lexical units, called *morphemes*. Morphemes may belong to various Part-of-Speech (POS) classes, and their concatenation forms textual units delimited by white space, which are commonly referred to as *words*. Hence, the task of POS tagging for Semitic languages consists of a segmentation subtask and a classification subtask. Crucially, words can be segmented into different alternative morpheme sequences, where in each segmentation morphemes may be ambiguous in terms of their POS tag. This results in a high level of overall ambiguity, aggravated by the lack of vocalization in modern Semitic texts.

One crucial problem concerning POS tagging of Semitic languages is how to adapt existing methods in the best way, and which architectural choices have to be made in light of the limited availability of annotated corpora (especially for Hebrew). This paper outlines some alternative architectures for POS tagging of Hebrew text, and studies them empirically. This leads to some general conclusions about the optimal architecture for disambiguating Hebrew, and (reasonably) other Semitic languages as well. The choice of tokenization level has major consequences for the implementation using HMMs, the sparseness of the statistics, the balance of the Markov condi-

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 39–46,
Ann Arbor, June 2005.

tioning, and the possible loss of information. The paper reports on extensive experiments for comparing different architectures and studying the effects of this choice on the overall result. Our best result is on par with the best reported POS tagging results for Arabic, despite the much smaller size of our annotated corpus.

The paper is structured as follows. Section 2 defines the task of POS tagging in Hebrew, describes the existing corpora and discusses existing related work. Section 3 concentrates on defining the different levels of tokenization, specifies the details of the probabilistic framework that the tagger employs, and describes the techniques used for smoothing the probability estimates. Section 4 compares the different levels of tokenization empirically, discusses their limitations, and proposes an improved model, which outperforms both of the initial models. Finally, section 5 discusses the conclusions of our study for segmentation and POS tagging of Hebrew in particular, and Semitic languages in general.

2 Task definition, corpora and related work

Words in Hebrew texts, similar to words in Arabic and other Semitic languages, consist of a stem and optional prefixes and suffixes. *Prefixes* include conjunctions, prepositions, complementizers and the definiteness marker (in a strict well-defined order). *Suffixes* include inflectional suffixes (denoting gender, number, person and tense), pronominal complements with verbs and prepositions, and possessive pronouns with nouns.

By the term *word segmentation* we henceforth refer to identifying the prefixes, the stem and suffixes of the word. By *POS tag disambiguation* we mean the assignment of a proper POS tag to each of these morphemes.

In defining the task of segmentation and POS tagging, we ignore part of the information that is usually found in Hebrew morphological analyses. The internal morphological structure of stems is not analyzed, and the POS tag assigned to stems includes no information about their root, template/pattern, inflectional features and suffixes. Only pronominal complement suffixes on verbs and prepositions are identified as separate morphemes. The construct state/absolute,[1] and the existence of a possessive suffix are identified using the POS tag assigned to the stem, and not as a separate segment or feature. Some of these conventions are illustrated by the segmentation and POS tagging of the word *wfnpgfnw* ("and that we met", pronounced *ve-she-nifgashnu*):[2]

w/CC:	conjunction
f/COM:	complementizer
npgfnw/VB:	verb

Our segmentation and POS tagging conform with the annotation scheme used in the Hebrew Treebank (Sima'an et al., 2001), described next.

2.1 Available corpora

The Hebrew Treebank (Sima'an et al., 2001) consists of syntactically annotated sentences taken from articles from the *Ha'aretz* daily newspaper. We extracted from the treebank a mapping from each word to its analysis as a sequence of POS tagged morphemes. The treebank version used in the current work contains 57 articles, which amount to 1,892 sentences, 35,848 words, and 48,332 morphemes. In addition to the manually tagged corpus, we have access to an untagged corpus containing 337,651 words, also originating from *Ha'aretz* newspaper.

The tag set, containing 28 categories, was obtained from the full morphological tagging by removing the gender, number, person and tense features. This tag set was used for training the POS tagger. In the evaluation of the results, however, we perform a further grouping of some POS tags, leading to a reduced POS tag set of 21 categories. The tag set and the grouping scheme are shown below:
{NN}, {NN-H}, {NNT}, {NNP}, {PRP,AGR}, {JJ}, {JJT}, {RB,MOD}, {RBR}, {VB,AUX}, {VB-M}, {IN,COM,REL}, {CC}, {QW}, {HAM}, {WDT,DT}, {CD,CDT}, {AT}, {H}, {POS}, {ZVL}.

2.2 Related work on Hebrew and Arabic

Due to the lack of substantial tagged corpora, most previous corpus-based work on Hebrew focus on the

[1] The Semitic construct state is a special form of a word that participates in compounds. For instance, in the Hebrew compound *bdiqt hjenh* ("*check of the claim*"), the word *bdiqt* ("check of"/"test of") is the construct form of the absolute form *bdiqh* ("check"/"test").

[2] In this paper we use Latin transliteration for Hebrew letters following (Sima'an et al., 2001).

development of techniques for learning probabilities from large unannotated corpora. The candidate analyses for each word were usually obtained from a morphological analyzer.

Levinger et al. (1995) propose a method for choosing a most probable analysis for Hebrew words using an unannotated corpus, where each analysis consists of the lemma and a set of morphological features. They estimate the relative frequencies of the possible analyses for a given word w by defining a set of "similar words" $SW(A)$ for each possible analysis A of w. Each word w' in $SW(A)$ corresponds to an analysis A' which differs from A in exactly one feature. Since each set is expected to contain different words, it is possible to approximate the frequency of the different analyses using the average frequency of the words in each set, estimated from the untagged corpus.

Carmel and Maarek (1999) follow Levinger et al. in estimating context independent probabilities from an untagged corpus. Their algorithm learns frequencies of morphological patterns (combinations of morphological features) from the unambiguous words in the corpus.

Several works aimed at improving the "similar words" method by considering the context of the word. Levinger (1992) adds a short context filter that enforces grammatical constraints and rules out impossible analyses. Segal's (2000) system includes, in addition to a somewhat different implementation of "similar words", two additional components: correction rules *à la* Brill (1995), and a rudimentary deterministic syntactic parser.

Using HMMs for POS tagging and segmenting Hebrew was previously discussed in (Adler, 2001). The HMM in Adler's work is trained on an untagged corpus, using the Baum-Welch algorithm (Baum, 1972). Adler suggests various methods for performing both tagging and segmentation, most notable are (a) The usage of word-level tags, which uniquely determine the segmentation and the tag of each morpheme, and (b) The usage of a two-dimensional Markov model with morpheme-level tags. Only the first method (word-level tags) was tested, resulting in an accuracy of 82%. In the present paper, both word-level tagging and morpheme-level tagging are evaluated.

Moving on to Arabic, Lee et al. (2003) describe a word segmentation system for Arabic that uses an n-gram language model over morphemes. They start with a seed segmenter, based on a language model and a stem vocabulary derived from a manually segmented corpus. The seed segmenter is improved iteratively by applying a bootstrapping scheme to a large unsegmented corpus. Their system achieves accuracy of 97.1% (per word).

Diab et al. (2004) use Support Vector Machines (SVMs) for the tasks of word segmentation and POS tagging (and also Base Phrase Chunking). For segmentation, they report precision of 99.09% and recall of 99.15%, when measuring *morphemes* that were correctly identified. For tagging, Diab et al. report accuracy of 95.49%, with a tag set of 24 POS tags. Tagging was applied to segmented words, using the "gold" segmentation from the annotated corpus (Mona Diab, p.c.).

3 Architectures for POS tagging Semitic languages

Our segmentation and POS tagging system consists of a *morphological analyzer* that assigns a set of possible candidate analyses to each word, and a *disambiguator* that selects from this set a single preferred analysis per word. Each candidate analysis consists of a segmentation of the word into morphemes, and a POS tag assignment to these morphemes. In this section we concentrate on the architectural decisions in devising an optimal disambiguator, given a morphological analyzer for Hebrew (or another Semitic language).

3.1 Defining the input/output

An initial crucial decision in building a disambiguator for a Semitic text concerns the "tokenization" of the input sentence: what constitutes a terminal (i.e., input) symbol. Unlike English POS tagging, where the terminals are usually assumed to be words (delimited by white spaces), in Semitic texts there are two reasonable options for fixing the kind of terminal symbols, which directly define the corresponding kind of nonterminal (i.e., output) symbols:

Words (W): The terminals are words as they appear in the text. In this case a nonterminal $\boldsymbol{a}$ that is assigned to a word w consists of *a sequence* of POS tags, each assigned to a mor-

pheme of w, delimited with a special segmentation symbol. We henceforth refer to such complex nonterminals as *analyses*. For instance, the analysis `IN-H-NN` for the Hebrew word *bbit* uniquely encodes the segmentation *b-h-bit*. In Hebrew, this unique encoding of the segmentation by the sequence of POS tags in the analysis is a general property: given a word w and a complex nonterminal $\boldsymbol{a} = [t_1 \dots t_p]$ for w, it is possible to extend $\boldsymbol{a}$ back to a full analysis $\tilde{\boldsymbol{a}} = [(m_1, t_1) \dots (m_p, t_p)]$, which includes the morphemes $m_1 \dots m_p$ that make out w. This is done by finding a match for $\boldsymbol{a}$ in $Analyses(w)$, the set of possible analyses of w. Except for very rare cases, this match is unique.

Morphemes (M): In this case the nonterminals are the usual POS tags, and the segmentation is given by the input morpheme sequence. Note that information about how morphemes are joined into words is lost in this case.

Having described the main input-output options for the disambiguator, we move on to describing the probabilistic framework that underlies their workings.

3.2 The probabilistic framework

Let w_1^k be the input sentence, a sequence of words $w_1 \dots w_k$. If tokenization is per word, then the disambiguator aims at finding the nonterminal sequence $\boldsymbol{a}_1^k$ that has the highest joint probability with the given sentence w_1^k:

$$\arg\max_{\boldsymbol{a}_1^k} P(w_1^k, \boldsymbol{a}_1^k) \tag{1}$$

This setting is the standard formulation of probabilistic tagging for languages like English.

If tokenization is per morpheme, the disambiguator aims at finding a combination of a segmentation m_1^n and a tagging t_1^n for m_1^n, such that their joint probability with the given sentence, w_1^k, is maximized:

$$\arg\max_{(m_1^n, t_1^n) \in ANALYSES(w_1^k)} P(w_1^k, m_1^n, t_1^n), \tag{2}$$

where $ANALYSES(w_1^k)$ is the set of possible analyses for the input sentence w_1^k (output by the morphological analyzer). Note that n can be different from k, and may vary for different segmentations. The original sentence can be uniquely recovered from the segmentation and the tagging. Since all the $\langle m_1^n, t_1^n \rangle$ pairs that are the input for the disambiguator were derived from w_1^k, we have $P(w_1^k | m_1^n, t_1^n) = 1$, and thus $P(w_1^k, m_1^n, t_1^n) = P(t_1^n, m_1^n)$. Therefore, Formula (2) can be simplified as:

$$\arg\max_{(m_1^n, t_1^n) \in ANALYSES(w_1^k)} P(m_1^n, t_1^n) \tag{3}$$

Formulas (1) and (3) can be represented in a unified formula that applies to both word tokenization and morpheme tokenization:

$$\arg\max_{(e_1^n, A_1^n) \in ANALYSES(w_1^k)} P(e_1^n, A_1^n) \tag{4}$$

In Formula (4) e_1^n represents either a sequence of words or a sequence of morphemes, depending on the level of tokenization, and A_1^n are the respective nonterminals – either POS tags or word-level analyses. Thus, the disambiguator aims at finding the most probable $\langle terminal\ sequence, nonterminal\ sequence \rangle$ for the given sentence, where in the case of word-tokenization there is only one possible terminal sequence for the sentence.

3.3 HMM probabilistic model

The actual probabilistic model used in this work for estimating $P(e_1^n, A_1^n)$ is based on Hidden Markov Models (HMMs). HMMs underly many successful POS taggers , e.g. (Church, 1988; Charniak et al., 1993).

For a k-th order Markov model ($k = 1$ or $k = 2$), we rewrite (4) as:

$$\arg\max_{e_1^n, A_1^n} P(e_1^n, A_1^n) \approx$$
$$\arg\max_{e_1^n, A_1^n} \prod_{i=1}^{n} P(A_i \mid A_{i-k}, \dots, A_{i-1}) P(e_i \mid A_i) \tag{5}$$

For reasons of data sparseness, actual models we use work with $k = 2$ for the morpheme level tokenization, and with $k = 1$ for the word level tokenization.

For these models, two kinds of probabilities need to be estimated: $P(e_i \mid A_i)$ (lexical model) and $P(A_i \mid A_{i-k}, \ldots, A_{i-1})$ (language model). Because the only manually POS tagged corpus that was available to us for training the HMM was relatively small (less than 4% of the Wall Street Journal (WSJ) portion of the Penn treebank), it is inevitable that major effort must be dedicated to alleviating the sparseness problems that arise. For smoothing the nonterminal language model probabilities we employ the standard backoff smoothing method of Katz (1987).

Naturally, the relative frequency estimates of the lexical model suffer from more severe data-sparseness than the estimates for the language model. On average, 31.3% of the test words do not appear in the training corpus. Our smoothing method for the lexical probabilities is described next.

3.4 Bootstrapping a better lexical model

For the sake of exposition, we assume word-level tokenization for the rest of this subsection. The method used for the morpheme-level tagger is very similar.

The smoothing of the lexical probability of a word w given an analysis $\mathbf{a}$, i.e., $P(w \mid \mathbf{a}) = \frac{P(w,\mathbf{a})}{P(\mathbf{a})}$, is accomplished by smoothing the joint probability $P(w, \mathbf{a})$ only, i.e., we do not smooth $P(\mathbf{a})$.[3] To smooth $P(w, \mathbf{a})$, we use a linear interpolation of the relative frequency estimates from the annotated training corpus (denoted $\mathbf{rf}_{tr}(w, \mathbf{a})$) together with estimates obtained by *unsupervised estimation* from a large unannotated corpus (denoted $\mathbf{em}_{auto}(w, \mathbf{a})$):

$$P(w, \mathbf{a}) = \lambda\, \mathbf{rf}_{tr}(w, \mathbf{a}) + (1-\lambda)\, \mathbf{em}_{auto}(w, \mathbf{a}) \quad (6)$$

where λ is an interpolation factor, experimentally set to 0.85.

Our unsupervised estimation method can be viewed as a single iteration of the Baum-Welch (Forward-Backward) estimation algorithm (Baum, 1972) with minor differences. We apply this method to the untagged corpus of 340K words. Our method starts out from a naively smoothed relative frequency lexical model in our POS tagger:

$$P_{LM_0}(w|\mathbf{a}) = \begin{cases} (1-p_0)\, \mathbf{rf}_{tr}(w, \mathbf{a}) & f_{tr}(w) > 0 \\ p_0 & \text{otherwise} \end{cases} \quad (7)$$

Where $f_{tr}(w)$ is the occurrence frequency of w in the training corpus, and p_0 is a constant set experimentally to 10^{-10}. We denote the tagger that employs a smoothed language model and the lexical model P_{LM_0} by the probability distribution P_{basic} (over analyses, i.e., morpheme-tag sequences).

In the unsupervised algorithm, the model P_{basic} is used to induce a *distribution of alternative analyses* (morpheme-tag sequences) for each of the sentences in the untagged corpus; we limit the number of alternative analyses per sentence to 300. This way we transform the untagged corpus into a "corpus" containing weighted analyses (i.e., morpheme-tag sequences). This corpus is then used to calculate the updated lexical model probabilities using maximum-likelihood estimation. Adding the test sentences to the untagged corpus ensures non-zero probabilities for the test words.

3.5 Implementation[4]

The set of candidate analyses was obtained from Segal's morphological analyzer (Segal, 2000). The analyzer's dictionary contains 17,544 base forms that can be inflected. After this dictionary was extended with the tagged training corpus, it recognizes 96.14% of the words in the test set.[5] For each train/test split of the corpus, we only use the training data for enhancing the dictionary. We used SRILM (Stolcke, 2002) for constructing language models, and for disambiguation.

4 Evaluation

In this section we report on an empirical comparison between the two levels of tokenization presented in the previous section. Analysis of the results leads to an improved morpheme-level model, which outperforms both of the initial models.

Each architectural configuration was evaluated in 5-fold cross-validated experiments. In a train/test

[3] the smoothed probabilities are normalized so that $\sum_w P(w, \mathbf{a}) = P(\mathbf{a})$

[4] http://www.cs.technion.ac.il/~barhaim/MorphTagger/

[5] Unrecognized words are assumed to be proper nouns, and the morphological analyzer proposes possible segmentations for the word, based on the recognition of possible prefixes.

split of the corpus, the training set includes 1,598 sentences on average, which on average amount to 28,738 words and 39,282 morphemes. The test set includes 250 sentences. We estimate *segmentation accuracy* – the percentage of words correctly segmented into morphemes, as well as *tagging accuracy* – the percentage of words that were correctly segmented for which each morpheme was assigned the correct POS tag.

For each parameter, the average over the five folds is reported, with the standard deviation in parentheses. We used two-tailed paired t-test for testing the significance of the difference between the average results of different systems. The significance level (p-value) is reported.

The first two lines in Table 1 detail the results obtained for both word (W) and morpheme (M) levels of tokenization. The tagging accuracy of the

	Accuracy per word (%)	
Tokenization	Tagging	Segmentation
W	89.42 (0.9)	96.43 (0.3)
M	90.21 (1.2)	96.25 (0.5)
M+*h*	90.51 (1.0)	96.74 (0.5)

Table 1: Level of tokenization - experimental results

morpheme tagger is considerably better than what is achieved by the word tagger (difference of 0.79% with significance level $p = 0.01$). This is in spite of the fact that the segmentation achieved by the word tagger is a little better (and a segmentation error implies incorrect tagging). Our hypothesis is that:

> *Morpheme-level taggers outperform word-level taggers in their tagging accuracy, since they suffer less from data sparseness. However, they lack some word-level knowledge that is required for segmentation.*

This hypothesis is supported by the number of once-occurring terminals in each level: 8,582 in the word level, versus 5,129 in the morpheme level.

Motivated by this hypothesis, we next consider what kind of word-level information is required for the morpheme-level tagger in order to do better in segmentation. One natural enhancement for the morpheme-level model involves adding information about word boundaries to the tag set. In the enhanced tag set, nonterminal symbols include additional features that indicate whether the tagged morpheme starts/ends a word. Unfortunately, we found that adding word boundary information in this way did not improve segmentation accuracy.

However, error analysis revealed a very common type of segmentation errors, which was found to be considerably more frequent in morpheme tagging than in word tagging. This kind of errors involves a missing or an extra covert definiteness marker *'h'*. For example, the word *bbit* can be segmented either as *b-bit* ("in a house") or as *b-h-bit* ("in the house"), pronounced *bebayit* and *babayit*, respectively. Unlike other cases of segmentation ambiguity, which often just manifest lexical facts about spelling of Hebrew stems, this kind of ambiguity is productive: it occurs whenever the stem's POS allows definiteness, and is preceded by one of the prepositions *b/k/l*. In morpheme tagging, this type of error was found on average in 1.71% of the words (46% of the segmentation errors). In word tagging, it was found only in 1.36% of the words (38% of the segmentation errors).

Since in Hebrew there should be agreement between the definiteness status of a noun and its related adjective, this kind of ambiguity can sometimes be resolved syntactically. For instance:

"bbit hgdwl" implies *b-h-bit* ("in the big house")
"bbit gdwl" implies *b-bit* ("in a big house")

By contrast, in many other cases both analyses are syntactically valid, and the choice between them requires consideration of a wider context, or some world knowledge. For example, in the sentence *hlknw lmsibh* ("we went to a/the party"), *lmsibh* can be analyzed either as *l-msibh* (indefinite,"to a party") or as *l-h-mbsibh* (definite,"to the party"). Whether we prefer "the party" or "a party" depends on contextual information that is not available for the POS tagger.

Lexical statistics can provide valuable information in such situations, since some nouns are more common in their definite form, while other nouns are more common as indefinite. For example, consider the word *lmmflh* ("to a/the government"), which can be segmented either as *l-mmflh* or *l-h-mmflh*. The

Tokenization	Analysis
W	(*lmmflh* IN-H-NN)
M	(IN *l*) (H *h*) (NN *mmflh*)
M+*h*	(IN *l*) (H-NN *hmmflh*)

Table 2: Representation of *l-h-mmflh* in each level of tokenization

stem *mmflh* ("government") was found 25 times in the corpus, out of which only two occurrences were indefinite. This strong lexical evidence in favor of *l-h-mmflh* is completely missed by the morpheme-level tagger, in which morphemes are assumed to be independent. The lexical model of the word-level tagger better models this difference, since it does take into account the frequencies of *l-mmflh* and *l-h-mmlh*, in measuring P(*lmmflh*|IN-NN) and P(*lmmflh*|IN-H-NN). However, since the word tagger considers *lmmflh*, *hmmflh* ("the government"), and *mmflh* ("a government") as independent words, it still exploits only part of the potential lexical evidence about definiteness.

In order to better model such situations, we changed the morpheme-level model as follows. In definite words the definiteness article *h* is treated as a manifestation of a morphological feature of the stem. Hence the definiteness marker's POS tag (H) is prefixed to the POS tag of the stem. We refer by *M+h* to the resulting model that uses this assumption, which is rather standard in theoretical linguistic studies of Hebrew. The M+*h* model can be viewed as an intermediate level of tokenization, between morpheme and word tokenization. The different analyses obtained by the three models of tokenization are demonstrated in Table 2.

As shown in Table 1, the M+*h* model shows remarkable improvement in segmentation (0.49%, $p < 0.001$) compared with the initial morpheme-level model (M). As expected, the frequency of segmentation errors that involve covert definiteness (*h*) dropped from 1.71% to 1.25%. The adjusted morpheme tagger also outperforms the word level tagger in segmentation (0.31%, $p = 0.069$). Tagging was improved as well (0.3%, $p = 0.068$). According to these results, tokenization as in the M+*h* model is preferable to both plain-morpheme and plain-word tokenization.

5 Conclusion

Developing a word segmenter and POS tagger for Hebrew with less than 30K annotated words for training is a challenging task, especially given the morphological complexity and high degree of ambiguity in Hebrew. For comparison, in English a baseline model that selects the most frequent POS tag achieves accuracy of around the 90% (Charniak et al., 1993). However, in Hebrew we found that a parallel baseline model achieves only 84% using the available corpus.

The architecture proposed in this paper addresses the severe sparseness problems that arise in a number of ways. First, the M+*h* model, which was found to perform best, is based on morpheme-level tokenization, which suffers of data sparseness less than word tokenization, and makes use of multi-morpheme nonterminals only in specific cases where it was found to be valuable. The number of nonterminal types found in the corpus for this model is 49 (including 11 types of punctuation marks), which is much closer to the morpheme-level model (39 types) than to the word-level model (205 types). Second, the bootstrapping method we present exploits additional resources such as a morphological analyzer and an untagged corpus, to improve lexical probabilities, which suffer from data sparseness the most. The improved lexical model contributes 1.5% to the tagging accuracy, and 0.6% to the segmentation accuracy (compared with using the basic lexical model), making it a crucial component of our system.

Among the few other tools available for POS tagging and morphological disambiguation in Hebrew, the only one that is freely available for extensive training and evaluation as performed in this paper is Segal's ((Segal, 2000), see section 2.2). Comparing our best architecture to the Segal tagger's results under the same experimental setting shows an improvement of 1.5% in segmentation accuracy and 4.5% in tagging accuracy over Segal's results.

Moving on to Arabic, in a setting comparable to (Diab et al., 2004), in which the correct segmentation is given, our tagger achieves accuracy per *morpheme* of 94.9%. This result is close to the re-

sult reported by Diab et al., although our result was achieved using a much smaller annotated corpus. We therefore believe that future work may benefit from applying our model, or variations thereof, to Arabic and other Semitic languages.

One of the main sources for tagging errors in our model is the coverage of the morphological analyzer. The analyzer misses the correct analysis of 3.78% of the test words. Hence, the upper bound for the accuracy of the disambiguator is 96.22%. Increasing the coverage while maintaining the quality of the proposed analyses (avoiding over-generation as much as possible), is crucial for improving the tagging results.

It should also be mentioned that a new version of the Hebrew treebank, now containing approximately 5,000 sentences, was released after the current work was completed. We believe that the additional annotated data will allow to refine our model, both in terms of accuracy and in terms of coverage, by expanding the tag set with additional morpho-syntactic features like gender and number, which are prevalent in Hebrew and other Semitic languages.

Acknowledgments

We thank Gilad Ben-Avi, Ido Dagan and Alon Itai for their insightful remarks on major aspects of this work. The financial and computational support of the Knowledge Center for Processing Hebrew is gratefully acknowledged. The first author would like to thank the Technion for partially funding his part of the research. The first and third authors are grateful to the ILLC of the University of Amsterdam for its hospitality while working on this research. We also thank Andreas Stolcke for his devoted technical assistance with SRILM.

References

Meni Adler. 2001. Hidden Markov Model for Hebrew part-of-speech tagging. Master's thesis, Ben Gurion University, Israel. In Hebrew.

Leonard Baum. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. In *Inequalities III:Proceedings of the Third Symposium on Inequalities, University of California, Los Angeles, pp.1-8*.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistic*, 21:784–789.

David Carmel and Yoelle Maarek. 1999. Morphological disambiguation for Hebrew search systems. In *Proceedings of the 4th international workshop,NGITS-99*.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. In *National Conference on Artificial Intelligence*, pages 784–789.

K. W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of the Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, TX.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In *HLT-NAACL 2004: Short Papers*, pages 149–152.

S.M. Katz. 1987. Estimation of probabilities from sparse data from the language model component of a speech recognizer. *IEEE Transactions of Acoustics, Speech and Signal Processing*, 35(3):400–401.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. 2003. Language model based Arabic word segmentation. In *ACL*, pages 399–406.

M. Levinger, U. Ornan, and A. Itai. 1995. Morphological disambiguation in Hebrew using a priori probabilities. *Computational Linguistics*, 21:383–404.

Moshe Levinger. 1992. Morphological disambiguation in Hebrew. Master's thesis, Computer Science Department, Technion, Haifa, Israel. In Hebrew.

Erel Segal. 2000. Hebrew morphological analyzer for Hebrew undotted texts. Master's thesis, Computer Science Department, Technion, Haifa, Israel. http://www.cs.technion.ac.il/-∼erelsgl/bxi/hmntx/teud.html.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a tree-bank of Modern Hebrew text. *Traitment Automatique des Langues*, 42:347–380.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *ICSLP*, pages 901–904, Denver, Colorado, September.

Part of Speech tagging for Amharic using Conditional Random Fields

Sisay Fissaha Adafre
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
sfissaha@science.uva.nl

Abstract

We applied Conditional Random Fields (CRFs) to the tasks of Amharic word segmentation and POS tagging using a small annotated corpus of 1000 words. Given the size of the data and the large number of unknown words in the test corpus (80%), an accuracy of 84% for Amharic word segmentation and 74% for POS tagging is encouraging, indicating the applicability of CRFs for a morphologically complex language like Amharic.

1 Introduction

Part-of-speech (POS) tagging is often considered as the first phase of a more complex natural language processing application. The task is particularly amenable to automatic processing. Specifically, POS taggers that are trained on pre-annotated corpora achieve human-like performance, which is adequate for most applications. The road to such high performance levels is, however, filled with a hierarchy of sub-problems. Most techniques generally assume the availability of large POS annotated corpora. The development of annotated corpora in turn requires a standard POS tagset. None of these resources are available for Amharic. This is due mainly to the fact that data preparation, i.e., developing a comprehensive POS tagset and annotating a reasonably sized text, is an arduous task. Although the POS tagging task, taken as a whole, seems challenging, a lot can be gained by analyzing it into subproblems and dealing with each one step-by-step, and also bringing in the experience from other languages in solving these problems, since POS taggers have been developed for several languages resulting in a rich body of knowledge.

Several attempts have been made in the past to develop algorithms for analyzing Amharic words. Among these is the stemming algorithm of Nega (1999), which reduces Amharic words into their common stem forms by removing affixes. Nega's work focuses on investigating the effectiveness of the stemming algorithm in information retrieval for Amharic. Abyot (2000) developed a word parser for Amharic verbs that analyses verbs into their constituting morphemes and determines their morphosyntactic categories. Abyot's work only covers verbs and their derivations. Mesfin (2001) developed a Hidden Markov Model (HMM) based part of speech tagger for Amharic. Building on the work of Mesfin, Atelach (2002) developed a stochastic syntactic parser for Amharic. Sisay and Haller (2003a; 2003b) applied finite-state tools, and corpus-based methods for the Amharic morphological analysis. This work provided important insights into the issues surrounding the development of Amharic natural language processing applications, especially, in compiling a preliminary POS tagset for Amharic.

In this paper, our aim is to explore recent developments in the morphological analysis of related languages, such as Arabic and Hebrew, and machine learning approaches, and apply them to the Amharic language. Amharic belongs to the Semitic family of languages, and hence shares a number of common morphological properties with Arabic and Hebrew for which active research is being carried out. Stud-

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 47–54,
Ann Arbor, June 2005.

ies on these languages propose two alternative POS tagging approaches which differ on the unit of analysis chosen; morpheme-based and word-based (Bar-Haim et al., 2004). The former presupposes a segmentation phase in which words are analysed into constituting morphemes which are then passed to the POS tagging step, whereas the latter applies POS tagging directly on fully-inflected word forms. Due to scarce resources, it is impossible for us to fully carry out these tasks for Amharic. Therefore, the segmentation and POS tagging tasks are carried out independently. Furthermore, POS tagging is applied only on fully-inflected word forms. The motivation for doing the segmentation task comes from the need to provide some measure of the complexity of the task in the context of the Amharic language. As regards implementation, new models have been introduced recently for segmentation and sequence-labeling tasks. One such model is Conditional Random Fields (CRFs) (Lafferty et al., 2001). In this paper, we describe important morphosyntactic characteristics of Amharic, and apply CRFs to Amharic word segmentation and POS tagging.

The paper is organized as follows. Section 2 provides a brief description of Amharic morphology. Section 3 presents some of the work done in the area of Amharic morphological analysis, and examines one POS tagset proposed by previous studies. This tagset has been revised and applied on a sample Amharic newspaper text, which is discussed in Section 4. Section 5 describes the tasks in greater detail. Section 6 provides a brief description of CRFs, the machine learning algorithm that will be applied in this paper. Section 7 describes the experimental setup and Section 8 presents the result of the experiment. Finally, Section 9 makes some concluding remarks.

2 Amharic Morphology

Amharic is one of the most widely spoken languages in Ethiopia. It has its own script that is borrowed from Ge'ez, another Ethiopian Semitic language (Leslau, 1995). The script is believed to have originated from the South Sabean script. It is a syllabary writing system where each character represents an open CV syllable, i.e., a combination of a consonant followed by a vowel (Daniels, 1997).

Amharic has a complex morphology. Word formation involves prefixation, suffixation, infixation, reduplication, and Semitic stem interdigitation, among others. Like other Semitic languages, e.g., Arabic, Amharic verbs and their derivations constitute a significant part of the lexicon. In Semitic languages, words, especially verbs, are best viewed as consisting of discontinuous morphemes that are combined in a non-concatenative manner. Put differently, verbs are commonly analyzed as consisting of root consonants, template patterns, and vowel patterns. With the exception of very few verb forms (such as the imperative), all derived verb forms take affixes in order to appear as independent words.

Most function words in Amharic, such as Conjunction, Preposition, Article, Relative marker, Pronominal affixes, Negation markers, are bound morphemes, which are attached to content words, resulting in complex Amharic words composed of several morphemes. Nouns inflect for the morphosyntactic features number, gender, definiteness, and case. Amharic adjectives share some morphological properties with nouns, such as definiteness, case, and number. As compared to nouns and verbs, there are fewer primary adjectives. Most adjectives are derived from nouns or verbs. Amharic has very few lexical adverbs. Adverbial meaning is usually expressed morphologically on the verb or through prepositional phrases. While prepositions are mostly bound morphemes, postpositions are typically independent words.

The segmentation task (cf. Section 7.1) considers the following bound morphemes as segments: Prepositions, Conjunctions, Relative Makers, Auxiliary verbs, Negation Marker and Coordinate Conjunction. Other bound morphemes such as definite article, agreement features (i.e., number, gender), case markers, etc are not considered as segments and will be treated as part of the word. These are chosen since they are commonly treated as separate units in most syntactic descriptions.

Although the above description of Amharic is far from complete, it highlights some of the major characteristics of Amharic, which it shares with other Semitic languages such as Arabic. It is, therefore, worthwhile to take into consideration the work done for other Semitic languages in proposing a method for Amharic natural language processing.

3 Amharic POS Tagset

Mesfin (2001) compiled a total of 25 POS tags: N, NV, NB, NP, NC, V, AUX, VCO, VP, VC, J, JC, JNU, JPN, JP, PREP, ADV, ADVC, C, REL, ITJ, ORD, CRD, PUNC, and UNC. These tags capture important properties of the language at a higher level of description. For example, the fact that there is no category for Articles indicates that Amharic does not have independent lexical forms for articles. However, a close examination of the description of some of the tags reveals some missclassification that we think will lead to tagging inconsistency. For example, the tag JPN is assigned to nouns with the "ye" prefix morpheme that function as an adjective, e.g. `yetaywan sahn - A Taiwan made plate` (Mesfin, 2001). This example shows that grammatical function takes precedence over morphological form in deciding the POS category of a word. In Amharic, the ye+NOUN construction can also be used to represent other kinds of relation such as Possession relation. On the other hand, the ye+NOUN construction is a simple morphological variant of the NOUN that can easily be recognized. Therefore, treating ye+NOUN construction as a subclass of a major noun class will result in a better tagging consistency than treating it as an adjective. Furthermore, a hierarchical tagset, organized into major classes and subclasses, seems to be a preferred design strategy (Wilson, 1996; Khoja et al., 2001). Although it is possible to guess (from the tagset description) some abstract classes such as, N* (nouns), V* (verbs), J* (adjectives), etc., such a hierarchical relation is not clearly indicated. One advantage of such a hierarchical organization is that it allows one to work at different levels of abstraction.

The POS tags that are used in this paper are obtained by collapsing some of the categories proposed by Mesfin (2001). The POS tags are Noun (N), Verb (V), Auxiliary verbs (AUX), Numerals (NU), Adjective (AJ), Adverb (AV), Adposition (AP), Interjection (I), Residual (R), and Punctuation (PU). The main reason for working with a set of abstract POS tags is resource limitation, i.e., the absence of a large annotated corpus. Since we are working on a small annotated corpus, 25 POS tags make the data sparse and the results unreliable. Therefore, we have found it necessary to revise the tagset.

4 Application of the Revised Tagset

The above abstract POS tags are chosen by taking into account the proposals made in Amharic grammar literature and the guidelines of other languages (Baye, 1986; Wilson, 1996; Khoja et al., 2001). It is, however, necessary to apply the revised tagset to a real Amharic text and see if it leads to any unforeseeable problems. It is also useful to see the distribution of POS tags in a typical Amahric newspaper text. Therefore, we selected 5 Amharic news articles and applied the above tagset.

All the tokens in the corpus are assigned one of the tags in the proposed tagset relatively easily. There do not seem to be any gaps in the tagset. Unlike Mesfin (2001), who assigns collocations a single POS tag, we have assumed that each token should be treated separately. This means that words that are part of a collocation are assigned tags individually. This in turn contributes towards a better tagging consistency by minimizing context dependent decision-making steps.

Table 1 shows the distribution of POS tags in the corpus. Nouns constitute the largest POS category in the corpus based on the above tagging scheme. This seems to be characteristic of other languages too. However, Amharic makes extensive use of noun clauses for representing different kinds of subordinate clauses. Noun clauses are headed by a verbal noun, which is assigned a noun POS tag. This adds to the skewedness of POS tag distributions which in turn biases the POS tagger that relies heavily on morphological features as we will show in Section 7. Interjections, on the other hand, do not occur in the sample corpus, as these words usually do not appear often in newspaper text.

Once the POS tagset has been compiled and tested, the next logical step is to explore automatic methods of analyzing Amharic words, which we explore in the next section.

5 POS Tagging of Amharic

Semitic languages like Arabic, Hebrew and Amharic have a much more complex morphology than English. In these languages, words usually consist of several bound morphemes that would normally have independent lexical entries in languages like English. Furthermore, in Arabic and Hebrew, the

Description	POS tag	Frequency
Noun	N	586
Verb	V	203
Auxiliary	AUX	20
Numeral	NU	65
Adjective	AJ	31
Adverb	AV	8
Adposition	AP	30
Interjection	I	0
Punctuation	PU	36
Residual	R	15

Table 1: Distribution of POS tags

diacritics that represent most vowels and gemination patterns are missing in written texts. Although Amharic does not have a special marker for gemination, the Amharic script fully encodes both the vowels and the consonants, hence it does not suffer from the ambiguity problem that may arise due to the missing vowels.

As mentioned briefly in Section 1, the morphological complexity of these languages opens up different alternative approaches in developing POS taggers for them (Bar-Haim et al., 2004; Diab et al., 2004). Bar-Haim et al. (2004) showed that morpheme-based tagging performs better than word-based tagging; they used Hidden Markov Models (HMMs) for developing the tagger.

On the basis of the idea introduced by Bar-Haim et al. (2004), we formulate the following two related tasks for the analysis of Amharic words: segmentation and POS tagging (sequence labeling). Segmentation refers to the analysis of a word into constituting morphemes. The POS tagging task, on the other hand, deals with the assignment of POS tags to words. The revised POS tags that are introduced in Section 3 will be used for this task. The main reason for choosing words as a unit of analysis and adopting the abstract POS tags is that the limited resource that we have prohibits us from carrying out fine-grained classification experiments. As a result of this, we choose to aim at a less ambitious goal of investigating to what extent the strategies used for unknown word recognitions can help fill the gap left by scarce resources. Therefore, we mainly focus on word-based tagging and explore different kinds of features that contribute to tagging accuracy.

Although the segmentation and POS tagging tasks look different, both can be reduced to sequence labeling tasks. Since the size of the annotated corpora is very small, a method needs to be chosen that allows an optimal utilization of the limited resources that are available for Amharic. In this respect, CRFs are more appropriate than HMMs since they allow us to integrate information from different sources (Lafferty et al., 2001). In the next section, we provide a brief description of CRFs.

6 Conditional Random Fields

Conditional Random Fields are conditional probability distributions that take the form of exponential models. A special case of CRFs, linear chain CRF, which takes the following form, has been widely used for sequence labeling tasks.

$$P\left(y \mid x\right) = \frac{1}{Z\left(x\right)} exp\left(\sum_{t=1}\sum_{k} \lambda_k f_k\left(t, y_{t-1}, y_t, x\right)\right),$$

where $Z\left(x\right)$ is the normalization factor, $X = \{x_1, \ldots, x_n\}$ is the observation sequence, $Y = \{y_1, \ldots, y_T\}$ is the label sequences, f_k and λ_k are the feature functions and their corresponding weights respectively (Lafferty et al., 2001).

An important property of these models is that probabilities are computed based on a set of feature functions, i.e. f_k, (usually binary valued), which are defined on both the observation X and label sequences Y. These feature functions describe different aspect of the data and may overlap, providing a flexible way of describing the task. CRFs have been shown to perform well in a number of natural language processing applications, such as POS tagging (Lafferty et al., 2001), shallow parsing or NP chunking (Sha and Pereira, 2003), and named entity recognition (McCallum and Li, 2003).

In POS tagging, context information such as surrounding words and their morphological features, i.e., suffixes and prefixes, significantly improves performance. CRFs allow us to integrate large set of such features easily. Therefore, it would be interesting to see to what extent the morphological features help in predicting Amharic POS tags. We used the minorThird implementation of CRF (Cohen, 2004).

7 Experiments

There are limited resources for the Amharic language, which can be used for developing POS tagger. One resource that may be relevant for the current task is a dictionary consisting of some 15,000 entries (Amsalu, 1987). Each entry is assigned one of the five POS tags; Noun, Verb, Adjectives, Adverb, and Adposition. Due to the morphological complexity of the language, a fully inflected dictionary consisting only of 15,000 entries is bound to have limited coverage. Furthermore, the dictionary contains entries for phrases, which do not fall into any of the POS categories. Therefore the actual number of useful entries is a lot less than 15,000.

The data for the experiment that will be described below consists of 5 annotated news articles (1000 words). The Amharic text has been transliterated using the SERA transliteration scheme, which encodes Amharic scripts using Latin alphabets (Daniel, 1996). This data is very small compared to the data used in other segmentation and POS tagging experiments. However, it is worthwhile to investigate how such a limited resource can meaningfully be used for tackling the aforementioned tasks.

7.1 Segmentation

The training data for segmentation task consists of 5 news articles in which the words are annotated with segment boundaries as shown in the following example.

```
...<seg>Ind</seg><seg>
astawequt</seg>#
<seg>le</seg><seg>arso
</seg>#<seg> aderu
</seg># <seg>be</seg>
<seg>temeTaTaN</seg>...
```

In this example, the morphemes are enclosed in `<seg>` and `</seg>` XML tags. Word-boundaries are indicated using the special symbol #. The reduction of the segmentation task to a sequence labeling task is achieved by converting the XML-annotated text into a sequence of character-tag pairs. Each character constitutes a training (test) instance. The following five tags are used for tagging the characters; B(egin), C(ontinue), E(nd), U(nique) and N(egative). Each character in the segment is assigned one of these tags depending on where it appears in the segment; at the beginning (B), at the end (E), inside (C), or alone (U). While the tags BCE are used to capture multi-character morphemes, the U tag is used to represent single-character morphemes. The negative tag (N) is assigned to the special symbol # used to indicate the word boundary. Though experiments have been carried out with less elaborate tagging schemes such as BIO (Begin-Inside-Outside), no significant performance improvement has been observed. Therefore, results are reported only for the BCEUN tagging scheme.

The set of features that are used for training are composed of character features, morphological features, dictionary features, the previous tag, and character bi-grams. We used a window of eleven characters centered at the current character. The character features consist of the current character, the five characters to the left and to the right of the current characters. Morphological features are generated by first merging the set of characters that appear between the word boundaries (both left and right) and the current character. Then a binary feature will be generated in which its value depends on whether the resulting segment appears in a precompiled list of valid prefix and suffix morphemes or not. The same segment is also used to generate another dictionary-based feature, i.e., it is checked whether it exists in the dictionary. Character bi-grams that appear to the left and the right of the current character are also used as features. Finally, the previous tag is also used as a feature.

7.2 POS Tagging

The experimental setup for POS tagging is similar to that of the segmentation task. However, in our current experiments, words, instead of characters, are annotated with their POS tags and hence we have more labels now. The following example shows the annotation used in the training data.

```
...<V>yemikahEdut</V>
<N>yemrmr</N>
<N>tegbarat</N>
<V>yatekorut</V>
<N>bemgb</N> <N>sebl</N>
...
```

Each word is enclosed in an XML tag that denotes its POS tag. These tags are directly used for the training of the sequence-labeling task. No additional reduction process is carried out.

The set of features that are used for training are composed of lexical features, morphological features, dictionary features, the previous two POS tags, and character bi-grams. We used a window of five words centered at the current word. The lexical features consist of the current word, the two words to the left and to the right of the current word. Morphological features are generated by extracting a segment of length one to four characters long from the beginning and end of the word. These segments are first checked against a precompiled list of valid prefix and suffix morphemes of the language. If the segment is a valid morpheme then an appropriate feature will be generated. Otherwise the null prefix or suffix feature will be generated to indicate the absence of an affix. The dictionary is used to generate a binary feature for a word based on the POS tag found in the dictionary. In other words, if the word is found in the dictionary, its POS tag will be used as a feature. For each word, a set of character bi-grams has been generated and each character bigram is used as a feature. Finally, the last two POS tags are also used as a feature.

8 Results

We conducted a 5-fold cross-validation experiment. In each run, one article is used as a test dataset and the remaining four articles are used for training. The results reported in the sections below are the average of these five runs. On average 80% of the words in the test files are unknown words. Most of the unknown words (on average 60%) are nouns.

8.1 Segmentation Result

As mentioned in Section 7.1, four sets of features, i.e., character features, morphological features, dictionary features, and previous label, are used for the segmentation task. Table 2 shows results for some combinations of these features. The results without the previous label feature are also shown (*Without Prev. Label*).

The simple character features are highly informative features, as can be seen in Table 2 (Row 1). Using only these features, the system with previous label feature already achieved an accuracy of 0.819. The dictionary feature improved the result by 2% whereas the morphological features brought minor improvements. As more features are added the variation between the different runs increases slightly. Performace significantly decreases when we omit the previous label feature as it is shown in *Without Prev. Label* column.

8.2 POS Tagging Results

Table 3 shows the word-based evaluation results of the POS tagging experiment. The baseline (Row 1) means assigning all the words the most frequently occurring POS tag, i.e., N (noun). The result obtained using only lexical features (Row 2) is better than the baseline. Adding morphological features improves the result almost by the same amount (Row 3). Incorporation of the dictionary feature, however, has brought only slight improvement. The addition of bi-gram features improved the result by 3%.

As mentioned before, it is not possible to compare the results, i.e. 74% accuracy (With Prev. Label), with other state of the art POS taggers since our data is very small compared to the data used by other POS taggers. It is also difficult to claim with absolute certainty as to the applicability of the technique we have applied. However, given the fact that 80% of the test instances are unseen instances, an accuracy of 74% is an acceptable result. This claim receives further support when we look at the results reported for unknown word guessing methods in other POS tagging experiments (Nakagawa et al., 2001). As we add more features, the system shows less variation among the different folds. As with segmentation task, the omission of the previous label feature decreases performace. The system with only lexical features and without previous label feature has the same performace as the baseline system.

8.3 Error Analysis

The results of both the segmentation and POS tagging tasks show that they are not perfect. An examination of the output of these systems shows certain patterns of errors. In case of the segmentation task, most of the words that are incorrectly segmented have the same beginning or ending charac-

	With Prev. Label		Without Prev. Label	
Features	accuracy	stddev	accuracy	stddev
Char.	0.819	0.7	0.661	4.7
Char.+Dict.	0.837	1.6	0.671	4.1
Char.+Dict.+Morph.	0.841	1.7	0.701	3.9

Table 2: Segmentation Results

	With Prev. Label		Without Prev. Label	
Features	accuracy	stddev	accuracy	stddev
Baseline	0.513	6.4	–	–
Lex.	0.613	5.3	0.513	6.4
Lex.+Morph.	0.700	5.0	0.688	5.2
Lex.+Morph.+Dict.	0.713	4.3	0.674	5.6
Lex.+Morph.+Dict.+Bigram	0.748	4.3	0.720	2.9

Table 3: Word-based evaluation results of POS tagging

ters as words with affix morphemes. Increasing the size of the lexical resources, such as the dictionary, can help the system in distinguishing between words that have affixes from those that do not.

The POS tagging system, on the other hand, has difficulties in distinguishing between nouns and other POS tags. This in turn shows how similar nouns are to words in other POS tags morphologically, since our experiment relies heavily on morphological features. This is not particularly surprising given that most Amharic affixes are shared among nouns and words in other POS tags. In Amharic, if a noun phrase contains only the head noun, most noun affixes, such as prepositions, definite article, and case marker appear on the head noun. If, on the other hand, a noun phrase contains prenominal constituents such as adjectives, numerals, and other nouns, then the above noun affixes appear on prenominal constituents, thereby blurring the morphological distinction between the nouns and other constituents. Furthermore, similar sets of morphemes are used for prepositions and subordinate conjunctions, which again obscures the distinction among the nouns and verbs. This, together with the fact that nouns are the dominant POS category in the data, resulted in most words being missclassified as nouns.

In general, we believe that the above problems can be alleviated by making more training data available to the system, which will enable us to determine improved parameters for both segmentation and POS tagging models.

9 Concluding Remarks

In this paper, we provided preliminary results of the application of CRFs for Amharic word segmentation and POS tagging tasks. Several features were examined for these tasks. Character features were found to be useful for the segmentation task whereas morphological and lexical features significantly improve the results of the POS tagging task. Dictionary-based features contribute more to the segmentation task than to the POS tagging task. In both experiments, omition of previous label feature hurts performance.

Although the size of the data limits the scope of the claims that can be made on the basis of the results, the results are good especially when we look at them from the perspective of the results achieved in unknown word recognition methods of POS tagging experiments. These results could be achieved since CRFs allow us to integrate several overlapping features thereby enabling optimum utilization of the available information.

In general, the paper dealt with a restricted aspect of the morphological analysis of Amharic, i.e., Amharic word segmentation and POS tagging. Furthermore, these tasks were carried out relatively independently. Future work should explore how these tasks could be integrated into a single system that

allows for fine-grained POS tagging of Amharic words. Parallel to this, resource development needs to be given due attention. As mentioned, the lack of adequate resources such as a large POS annotated corpus imposes restrictions on the kind of methods that can be applied. Therefore, the development of a standard Amharic POS tagset and annotation of a reasonably sized corpus should be given priority.

Acknowledgements

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001.

References

Nega Alemayehu. 1999. *Development of stemming algorithm for Amharic text retrieval*. PhD Thesis, University of Sheffield.

Atelach Alemu. 2002. *Automatic Sentence Parsing for Amharic Text: An Experiment using Probabilistic Context Free Grammars*. Master Thesis, Addis Ababa University.

Amsalu Aklilu. 1987. *Amharic-English Dictionary*. Kuraz Publishing Agency.

Roy Bar-Haim, Khalil Simaan and Yoad Winter. 2004. Part-of-Speech Tagging for Hebrew and Other Semitic Languages. Technical Report.

Abiyot Bayou. 2000. *Developing automatic word parser for Amharic verbs and their derivation*. Master Thesis, Addis Ababa University.

W. Cohen. 2004. *Methods for Identifying Names and Ontological Relations in Text using Heuristics for Inducing Regularities from Data.* `http://minorthird.sourceforge.net`

Peter T. Daniels, 1997. Script of Semitic Languages in: Robert Hetzron, editor,*Proceedings of the Corpus Linguistics.* 16–45.

Mona Diab, Kadri Hacioglu and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From row text to base phrase chunks. In Daniel Marku, Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Short papers*, pages 149–152, Boston, Massachusetts, USA, May 2–May 7. Association for Computational Linguistics

Sisay Fissaha Adafre and Johann Haller. 2003a. Amharic verb lexicon in the context of machine translation. *Traitement Automatique des Langues Naturelles* 2:183–192

Sisay Fissaha Adafre and Johann Haller. 2003b. Application of corpus-based techniques to Amharic texts. *Machine Translation for Semitic languages* MT Summit IX Workshop, New Orleans

Mesfin Getachew. 2001. *Automatic part of speech tagging for Amharic language: An experiment using stochastic HMM*. Master Thesis, Addis Ababa University.

Wolf Leslau. 1995. *Reference Grammar of Amharic*. Otto Harrassowitz, Wiesbaden.

A. McCallum and W. Li. 2003. Early results for Named Entity Recognition with conditional random fields, feature induction and web-enhanced lexicons. *Proceedings of the 7th CoNLL.*

Tetsuji Nakagawa, Taku Kudo and Yuji Matsumoto. 2001. Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines. NLPRS pages 325-331, Boston, Massachusetts, USA, May 2–May 7. `http://www.afnlp.org/nlprs2001/pdf/0053-01.pdf`

J. Lafferty, F. Pereira and A. McCallum. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the International Conference on Machine Learning.*

F. Sha and F. Pereira. 2004. Shallow parsing with conditional random fields. *Proceedings of Human Language Technology-NAACL.*

Khoja S., Garside R., and Knowles G. 2001. A Tagset for the Morphosyntactic Tagging of Arabic *Proceedings of the Corpus Linguistics.* Lancaster University (UK), Volume 13 - Special issue, 341.

Leech G. Wilson. 1996. *Recommendations for the Morphosyntactic Annotation of Corpora.* EAGLES Report EAG-TCWG-MAC/R, `http://www.ilc.pi.cnr.it/EAGLES96/annotate/annotate.html`

Daniel Yacob. 1996. *System for Ethiopic Representation in ASCII.* `http://www.abyssiniagateway.net/fidel`

Baye Yimam. 1999. Root. *Ethiopian Journal of Language Studies*, 9:56–88.

Baye Yimam. 1986. *Yamara Swasw* E.M.P.D.A, Addis Ababa.

POS Tagging of Dialectal Arabic: A Minimally Supervised Approach

Kevin Duh and Katrin Kirchhoff
Department of Electrical Engineering
University of Washington, Seattle, WA, 98195
{duh,katrin}@ee.washington.edu

Abstract

Natural language processing technology for the dialects of Arabic is still in its infancy, due to the problem of obtaining large amounts of text data for spoken Arabic. In this paper we describe the development of a part-of-speech (POS) tagger for Egyptian Colloquial Arabic. We adopt a minimally supervised approach that only requires raw text data from several varieties of Arabic and a morphological analyzer for Modern Standard Arabic. No dialect-specific tools are used. We present several statistical modeling and cross-dialectal data sharing techniques to enhance the performance of the baseline tagger and compare the results to those obtained by a supervised tagger trained on hand-annotated data and, by a state-of-the-art Modern Standard Arabic tagger applied to Egyptian Arabic.

1 Introduction

Part-of-speech (POS) tagging is a core natural language processing task that can benefit a wide range of downstream processing applications. Tagging is often the first step towards parsing or chunking (Osborne, 2000; Koeling, 2000), and knowledge of POS tags can benefit statistical language models for speech recognition or machine translation (Heeman, 1998; Vergyri et al., 2004). Many approaches for POS tagging have been developed in the past, including rule-based tagging (Brill, 1995), HMM taggers (Brants, 2000; Cutting and others, 1992), maximum-entropy models (Rathnaparki, 1996), cyclic dependency networks (Toutanova et al., 2003), memory-based learning (Daelemans et al., 1996), etc. All of these approaches require either a large amount of annotated training data (for supervised tagging) or a lexicon listing all possible tags for each word (for unsupervised tagging). Taggers have been developed for a variety of languages, including Modern Standard Arabic (MSA) (Khoja, 2001; Diab et al., 2004). Since large amount of text material as well as standard lexicons can be obtained in these cases, POS tagging is a straightforward task.

The dialects of Arabic, by contrast, are spoken rather than written languages. Apart from small amounts of written dialectal material in e.g. plays, novels, chat rooms, etc., data can only be obtained by recording and manually transcribing actual conversations. Moreover, there is no universally agreed upon writing standard for dialects (though several standardization efforts are underway); any large-scale data collection and transcription effort therefore requires extensive training of annotators to ensure consistency. Due to this data acquisition bottleneck, the development of NLP technology for dialectal Arabic is still in its infancy. In addition to the problems of sparse training data and lack of writing standards, tagging of dialectal Arabic is difficult for the following reasons:

- Resources such as lexicons, morphological analyzers, tokenizers, etc. are scarce or non-existent for dialectal Arabic.
- Dialectal Arabic is a spoken language. Tagging spoken language is typically harder than tag-

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 55–62,
Ann Arbor, June 2005.

ging written language, due to the effect of disfluencies, incomplete sentences, varied word order, etc.

- The rich morphology of Arabic leads to a large number of possible word forms, which increases the number of out-of-vocabulary (OOV) words.
- The lack of short vowel information results in high lexical ambiguity.

In this paper we present an attempt at developing a POS tagger for dialectal Arabic in a minimally supervised way. Our goal is to utilize existing resources and data for several varieties of Arabic in combination with unsupervised learning algorithms, rather than developing dialect-specific tools. The resources available to us are the CallHome Egyptian Colloquial Arabic (ECA) corpus, the LDC Levantine Arabic (LCA) corpus, the LDC MSA Treebank corpus, and a generally available morphological analysis tool (the LDC-distributed Buckwalter stemmer) for MSA. The target dialect is ECA, since this is the only dialectal corpus for which POS annotations are available. Our general approach is to bootstrap the tagger in an unsupervised way using POS information from the morphological analyzer, and to subsequently improve it by integrating additional data from other dialects and by general machine learning techniques. We compare the result against the performance of a tagger trained in a supervised way and an unsupervised tagger with a hand-developed ECA lexicon.

2 Data

The ECA corpus consists of telephone conversations between family members or close friends, with one speaker being located in the U.S. and the other in Egypt. We use the combined train, eval96 and hub5 sections of the corpus for training, the dev set for development and the eval97 set for evaluation. The LCA data consists of telephone conversations on pre-defined topics between Levantine speakers previously unknown to each other; all of the available data was used. The Treebank corpus is a collection of MSA newswire text from Agence France Press, An Nahar News, and Unmah Press. We use parts 1 (v3.0), 2 (v2.0) and 3 (v1.0). The sizes of the various corpora are shown in Table 1.

The ECA corpus was originally transcribed in a "romanized" form; a script representation was then derived automatically from the romanized transcripts. The script, therefore, does not entirely conform to the MSA standard: romanized forms may represent actual pronunciations and contain such MSA $\rightarrow$ ECA changes as /θ/ $\rightarrow$ /s/ or /t/ and /ð/ $\rightarrow$ /z/ or /d/. The resulting representation cannot be unambiguously mapped back to MSA script; the variants /s/ or /t/, for instance, are represented by س or ت, rather than ث. This introduces additional noise into the data, but it also mimics the real-world situation of variable spelling standards that need to be handled by a robust NLP system. We use the script representation of this corpus for all our experiments. The ECA corpus is accompanied by a lexicon containing the morphological analysis of all words, i.e. an analysis in terms of stem and morphological characteristics such as person, number, gender, POS, etc. Since the analysis is based on the romanized form, a single tag can be assigned to the majority of words (75% of all tokens) in the corpus. We use this assignment as the reference annotation for our experiments to evaluate the output of our tagger. The remaining 25% tokens (ambiguous words) have 2 or more tags in the lexicon and are thus ignored during evaluation since the correct reference tag cannot be determined.

Both the LCA and the MSA Treebank data sets were transcribed in standard MSA script. The LCA corpus only consists of raw orthographic transcriptions; no further annotations exist for this data set. Each word in the Treebank corpus is associated with all of its possible POS tags; the correct tag has been marked manually. We use the undecomposed word forms rather than the forms resulting from splitting off conjunctions, prepositions, and other clitics. Although improved tokenization can be expected to result in better tagging performance, tokenization tools for dialectal Arabic are currently not available, and our goal was to create comparable conditions for tagging across all of our data sets. Preprocessing of the data thus only included removing punctuation from the MSA data and removing word fragments from the spoken language corpora. Other disfluencies (fillers and repetitions) were retained since they are likely to have predictive value. Finally, singleton words (e.g. inconsistent spellings) were removed

from the LCA data. The properties of the different data sets (number of words, n-grams, and ambiguous words) are displayed in Table 1.

	ECA			LCA	MSA
	train	dev	test		
sentences	25k	6k	2.7k	114k	20k
# tokens	156k	31k	12k	476k	552k
# types	15k	5k	1.5k	16k	65k
# bigrams	81k	20k	7k	180k	336k
# trigrams	125k	26k	10k	320k	458k
% ambig.	24.4	27.8	28.2	—	—

Table 1: Corpus statistics for ECA, LCA and MSA.

The only resource we utilize in addition to raw data is the LDC-distributed Buckwalter stemmer. The stemmer analyzes MSA script forms and outputs all possible morphological analyses (stems and POS tags, as well as diacritizations) for the word. The analysis is based on an internal stem lexicon combined with rules for affixation. Although the stemmer was developed primarily for MSA, it can accommodate a certain percentage of dialectal words. Table 2 shows the percentages of word types and tokens in the ECA and LCA corpora that received an analysis from the Buckwalter stemmer. Since both sets contain dialectal as well as standard MSA forms, it is not possible to determine precisely how many of the unanalyzable forms are dialectal forms vs. words that were rejected for other reasons, such as misspellings. The higher percentage of rejected word types in the ECA corpus is most likely due to the non-standard script forms described above.

	Type		Token	
N	ECA	LCA	ECA	LCA
0	37.6	23.3	18.2	28.2
1	34.0	52.5	33.6	40.4
2	19.4	17.7	26.4	19.9
3	7.2	5.2	16.2	10.5
4	1.4	1.0	5.0	2.3
5	0.1	0.1	0.4	0.6

Table 2: Percentage of word types/tokens with N possible tags, as determined by the Buckwalter stemmer. Words with 0 tags are unanalyzable.

The POS tags used in the LDC ECA annotation and in the Buckwalter stemmer are rather fine-grained; furthermore, they are not identical. We therefore mapped both sets of tags to a unified, simpler tagset consisting only of the major POS categories listed in Table 2. The mapping from the original Buckwalter tag to the simplified set was based on the conversion scheme suggested in (Bies, 2003). The same underlying conversion rules were applicable to most of the LDC tags; those cases that could not be determined automatically were converted by hand.

Symbol	Gloss	(%)
CC	coordinating conjunction	7.15
DT	determiner	2.23
FOR	foreign word	1.18
IN	preposition	7.46
JJ	adjective	6.02
NN	noun	19.95
NNP	proper noun	3.55
NNS	non-singular noun	3.04
NOTAG	non-word	0.05
PRP	pronoun	5.85
RB	adverb	4.13
RP	particle	9.71
UH	disfluency, interjection	9.55
VBD	perfect verb	6.53
VBN	passive verb	1.88
VBP	imperfect verb	10.62
WP	relative pronoun	1.08

Table 3: Collapsed tagset and percentage of occurrence of each tag in the ECA corpus.

Prior to the development of our tagger we computed the cross-corpus coverage of word n-grams in the ECA development set, in order to verify our assumption that utilizing data from other dialects might be helpful. Table 4 demonstrates that the n-gram coverage of the ECA development set increases slightly by adding LCA and MSA data.

	Types			Tokens		
	1gr	2gr	3gr	1gr	2gr	3gr
ECA	64	33	12	94	58	22
LCA	31	9	1.4	69	20	3.7
ECA + LCA	**68**	**35**	**13**	**95**	**60**	**23**
MSA	32	3.7	0.2	66	8.6	0.3
ECA + MSA	**71**	**34**	12	**95**	**60**	22

Table 4: Percentages of n-gram types and tokens in ECA dev set that are covered by the ECA training set, the LCA set, combined ECA training + LCA set, and MSA sets. Note that adding the LCA or MSA improves the coverage slightly.

3 Baseline Tagger

We use a statistical trigram tagger in the form of a hidden Markov model (HMM). Let $w_{0:M}$ be a sequence of words $(w_0, w_1, \ldots, w_M)$ and $t_{0:M}$ be the corresponding sequence of tags. The trigram HMM computes the conditional probability of the word and tag sequence $p(w_{0:M}, t_{0:M})$ as:

$$P(t_{0:M}|w_{0:M}) = \prod_{i=0}^{M} p(w_i|t_i)p(t_i|t_{i-1}, t_{i-2}) \quad (1)$$

The **lexical model** $p(w_i|t_i)$ characterizes the distribution of words for a specific tag; the **contextual model** $p(t_i|t_{i-1}, t_{i-2})$ is trigram model over the tag sequence. For notational simplicity, in subsequent sections we will denote $p(t_i|t_{i-1}, t_{i-2})$ as $p(t_i|h_i)$, where h_i represents the tag history. The HMM is trained to maximize the likelihood of the training data. In supervised training, both tag and word sequences are observed, so the maximum likelihood estimate is obtained by relative frequency counting, and, possibly, smoothing. During unsupervised training, the tag sequences are hidden, and the Expectation-Maximization Algorithm is used to iteratively update probabilities based on expected counts. Unsupervised tagging requires a lexicon specifying the set of possible tags for each word. Given a test sentence $w'_{0:M}$, the Viterbi algorithm is used to find the tag sequence maximizing the probability of tags given words: $t^*_{0:M} = argmax_{t_{0:M}} p(t_{0:M}|w'_{0:M})$. Our taggers are implemented using the Graphical Models Toolkit (GMTK) (Bilmes and Zweig, 2002), which allows training a wide range of probabilistic models with both hidden and observed variables.

As a first step, we compare the performance of four different baseline systems on our ECA dev set. First, we trained a supervised tagger on the MSA treebank corpus (System I), in order to gauge how a standard system trained on written Arabic performs on dialectal speech. The second system (System II) is a supervised tagger trained on the manual ECA POS annotations. System III is an unsupervised tagger on the ECA training set. The lexicon for this system is derived from the reference annotations of the training set – thus, the correct tag is not known during training, but the lexicon is constrained by expert information. The difference in accuracy between Systems II and III indicates the loss due to the unsupervised training method. Finally, we trained a system using only the unannotated ECA data and a lexicon generated by applying the MSA analyzer to the training corpus and collecting all resulting tags for each word. In this case, the lexicon is much less constrained; moreover, many words do not receive an output from the stemmer at all. This is the training method with the least amount of supervision and therefore the method we are interested in most.

Table 5 shows the accuracies of the four systems on the ECA development set. Also included is a breakdown of accuracy by analyzable (AW), unanalyzable (UW), and out-of-vocabulary (OOV) words. Analyzable words are those for which the stemmer outputs possible analyses; unanalyzable words cannot be processed by the stemmer. The percentage of unanalyzable word tokens in the dev set is 18.8%. The MSA-trained tagger (System I) achieves an accuracy of 97% on a held-out set (117k words) of MSA data, but performs poorly on ECA due to a high OOV rate (43%). By contrast, the OOV rate for taggers trained on ECA data is 9.5%. The minimally-supervised tagger (System IV) achieves a baseline accuracy of 62.76%. In the following sections, we present several methods to improve this system, in order to approximate as closely as possible the performance of the supervised systems. [1]

System	Total	AW	UW	OOV
I	39.84	55.98	21.05	19.21
II	92.53	98.64	99.09	32.20
III	84.88	90.17	99.11	32.64
IV	62.76	67.07	20.74	21.84

Table 5: Tagging accuracy (%) for the 4 baseline systems. AW = analyzable words, UW unanalyzable words, OOV = out-of-vocabulary words.

4 System Improvements

4.1 Adding Affix Features

The low accuracy of unanalyzable and OOV words may significantly impact downstream applications.

[1] The accuracy of a naive tagger which labels all words with the most likely tag (NN) achieves an accuracy of 20%. A tagger which labels words with the most likely tag amongst its possible tags achieves an accuracy of 52%.

One common way to improve accuracy is to add word features. In particular, we are interested in features that can be derived automatically from the script form, such as affixes. Affix features are added in a Naive Bayes fashion to the basic HMM model defined in Eq.1. In addition to the lexical model $p(w_i|t_i)$ we now have prefix and suffix models $p(a_i|t_i)$ and $p(b_i|t_i)$, where a and b are the prefix and suffix variables, respectively. The affixes used are: هم-, ها-, كم-, له-, لك-, لي-, ني-, ة-, ال-, فال-, وال-, بال-, لل-. Affixes are derived for each word by simple substring matching. More elaborate techniques are not used due to the philosophy of staying within a minimally supervised approach that does not require dialect-specific knowledge.

4.2 Constraining the Lexicon

The quality of the lexicon has a major impact on unsupervised HMM training. Banko et. al. (2004) demonstrated that tagging accuracy improves when the number of possible tags per word in a "noisy lexicon" can be restricted based on corpus frequency. In the current setup, words that are not analyzable by the MSA stemmer are initally assigned all possible tags, with the exception of obvious restricted tags like the begin and end-of-sentence tags, NOTAG, etc. Our goal is to constrain the set of tags for these unanalyzable words. To this end, we cluster both analyzable and unanalyzable words, and reduce the set of possible tags for unanalyzable words based on its cluster membership. Several different clustering algorithms could in principle be used; here we utilize Brown's clustering algorithm (Brown and others, 1992), which iteratively merges word clusters with high mutual information based on distributional criteria. The tagger lexicon is then derived as follows:

1. Generate K clusters of words from data.
2. For each cluster C, calculate $P(t|C) = \sum_{w \in A,C} P(t|w)P(w|C)$ where t and w are the word and tag, and A is the set of analyzable words.
3. The cluster's tagset is determined by choosing all tags t' with $P(t'|C)$ above a certain threshold γ.
4. All unanalyzable words within this cluster are given these possible tags.

The number of clusters K and the threshold γ are variables that affect the final tagset for unanalyzable words. Using $K = 200$ and $\gamma = 0.05$ for instance, the number of tags per unanalyzable word reduces to an average of four and ranges from one to eight tags. There is a tradeoff regarding the degree of tagset reduction: choosing fewer tags results in less confusability but may also involve the removal of the correct tag from a word's lexicon entry. We did not optimize for K and γ since an annotated development set for calculating accuracy is not available in a minimally supervised approach in practice. Nevertheless, we have observed that tagset reduction generally leads to improvements compared to the baseline system with an unconstrained lexicon.

The improvements gained from adding affix features to System IV and constraining the lexicon are shown in Table 6. We notice that adding affix features yields improvements in OOV accuracy. The relationship between the constrained lexicon and unanalyzable word accuracy is less straighforward. In this case, the degradation of unanalyzable word accuracy is due to the fact that the constrained lexicon over-restricts the tagsets of some frequent unanalyzable words. However, the constrained lexicon generally improves the overall accuracy and is thus a viable technique. Finally, the combination of affix features and constrained lexicon results in a tagger with 69.83% accuracy, which is a 7% absolute improvement over System IV.

System	Total	AW	UW	OOV
System IV	62.76	67.07	20.74	21.84
+affixes	67.48	71.30	22.82	**29.82**
+constrained lex	66.25	70.29	21.28	26.32
+both	**69.83**	**74.10**	**24.65**	27.68

Table 6: Improvements in tagging accuracy from adding affix features and constraining lexicon.

5 Cross-Dialectal Data Sharing

Next we examine whether unannotated corpora in other dialects (LCA) can be used to further improve the ECA tagger. The problem of data sparseness for Arabic dialects would be less severe if we were able to exploit the commonalities between similar dialects. In natural language processing, Kim & Khu-

danpur (2004) have explored techniques for using parallel Chinese/English corpora for language modeling. Parallel corpora have also been used to infer morphological analyzers, POS taggers, and noun phrase bracketers by projections via word alignments (Yarowsky et al., 2001). In (Hana et al., 2004), Czech data is used to develop a morphological analyzer for Russian.

In contrast to these works, we do not require parallel/comparable corpora or a bilingual dictionary, which may be difficult to obtain. Our goal is to develop general algorithms for utilizing the commonalities across dialects for developing a tool for a specific dialect. Although dialects can differ very strongly, they are similar in that they exhibit morphological simplifications and a different word order compared to MSA (e.g. SVO rather than VSO order), and close dialects share some vocabulary.

Each of the tagger components (i.e. contextual model $p(t_i|h_i)$, lexical model $p(w_i|t_i)$, and affix model $p(a_i|t_i)p(b_i|t_i)$) can be shared during training. In the following, we present two approaches for sharing data between dialects, each derived from following different assumptions about the underlying data generation process.

5.1 Contextual Model Interpolation

Contextual model interpolation is a widely-used data-sharing technique which assumes that models trained on data from different sources can be "mixed" in order to provide the most appropriate probability distribution for the target data. In our case, we have LCA as an out-of-domain data source, and ECA as the in-domain data source, with the former being about 4 times larger than the latter. If properly combined, the larger amount of out-of-domain data might improve the robustness of the in-domain tagger. We therefore use a linear interpolation of in-domain and out-of-domain contextual models. The joint probability $p(w_{0:M}, t_{0:M})$ becomes:

$$\prod_{i=0}^{M} p_E(w_i|t_i)(\lambda p_E(t_i|h_i) + (1-\lambda)p_L(t_i|h_i)) \quad (2)$$

Here λ defines the interpolation weights for the ECA contextual model $p_E(t_i|h_i)$ and the LCA contextual model $p_L(t_i|h_i)$. $p_E(w_n|t_n)$ is the ECA lexical model. The interpolation weight λ is estimated by maximizing the likelihood of a held-out data set given the combined model. As an extension, we allow the interpolation weights to be a function of the current tag: $\lambda(t_i)$, since class-dependent interpolation has shown improvements over basic interpolation in applications such as language modeling (Bulyko et al., 2003).

5.2 Joint Training of Contextual Model

As an alternative to model interpolation, we consider training a single model *jointly* from the two different data sets. The underlying assumption of this technique is that tag sequences in LCA and ECA are generated by the *same* process, whereas the observations (the words) are generated from the tag by two different processes in the two different dialects. The HMM model for joint training is expressed as:

$$\prod_{i=0}^{M}(\alpha_i p_E(w_i|t_i) + (1-\alpha_i)p_L(w_i|t_i))p_{E+L}(t_i|h_i)$$
$$\text{where} \quad \alpha_i = \begin{cases} 1 \text{ if word } w_i \text{ is ECA} \\ 0 \text{ otherwise} \end{cases} \quad (3)$$

A single conditional probability table is used for the contextual model, whereas the lexical model switches between two different parameter tables, one for LCA observations and another for ECA observations. During training, the contextual model is trained jointly from both ECA and LCA data; however, the data is divided into ECA and LCA portions when updating the lexical models. Both the contextual and lexical models are trained within the same training pass. A graphical model representation of this system is shown in Figure 1. This model can be implemented using the functionality of switching parents (Bilmes, 2000) provided by GMTK.

During decoding, the tagger can in principle switch its lexical model to ECA or LCA, depending on the input; this system thus is essentially a multi-dialect tagger. In the experiments reported below, however, we exclusively test on ECA, and the LCA lexical model is not used. Due to the larger amount of data available for contextual model, joint training can be expected to improve the performance on the target dialect. The affix models can be trained jointly in a similar fashion.

5.3 Data sharing results

The results for data sharing are shown in Table 7. The systems Interpolate-λ and Interpolate-$\lambda(t_i)$ are taggers built by interpolation and class-dependent interpolation, respectively. For joint training, we present results for two systems: JointTrain(1:4) is trained on the existing collection ECA and LCA data, which has a 1:4 ratio in terms of corpus size; JointTrain(2:1) weights the ECA data twice, in order to bias the training process more towards ECA's distribution. We also provide results for two more taggers: the first (CombineData) is trained "naively" on the pooled data from both ECA and LCA, without any weighting, interpolation, or changes to the probabilistic model. The second (CombineLex) uses a contextual model trained on ECA and a lexical model estimated from both ECA and LCA data. The latter was trained in order to assess the potential for improvement due to the reduction in OOV rate on the dev set when adding the LCA data (cf. Table 4). All the above systems utilize the constrained lexicon, as it consistently gives improvements.

Table 7 shows that, as expected, the brute-force combination of training data is not helpful and degrades performance. CombineLex results in higher accuracy but does not outperform models in Table 6. The same is true of the taggers using model interpolation. The best performance is obtained by the system using the joint contextual model with separate lexical models, with 2:1 weighting of ECA vs. LCA data. Finally, we added word affix information to the best shared-data system, which resulted in an accuracy of 70.88%. In contrast, adding affix to CombineData achieves 61.78%, suggesting that improvements in JointTrain comes from the joint training technique rather than simple addition of new data. This result is directly comparable to the best system in Section 4 (last row of Table 6)[2].

The analysis of tagging errors revealed that the most frequent confusions are between VBD/NNS, VBP/VBD, and JJ/NN. Commonly mistagged words include cases like يعني ("means"-3rd.sg), which is labeled as a particle in the reference but is most often tagged as a verb, which is also a reasonable tag.

[2]We also experimented with joint training of ECA+MSA. This gave good OOV accuracy, but overall it did not improve over the best system in Section 4. Also, note that all accuracies are calculated by ignoring the scoring of ambiguous words, which have several possible tags as the correct reference. If we score the ambiguous words as correct when the hypothesized tag is within this set, the accuracy of ECA+LCA+affi x JointTrain rises to 77.18%, which is an optimistic upper-bound on the total accuracy.

System	Total	AW	UW	OOV
CombineData	60.79	64.21	20.27	26.10
CombineLex	65.13	69.47	18.81	22.34
Interpolate-λ	62.82	67.42	16.98	17.44
Interpolate-$\lambda(t_i)$	63.49	67.96	17.19	19.33
JointTrain(1:4)	62.53	66.18	27.78	26.52
JointTrain(2:1)	**66.95**	**71.02**	**31.72**	**26.81**
JointTrain(2:1)+affix				
w/ ECA+LCA	**70.88**	**75.20**	28.17	**34.06**
w/ ECA+MSA	67.85	71.50	17.76	31.76

Table 7: Tagging accuracy for various data sharing methods.

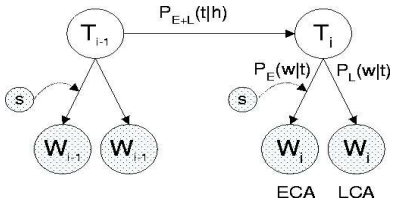

Figure 1: Graphical Model of Joint Training: switching between different lexical models while sharing the underlying contextual model. The variable s represents the α term in Eq. 3 and chooses the lexical model depending on the origin of the word.

6 Discussion and Future Work

Table 8 highlights the performance of the various taggers on the ECA *evaluation set*. The accuracy of the unsupervised HMM tagger (System IV) improves from 58.47% to 66.61% via the affix features and constrained lexicon, and to a 68.48% by including joint training. These improvements are statistical significant at the 0.005 level according to a difference-of-proportions test.

Several of the methods proposed here deserve further work: first, additional ways of constraining the lexicon can be explored, which may include imposing probability distributions on the possible tags for unanalyzable words. Other clustering algorithms (e.g. root-based clustering of Arabic (De Roeck and

Al-Fares, 2000)), may be used instead of, or in addition to, distribution-based clustering.

Cross-dialectal data sharing for tagging also deserves more research. For instance, the performance of the contextual model interpolation might be increased if one trains interpolation weights dependent on the classes based on previous two tags. Joint training of contextual model and data sharing for lexical models can be combined; other dialectal data may also be added into the same joint training framework. It would also be useful to extend these methods to create a more fine-grained part-of-speech tagger with case, person, number, etc. information. Stems, POS, and fine-grained POS can be combined into a factorial hidden Markov model, so that relationships between the stems and POS as well as data sharing between dialects can be simultaneously exploited to build a better system.

In conclusion, we have presented the first steps towards developing a dialectal Arabic tagger with minimal supervision. We have shown that adding affix features and constraining the lexicon for unanalyzable words are simple resource-light methods to improve tagging accuracy. We also explore the possibility of improving an ECA tagger using LCA data and present two data sharing methods. The combination of these techniques yield a 10% improvement over the baseline.

System	Total	AW	UW	OOV
System IV	58.47	64.71	22.34	17.50
+affix+lexicon	66.61	72.87	20.17	**25.49**
Interpolate II	60.07	66.56	20.55	17.61
JointTr.+affix	**68.48**	**76.20**	**48.44**	17.76
CombineLex	61.35	68.12	16.02	16.87

Table 8: Tagging accuracy on ECA evaluation set

Acknowledgements
This material is based on work funded by the NSF and the CIA under NSF Grant No. IIS-0326276. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of these agencies.

References

M. Banko and R. Moore. 2004. Part-of-speech tagging in context. In *Proc. of COLING 2004*.

A. Bies. 2003. Guide to collapsing Arabic tagset. http://www.ircs.upenn.edu/arabic/Jan03release/arabic-POStags-collapse-to-PennPOStags.txt.

J. Bilmes and G. Zweig. 2002. The Graphical Models Toolkit: an open-source software system for speech and time series processing. In *Proc. of ICASSP*.

J. Bilmes. 2000. Dynamic Bayesian multi-networks. In *The 16th Conf. on Uncertainty in Artificial Intelligence*.

T. Brants. 2000. TnT - a statistical part-of-speech tagger. In *Proc. of 6th Applied Natural Language Processing Conf.*

E. Brill. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. In *The 3rd Workshop on Very Large Corpora*.

P.F. Brown et al. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

I. Bulyko, M. Ostendorf, and A. Stolcke. 2003. Getting more mileage from web text for conversational speech language modeling using class-dependent mixtures. In *Proc. of HLT*.

D. Cutting et al. 1992. A practical part-of-speech tagger. In *Proc. 3rd Conf. on Applied Natural Language Processing*.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. 1996. MBT: A memory-based part of speech tagger-generator. In *4th Workshop on Very Large Corpora*.

A. De Roeck and W. Al-Fares. 2000. A morphologically sensitive clustering algorithm for identifying arabic roots. In *Proceedings of the 38th Annual Meeting of the ACL*.

M. Diab, K. Hacioglu, and D. Jurafsky. 2004. Automatic tagging of Arabic text: from raw text to base phrase chunks. In *Proceedings of HLT/NAACL*.

J. Hana, A. Feldman, and C. Brew. 2004. A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In *Proc. of EMNLP 2004*, July.

P. Heeman. 1998. POS vs. classes in language modeling. In *Proc. of 6th Workshop on Very Large Corpora*.

S. Khoja. 2001. APT: Arabic part-of-speech tagger. In *Proc. of the NAACL Student Workshop*.

W. Kim and S. Khudanpur. 2004. Lexical triggers and latent semantic analysis for cross-lingual language model adaptation. *ACM Trans. on Asian Language Info. Processing*, 3:94–112.

R. Koeling. 2000. Chunking with maximum entropy models. In *Proceedings of CoNLL*.

M. Osborne. 2000. Shallow parsing as part-of-speech tagging. In *Proc. of CoNLL*.

A. Rathnaparki. 1996. A maximum-entropy part-of-speech tagger. In *Proc. of EMNLP*.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*.

D. Vergyri, K. Kirchhoff, K. Duh, and A. Stolcke. 2004. Morphology-based language modeling for Arabic speech recognition. In *Proc. of ICSLP*.

D. Yarowsky, G. Ngai, and R. Wicentowski. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proc. of HLT*.

The Impact of Morphological Stemming on Arabic Mention Detection and Coreference Resolution

Imed Zitouni, Jeff Sorensen, Xiaoqiang Luo, Radu Florian
{izitouni, sorenj, xiaoluo, raduf}@watson.ibm.com
IBM T.J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA

Abstract

Arabic presents an interesting challenge to natural language processing, being a highly inflected and agglutinative language. In particular, this paper presents an in-depth investigation of the entity detection and recognition (EDR) task for Arabic. We start by highlighting why segmentation is a necessary prerequisite for EDR, continue by presenting a finite-state statistical segmenter, and then examine how the resulting segments can be better included into a mention detection system and an entity recognition system; both systems are statistical, build around the maximum entropy principle. Experiments on a clearly stated partition of the ACE 2004 data show that stem-based features can significantly improve the performance of the EDT system by 2 absolute F-measure points. The system presented here had a competitive performance in the ACE 2004 evaluation.

1 Introduction

Information extraction is a crucial step toward understanding and processing language. One goal of information extraction tasks is to identify important conceptual information in a discourse. These tasks have applications in summarization, information retrieval (one can get all hits for Washington/person and not the ones for Washington/state or Washington/city), data mining, question answering, language understanding, etc.

In this paper we focus on the Entity Detection and Recognition task (EDR) for Arabic as described in ACE 2004 framework (ACE, 2004). The EDR has close ties to the named entity recognition (NER) and coreference resolution tasks, which have been the focus of several recent investigations (Bikel et al., 1997; Miller et al., 1998; Borthwick, 1999; Mikheev et al., 1999; Soon et al., 2001; Ng and Cardie, 2002; Florian et al., 2004), and have been at the center of evaluations such as: MUC-6, MUC-7, and the CoNLL'02 and CoNLL'03 shared tasks. Usually, in computational linguistics literature, a *named entity* is an instance of a location, a person, or an organization, and the NER task consists of identifying each of these occurrences. Instead, we will adopt the nomenclature of the Automatic Content Extraction program (NIST, 2004): we will call the instances of textual references to objects/abstractions *mentions*, which can be either named (e.g. John Mayor), nominal (the president) or pronominal (she, it). An entity is the aggregate of all the mentions (of any level) which refer to one conceptual entity. For instance, in the sentence

> President John Smith said he has no comments

there are two mentions (named and pronomial) but only one entity, formed by the set {John Smith, he}.

We separate the EDR task into two parts: a mention detection step, which identifies and classifies all the mentions in a text – and a coreference resolution step, which combinines the detected mentions into groups that refer to the same object. In its entirety, the EDR task is arguably harder than traditional named entity recognition, because of the additional complexity involved in extracting non-named mentions (nominal and pronominal) and the requirement of grouping mentions into entities. This is particularly true for Arabic where nominals and pronouns are also attached to the word they modify. In fact, most Arabic words are morphologically derived from a list of base forms or *stems*, to which prefixes and suffixes can be attached to form Arabic surface forms (blank-delimited words). In addition to the different forms of the Arabic word that result from the

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 63–70,
Ann Arbor, June 2005.

derivational and inflectional process, most prepositions, conjunctions, pronouns, and possessive forms are attached to the Arabic surface word. It is these orthographic variations and complex morphological structure that make Arabic language processing challenging (Xu et al., 2001; Xu et al., 2002).

Both tasks are performed with a statistical framework: the mention detection system is similar to the one presented in (Florian et al., 2004) and the coreference resolution system is similar to the one described in (Luo et al., 2004). Both systems are built around from the maximum-entropy technique (Berger et al., 1996). We formulate the mention detection task as a sequence classification problem. While this approach is language independent, it must be modified to accomodate the particulars of the Arabic language. The Arabic words may be composed of zero or more prefixes, followed by a stem and zero or more suffixes. We begin with a segmentation of the written text before starting the classification. This segmentation process consists of separating the normal whitespace delimited words into (hypothesized) prefixes, stems, and suffixes, which become the subject of analysis (tokens). The resulting granularity of breaking words into prefixes and suffixes allows different mention type labels beyond the stem label (for instance, in the case of nominal and pronominal mentions). Additionally, because the prefixes and suffixes are quite frequent, directly processing unsegmented words results in significant data sparseness.

We present in Section 2 the relevant particularities of the Arabic language for natural language processing, especially for the EDR task. We then describe the segmentation system we employed for this task in Section 3. Section 4 briefly describes our mention detection system, explaining the different feature types we use. We focus in particular on the stem n-gram, prefix n-gram, and suffix n-gram features that are specific to a morphologically rich language such as Arabic. We describe in Section 5 our coreference resolution system where we also describe the advantage of using stem based features. Section 6 shows and discusses the different experimental results and Section 7 concludes the paper.

2 Why is Arabic Information Extraction difficult?

The Arabic language, which is the mother tongue of more than 300 million people (Center, 2000), present significant challenges to many natural language processing applications. Arabic is a highly inflected and derived language. In Arabic morphology, most morphemes are comprised of a basic word form (the root or stem), to which many affixes can be attached to form Arabic words. The Arabic alphabet consists of 28 letters that can be extended to ninety by additional shapes, marks, and vowels (Tayli and Al-Salamah, 1990). Unlike Latin-based alphabets, the orientation of writing in Arabic is from right to left. In written Arabic, short vowels are often omitted. Also, because variety in expression is appreciated as part of a good writing style, the synonyms are widespread. Arabic nouns encode information about gender, number, and grammatical cases. There are two genders (masculine and feminine), three numbers (singular, dual, and plural), and three grammatical cases (nominative, genitive, and accusative).

A noun has a nominative case when it is a subject, accusative case when it is the object of a verb, and genitive case when it is the object of a preposition. The form of an Arabic noun is consequently determined by its gender, number, and grammatical case. The definitive nouns are formed by attaching the Arabic article ال to the immediate front of the nouns, such as in the word الشركة (the company). Also, prepositions such as ب (by), and ل (to) can be attached as a prefix as in للشركة (to the company). A noun may carry a possessive pronoun as a suffix, such as in شركتهم (their company). For the EDR task, in this previous example, the Arabic blank-delimited word شركتهم should be split into two tokens: شركة and هم. The first token شركة is a mention that refers to an organization, whereas the second token هم is also a mention, but one that may refer to a person. Also, the prepositions (i.e., ب and ل) not be considered a part of the mention.

Arabic has two kinds of plurals: broken plurals and sound plurals (Wightwick and Gaafar, 1998; Chen and Gey, 2002). The formation of broken plurals is common, more complex and often irregular. As an example, the plural form of the noun رجل (man) is رِجَال (men), which is formed by inserting the infix ا. The plural form of the noun كتَاب (book) is كتب (books), which is formed by deleting the infix ا. The plural form and the singular form may also be completely different (e.g. إمرأة for woman, but نسَاء for women). The sound plurals are formed by adding plural suffixes to singular nouns (e.g., بَاحث meaning researcher): the plural suffix is ات for feminine nouns in grammatical cases (e.g., بَاحثَات), ون for masculine nouns in the nominative case (e.g., بَاحثون), and ين for masculine nouns in the genitive and accusative cases (e.g., بَاحثين). The dual suffix is ان for the nominative case (e.g., بَاحثَان), and ين for the genitive or accusative (e.g., بَاحثين).

Because we consider pronouns and nominals as mentions, it is essential to segment Arabic words into these subword tokens. We also believe that the in-

formation denoted by these affixes can help with the coreference resolution task[1].

Arabic verbs have perfect and imperfect tenses (Abbou and McCarus, 1983). Perfect tense denotes completed actions, while imperfect denotes ongoing actions. Arabic verbs in the perfect tense consist of a stem followed by a subject marker, denoted as a suffix. The subject marker indicates the person, gender, and number of the subject. As an example, the verb قَابل (to meet) has a perfect tense قَابَلت for the third person feminine singular, and قَابَلُوا for the third person masculine plural. We notice also that a verb with a subject marker and a pronoun suffix can be by itself a complete sentence, such us in the word قَابلتهم: it has a third-person feminine singular subject-marker ت (she) and a pronoun suffix هم (them). It is also a complete sentence meaning "she met them." The subject markers are often suffixes, but we may find a subject marker as a combination of a prefix and a suffix as in تقَابلهم (she meets them). In this example, the EDR system should be able to separate تقَابلهم, to create two mentions (ت and هم). Because the two mentions belong to different entities, the EDR system should not chain them together. An Arabic word can potentially have a large number of variants, and some of the variants can be quite complex. As an example, consider the word ولبَاحثيهَا (and to her researchers) which contains two prefixes and one suffix (و + ل + بَاحثي + هَا).

3 Arabic Segmentation

Lee et al. (2003) demonstrates a technique for segmenting Arabic text and uses it as a morphological processing step in machine translation. A trigram language model was used to score and select among hypothesized segmentations determined by a set of prefix and suffix expansion rules.

In our latest implementation of this algorithm, we have recast this segmentation strategy as the composition of three distinct finite state machines. The first machine, illustrated in Figure 1 encodes the prefix and suffix expansion rules, producing a lattice of possible segmentations. The second machine is a dictionary that accepts characters and produces identifiers corresponding to dictionary entries. The final machine is a trigram language model, specifically a Kneser-Ney (Chen and Goodman, 1998) based backoff language model. Differing from (Lee et al., 2003), we have also introduced an explicit model for unknown words based upon a character unigram model, although this model is dominated by an empirically chosen unknown word penalty. Using 0.5M words from the combined Arabic Treebanks 1V2, 2V2 and 3V1, the dictionary based segmenter achieves a exact word match 97.8% correct segmentation.

[1]As an example, we do not chain mentions with different gender, number, etc.

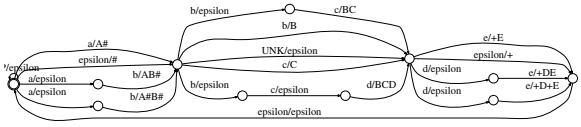

Figure 1: Illustration of dictionary based segmentation finite state transducer

3.1 Bootstrapping

In addition to the model based upon a dictionary of stems and words, we also experimented with models based upon character n-grams, similar to those used for Chinese segmentation (Sproat et al., 1996). For these models, both arabic characters and spaces, and the inserted prefix and suffix markers appear on the arcs of the finite state machine. Here, the language model is conditioned to insert prefix and suffix markers based upon the frequency of their appearance in n-gram character contexts that appear in the training data. The character based model alone achieves a 94.5% exact match segmentation accuracy, considerably less accurate then the dictionary based model.

However, an analysis of the errors indicated that the character based model is more effective at segmenting words that do not appear in the training data. We seeked to exploit this ability to generalize to improve the dictionary based model. As in (Lee et al., 2003), we used unsupervised training data which is automatically segmented to discover previously unseen stems. In our case, the character n-gram model is used to segment a portion of the Arabic Gigaword corpus. From this, we create a vocabulary of stems and affixes by requiring that tokens appear more than twice in the supervised training data or more than ten times in the unsupervised, segmented corpus.

The resulting vocabulary, predominately of word stems, is 53K words, or about six times the vocabulary observed in the supervised training data. This represents about only 18% of the total number of unique tokens observed in the aggregate training data. With the addition of the automatically acquired vocabulary, the segmentation accuracy achieves 98.1% exact match.

3.2 Preprocessing of Arabic Treebank Data

Because the Arabic treebank and the gigaword corpora are based upon news data, we apply some small amount of regular expression based preprocessing. Arabic specific processing include removal of the characters *tatweel* (ـ), and vowels. Also, the following characters are treated as an equivalence class during all lookups and processing: (1) ى ,ي, and (2) آ، أ، إ، ا. We define a token and introduce whitespace boundaries between every span of one or more alphabetic or numeric characters. Each punctuation symbol is considered a separate token. Character classes, such as punctuation, are defined according to the Unicode Standard (Aliprand et al., 2004).

4 Mention Detection

The mention detection task we investigate identifies, for each mention, four pieces of information:

1. the mention type: person (PER), organization (ORG), location (LOC), geopolitical entity (GPE), facility (FAC), vehicle (VEH), and weapon (WEA)
2. the mention level (named, nominal, pronominal, or premodifier)
3. the mention class (generic, specific, negatively quantified, etc.)
4. the mention sub-type, which is a sub-category of the mention type (ACE, 2004) (e.g. OrgGovernmental, FacilityPath, etc.).

4.1 System Description

We formulate the mention detection problem as a classification problem, which takes as input segmented Arabic text. We assign to each token in the text a label indicating whether it starts a specific mention, is inside a specific mention, or is outside any mentions. We use a maximum entropy Markov model (MEMM) classifier. The principle of maximum entropy states that when one searches among probability distributions that model the observed data (evidence), the preferred one is the one that maximizes the entropy (a measure of the uncertainty of the model) (Berger et al., 1996). One big advantage of this approach is that it can combine arbitrary and diverse types of information in making a classification decision.

Our mention detection system predicts the four labels types associated with a mention through a cascade approach. It first predicts the boundary and the main entity type for each mention. Then, it uses the information regarding the type and boundary in different second-stage classifiers to predict the subtype, the mention level, and the mention class. After the first stage, when the boundary (starting, inside, or outside a mention) has been determined, the other classifiers can use this information to analyze a larger context, capturing the patterns around the entire mentions, rather than words. As an example, the token sequence that refers to a mention will become a single recognized unit and, consequently, lexical and syntactic features occuring inside or outside of the entire mention span can be used in prediction.

In the first stage (entity type detection and classification), Arabic blank-delimited words, after segmenting, become a series of tokens representing prefixes, stems, and suffixes (cf. section 2). We allow any contiguous sequence of tokens can represent a mention. Thus, prefixes and suffixes can be, and often are, labeled with a different mention type than the stem of the word that contains them as constituents.

4.2 Stem n-gram Features

We use a large set of features to improve the prediction of mentions. This set can be partitioned into 4 categories: lexical, syntactic, gazetteer-based, and those obtained by running other named-entity classifiers (with different tag sets). We use features such as the shallow parsing information associated with the tokens in a window of 3 tokens, POS, etc.

The context of a current token t_i is clearly one of the most important features in predicting whether t_i is a mention or not (Florian et al., 2004). We denote these features as backward token tri-grams and forward token tri-grams for the previous and next context of t_i respectively. For a token t_i, the backward token n-gram feature will contains the previous $n-1$ tokens in the history $(t_{i-n+1}, \ldots t_{i-1})$ and the forward token n-gram feature will contains the next $n-1$ tokens $(t_{i+1}, \ldots t_{i+n-1})$.

Because we are segmenting arabic words into multiple tokens, there is some concern that trigram contexts will no longer convey as much contextual information. Consider the following sentence extracted from the development set: هذَا يمثل المَقر لِلمَكتب السيَاسي لِلحزب (translation "This represents the location for Political Party Office"). The "Political Party Office" is tagged as an organization and, as a word-for-word translation, is expressed as "to the Office of the political to the party". It is clear in this example that the word مَقر (location for) contains crucial information in distinguishing between a location and an organization when tagging the token مَكتب

(office). After segmentation, the sentence becomes: هذَا + ي + مثل + اَل + مَقر + ل + اَل + مَكتب + اَل + سيَاسي + ل + اَل + حزب.

When predicting if the token مَكتب (office) is the beginning of an organization or not, backward and forward token n-gram features contain only ل + اَل (for the) and سيَاسي + اَل (the political). This is most likely not enough context, and addressing the problem by increasing the size of the n-gram context quickly leads to a data sparseness problem.

We propose in this paper the *stem n-gram* features as additional features to the lexical set. If the current token t_i is a stem, the backward stem n-gram feature contains the previous $n-1$ stems and the forward stem n-gram feature will contain the following $n-1$ stems. We proceed similarly for prefixes and suffixes: if t_i is a prefix (or suffix, respectively) we take the previous and following prefixes (or suffixes)[2]. In the sentence shown above, when the system is predicting if the token مَكتب (office) is the beginning of an organization or not, the backward and forward stem n-gram features contain مثل مَقر (represent location of) and سيَاسي حزب (political office). The stem features contain enough information in this example to make a decision that مَكتب (office) is the beginning of an organization. In our experiments, n is 3, therefore we use stem trigram features.

5 Coreference Resolution

Coreference resolution (or entity recognition) is defined as grouping together mentions referring to the same object or *entity*. For example, in the following text,

(I) "John believes Mary to be the best student"

three mentions "John", "Mary", "student" are underlined. "Mary" and "student" are in the same entity since both refer to the same person.

The coreference system system is similar to the Bell tree algorithm as described by (Luo et al., 2004).

In our implementation, the link model between a candidate entity e and the current mention m is computed as

$$P_L(L=1|e,m) \approx \max_{m_k \in e} \hat{P}_L(L=1|e,m_k,m), \quad (1)$$

where m_k is one mention in entity e, and the basic model building block $\hat{P}_L(L=1|e,m_k,m)$ is an exponential or maximum entropy model (Berger et al., 1996).

For the start model, we use the following approximation:

$$P_S(S=1|e_1,e_2,\cdots,e_t,m) \approx 1 - \max_{1\le i\le t} P_L(L=1|e_i,m) \quad (2)$$

The start model (cf. equation 2) says that the probability of starting a new entity, given the current mention m and the previous entities $e_1,e_2,\cdots,e_t$, is simply 1 minus the maximum link probability between the current mention and one of the previous entities.

The maximum-entropy model provides us with a flexible framework to encode features into the the system. Our Arabic entity recognition system uses many language-indepedent features such as strict and partial string match, and distance features (Luo et al., 2004). In this paper, however, we focus on the addition of Arabic stem-based features.

5.1 Arabic Stem Match Feature

Features using the word context (left and right tokens) have been shown to be very helpful in coreference resolution (Luo et al., 2004). For Arabic, since words are morphologically derived from a list of roots (stems), we expected that a feature based on the right and left stems would lead to improvement in system accuracy.

Let m_1 and m_2 be two candidate mentions where a mention is a string of tokens (prefixes, stems, and suffixes) extracted from the segmented text. In order to make a decision in either linking the two mentions or not we use additional features such as: do the stems in m_1 and m_2 match, do stems in m_1 match *all* stems in m_2, do stems in m_1 *partially* match stems in m_2. We proceed similarly for prefixes and suffixes. Since prefixes and suffixes can belong to different mention types, we build a parse tree on the segmented text and we can explore features dealing with the gender and number of the token. In the following example, between parentheses we make a word-for-word translations in order to better explain our stemming feature. Let us take the two mentions لِلمَكتب السيَاسي لِلحزب (to-the-office the-politic to-the-party) and مَكتب الحزبي (office the-party's) segmented as ل + اَل + مَكتب + اَل + سيَاسي + ل + اَل + حزب and مَكتب + اَل + حزب + ي respectively. In our

[2]Thus, the difference to token n-grams is that the tokens of different type are removed from the streams, before the features are created.

development corpus, these two mentions are chained to the same entity. The stemming match feature in this case will contain information such us *all stems* of m_2 match, which is a strong indicator that these mentions should be chained together. Features based on the words alone would not help this specific example, because the two strings m_1 and m_2 do not match.

6 Experiments

6.1 Data

The system is trained on the Arabic ACE 2003 and part of the 2004 data. We introduce here a clearly defined and replicable split of the ACE 2004 data, so that future investigations can accurately and correctly compare against the results presented here.

There are 689 Arabic documents in LDC's 2004 release (version 1.4) of ACE data from three sources: the Arabic Treebank, a subset of the broadcast (bnews) and newswire (nwire) TDT-4 documents. The 178-document devtest is created by taking the last (in chronological order) 25% of documents in each of three sources: 38 Arabic treebank documents dating from "20000715" (i.e., July 15, 2000) to "20000815," 76 bnews documents from "20001205.1100.0489" (i.e., Dec. 05 of 2000 from 11:00pm to 04:89am) to "20001230.1100.1216," and 64 nwire documents from "20001206.1000.0050" to "20001230.0700.0061." The time span of the test set is intentionally non-overlapping with that of the training set within each data source, as this models how the system will perform in the real world.

6.2 Mention Detection

We want to investigate the usefulness of stem n-gram features in the mention detection system. As stated before, the experiments are run in the ACE'04 framework (NIST, 2004) where the system will identify mentions and will label them (cf. Section 4) with a type (person, organization, etc), a sub-type (OrgCommercial, OrgGovernmental, etc), a mention level (named, nominal, etc), and a class (specific, generic, etc). Detecting the mention boundaries (set of consecutive tokens) and their main type is one of the important steps of our mention detection system. The score that the ACE community uses (ACE value) attributes a higher importance (outlined by its weight) to the main type compared to other subtasks, such as the mention level and the class. Hence, to build our mention detection system we spent a lot of effort in improving the first step: detecting the mention boundary and their main type. In this paper, we report the results in terms of precision, recall, and F-measure[3].

Lexical features			
	Precision (%)	Recall (%)	F-measure (%)
Total	73.3	58.0	**64.7**
FAC	76.0	24.0	36.5
GPE	79.4	65.6	71.8
LOC	57.7	29.9	39.4
ORG	63.1	46.6	53.6
PER	73.2	63.5	68.0
VEH	83.5	29.7	43.8
WEA	77.3	25.4	38.2
Lexical features + Stem			
	Precision (%)	Recall (%)	F-measure (%)
Total	73.6	59.4	**65.8**
FAC	72.7	29.0	41.4
GPE	79.9	67.2	73.0
LOC	58.6	31.9	41.4
ORG	62.6	47.2	53.8
PER	73.8	64.6	68.9
VEH	81.7	35.9	49.9
WEA	78.4	29.9	43.2

Table 1: Performance of the mention detection system using lexical features only.

To assess the impact of stemming n-gram features on the system under different conditions, we consider two cases: one where the system only has access to lexical features (the tokens and direct derivatives including standard n-gram features), and one where the system has access to a richer set of information, including lexical features, POS tags, text chunks, parse tree, and gazetteer information. The former framework has the advantage of being fast (making it more appropriate for deployment in commercial systems). The number of parameters to optimize in the MaxEnt framework we use when only lexical features are explored is around 280K parameters. This number increases to 443K approximately when all information is used except the stemming feature. The number of parameters introduced by the use of stemming is around 130K parameters. Table 1 reports experimental results using lexical features only; we observe that the stemming n-gram features boost the performance by one point (64.7 vs. 65.8). It is important to notice the stemming n-gram features improved the performance of each category of the main type.

In the second case, the systems have access to a large amount of feature types, including lexical, syntactic, gazetteer, and those obtained by running other

[3]The ACE value is an important factor for us, but its relative complexity, due to different weights associated with the subparts, makes for a hard comparison, while the F-measure is relatively easy to interpret.

AllFeatures			
	Precision (%)	Recall (%)	F-measure (%)
Total	74.3	64.0	**68.8**
FAC	72.3	36.8	48.8
GPE	80.5	70.8	75.4
LOC	61.1	35.4	44.8
ORG	61.4	50.3	55.3
PER	75.3	70.2	72.7
VEH	83.2	38.1	52.3
WEA	69.0	36.6	47.8
All-Features + Stem			
	Precision (%)	Recall (%)	F-measure (%)
Total	74.4	64.6	**69.2**
FAC	68.8	38.5	49.4
GPE	80.8	71.9	76.1
LOC	60.2	36.8	45.7
ORG	62.2	51.0	56.1
PER	75.3	70.2	72.7
VEH	81.4	41.8	55.2
WEA	70.3	38.8	50.0

Table 2: Performance of the mention detection system using lexical, syntactic, gazetteer features as well as features obtained by running other named-entity classifiers

named-entity classifiers (with different semantic tag sets). Features are also extracted from the shallow parsing information associated with the tokens in window of 3, POS, etc. The *All-features* system incorporates all the features except for the stem n-grams. Table 2 shows the experimental results with and without the stem n-grams features. Again, Table 2 shows that using stem n-grams features gave a small boost to the whole main-type classification system[4]. This is true for all types. It is interesting to note that the increase in performance in both cases (Tables 1 and 2) is obtained from increased recall, with little change in precision. When the prefix and suffix n-gram features are removed from the feature set, we notice in both cases (Tables 1 and 2) a insignificant decrease of the overall performance, which is expected: what should a feature of preceeding (or following) prepositions or finite articles captures?

As stated in Section 4.1, the mention detection system uses a cascade approach. However, we were curious to see if the gain we obtained at the first level was successfully transfered into the overall performance of the mention detection system. Table 3 presents the performance in terms of precision, recall, and F-measure of the whole system. Despite the fact that the improvement was small in terms of F-measure (59.4 vs. 59.7), the stemming n-gram features gave interesting improvement in terms of ACE value to the hole EDR system as showed in section 6.3.

[4]The difference in performance is not statistically significant

	Precision (%)	Recall (%)	F-measure (%)
All-Features	64.2	55.3	59.4
All-Features+Stem	64.4	55.7	59.7
Lexical	64.4	50.8	56.8
Lexical+Stem	64.6	52.0	57.6

Table 3: Performance of the mention detection system including all ACE'04 subtasks

6.3 Coreference Resolution

In this section, we present the coreference results on the devtest defined earlier. First, to see the effect of stem matching features, we compare two coreference systems: one with the stem features, the other without. We test the two systems on both "true" and system mentions of the devtest set. "True" mentions mean that input to the coreference system are mentions marked by human, while system mentions are output from the mention detection system. We report results with two metrics: ECM-F and ACE-Value. ECM-F is an entity-constrained mention F-measure (cf. (Luo et al., 2004) for how ECM-F is computed), and ACE-Value is the official ACE evaluation metric. The result is shown in Table 4: the baseline numbers without stem features are listed under "Base," and the results of the coreference system with stem features are listed under "Base+Stem."

On true mention, the stem matching features improve ECM-F from 77.7% to 80.0%, and ACE-value from 86.9% to 88.2%. The similar improvement is also observed on system mentions.The overall ECM-F improves from 62.3% to 64.2% and the ACE value improves from 61.9 to 63.1%. Note that the increase on the ACE value is smaller than ECM-F. This is because ACE-value is a weighted metric which emphasizes on NAME mentions and heavily discounts PRONOUN mentions. Overall the stem features give rise to consistent gain to the coreference system.

7 Conclusion

In this paper, we present a fully fledged Entity Detection and Tracking system for Arabic. At its base, the system fundamentally depends on a finite state segmenter and makes good use of the relationships that occur between word stems, by introducing features which take into account the type of each segment. In mention detection, the features are represented as stem n-grams, while in coreference resolution they are captured through stem-tailored match features.

	Base		Base+Stem	
	ECM-F	ACEVal	ECM-F	ACEVal
Truth	77.7	86.9	80.0	88.2
System	62.3	61.9	64.2	63.1

Table 4: Effect of Arabic stemming features on coreference resolution. The row marked with "Truth" represents the results with "true" mentions while the row marked with "System" represents that mentions are detected by the system. Numbers under "ECM-F" are Entity-Constrained-Mention F-measure and numbers under "ACE-Val" are ACE-values.

These types of features result in an improvement in both the mention detection and coreference resolution performance, as shown through experiments on the ACE 2004 Arabic data. The experiments are performed on a clearly specified partition of the data, so comparisons against the presented work can be correctly and accurately made in the future. In addition, we also report results on the official test data.

The presented system has obtained competitive results in the ACE 2004 evaluation, being ranked amongst the top competitors.

8 Acknowledgements

This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916. The views and findings contained in this material are those of the authors and do not necessarily reflect the position of policy of the U.S. government and no official endorsement should be inferred.

References

Peter F. Abbou and Ernest N. McCarus, editors. 1983. *Elementary modern standard Arabic.* Cambridge University Press.

ACE. 2004. Automatic content extraction. http://www.ldc.upenn.edu/Projects/ACE/.

Joan Aliprand, Julie Allen, Joe Becker, Mark Davis, Michael Everson, Asmus Freytag, John Jenkins, Mike Ksar, Rick McGowan, Eric Muller, Lisa Moore, Michel Suignard, and Ken Whistler. 2004. The unicode standard. http://www.unicode.org/.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, pages 194–201.

A. Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition.* Ph.D. thesis, New York University.

Egyptian Demographic Center. 2000. http://www.frcu.eun.eg/www/homepage/cdc/cdc.htm.

Aitao Chen and Fredic Gey. 2002. Building an arabic stemmer for information retrieval. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, National Institute of Standards and Technology, November.

S. F. Chen and J. Goodman. 1998. An empirical study of smoothing techinques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, Cambridge, Massachusettes, August.

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

Y.-S. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan. 2003. Language model based Arabic word segmentation. In *Proceedings of the ACL'03*, pages 399–406.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proc. of ACL'04*.

A. Mikheev, M. Moens, and C. Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of EACL'99*.

S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwarz, R. Stone, and R. Weischedel. 1998. Bbn: Description of the SIFT system as used for MUC-7. In *MUC-7*.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the ACL'02*, pages 104–111.

NIST. 2004. Proceedings of ace evaluation and pi meeting 2004 workshop. Alexandria, VA, September. NIST.

W. M. Soon, H. T. Ng, and C. Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

R. Sproat, C. Shih, W. Gale, and N. Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3).

M. Tayli and A. Al-Salamah. 1990. Building bilingual microcomputer systems. *Communications of the ACM*, 33(5):495–505.

J. Wightwick and M. Gaafar. 1998. *Arabic Verbs and Essentials of Grammar.* Passport Books.

J. Xu, A. Fraser, and R. Weischedel. 2001. Trec2001 cross-lingual retrieval at bbn. In *TREC 2001*, Gaithersburg: NIST.

J. Xu, A. Fraser, and R. Weischedel. 2002. Empirical studies in strategies for arabic information retrieval. In *SIGIR 2002*, Tampere, Finland.

Classifying Amharic News Text Using Self-Organizing Maps

Samuel Eyassu
Department of Information Science
Addis Ababa University, Ethiopia
samueleya@yahoo.com

Björn Gambäck*
Swedish Institute of Computer Science
Box 1263, SE–164 29 Kista, Sweden
gamback@sics.se

Abstract

The paper addresses using artificial neural networks for classification of Amharic news items. Amharic is the language for countrywide communication in Ethiopia and has its own writing system containing extensive systematic redundancy. It is quite dialectally diversified and probably representative of the languages of a continent that so far has received little attention within the language processing field.

The experiments investigated document clustering around user queries using Self-Organizing Maps, an unsupervised learning neural network strategy. The best ANN model showed a precision of 60.0% when trying to cluster unseen data, and a 69.5% precision when trying to classify it.

1 Introduction

Even though the last years have seen an increasing trend in investigating applying language processing methods to other languages than English, most of the work is still done on very few and mainly European and East-Asian languages; for the vast number of languages of the African continent there still remains plenty of work to be done. The main obstacles to progress in language processing for these are two-fold. Firstly, the peculiarities of the languages themselves might force new strategies to be developed. Secondly, the lack of already available resources and tools makes the creation and testing of new ones more difficult and time-consuming.

Many of the languages of Africa have few speakers, and some lack a standardised written form, both creating problems for building language processing systems and reducing the need for such systems. However, this is not true for the major African languages and as example of one of those this paper takes Amharic, the Semitic language used for countrywide communication in Ethiopia. With more than 20 million speakers, Amharic is today probably one of the five largest on the continent (albeit difficult to determine, given the dramatic population size changes in many African countries in recent years).

The Ethiopian culture is ancient, and so are the written languages of the area, with Amharic using its own script. Several computer fonts for the script have been developed, but for many years it had no standardised computer representation[1] which was a deterrent to electronic publication. An exponentially increasing amount of digital information is now being produced in Ethiopia, but no deep-rooted culture of information exchange and dissemination has been established. Different factors are attributed to this, including lack of digital library facilities and central resource sites, inadequate resources for electronic publication of journals and books, and poor documentation and archive collections. The difficulties to access information have led to low expectations and under-utilization of existing information resources, even though the need for accurate and fast information access is acknowledged as a major factor affecting the success and quality of research and development, trade and industry (Furzey, 1996).

*Author for correspondence.

[1] An international standard for Amharic was agreed on only in year 1998, following Amendment 10 to ISO–10646–1. The standard was finally incorporated into Unicode in year 2000: www.unicode.org/charts/PDF/U1200.pdf

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 71–78,
Ann Arbor, June 2005.

In recent years this has lead to an increasing awareness that Amharic language processing resources and digital information access and storage facilities must be created. To this end, some work has now been carried out, mainly by Ethiopian Telecom, the Ethiopian Science and Technology Commission, Addis Ababa University, the Ge'ez Frontier Foundation, and Ethiopian students abroad. So have, for example, Sisay and Haller (2003) looked at Amharic word formation and lexicon building; Nega and Willett (2002) at stemming; Atelach et al. (2003a) at treebank building; Daniel (Yacob, 2005) at the collection of an (untagged) corpus, tentatively to be hosted by Oxford University's Open Archives Initiative; and Cowell and Hussain (2003) at character recognition.[2] See Atelach et al. (2003b) for an overview of the efforts that have been made so far to develop language processing tools for Amharic.

The need for investigating Amharic information access has been acknowledged by the European Cross-Language Evaluation Forum, which added an Amharic–English track in 2004. However, the task addressed was for accessing an English database in English, with only the original questions being posed in Amharic (and then translated into English). Three groups participated in this track, with Atelach et al. (2004) reporting the best results.

In the present paper we look at the problem of mapping questions posed in Amharic onto a collection of Amharic news items. We use the Self-Organizing Map (SOM) model of artificial neural networks for the task of retrieving the documents matching a specific query. The SOMs were implemented using the Matlab Neural Network Toolbox.

The rest of the paper is laid out as follows. Section 2 discusses artificial neural networks and in particular the SOM model and its application to information access. In Section 3 we describe the Amharic language and its writing system in more detail together with the news items corpora used for training and testing of the networks, while Sections 4 and 5 detail the actual experiments, on text retrieval and text classification, respectively. Finally, Section 6 sums up the main contents of the paper.

[2] In the text we follow the Ethiopian practice of referring to Ethiopians by their given names. However, the reference list follows Western standard and is ordered according to surnames (i.e., the father's name for an Ethiopian).

2 Artificial Neural Networks

Artificial Neural Networks (ANN) is a computational paradigm inspired by the neurological structure of the human brain, and ANN terminology borrows from neurology: the brain consists of millions of neurons connected to each other through long and thin strands called axons; the connecting points between neurons are called synapses.

ANNs have proved themselves useful in deriving meaning from complicated or imprecise data; they can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computational and statistical techniques. Traditionally, the most common ANN setup has been the backpropagation architecture (Rumelhart et al., 1986), a supervised learning strategy where input data is fed forward in the network to the output nodes (normally with an intermediate hidden layer of nodes) while errors in matches are propagated backwards in the net during training.

2.1 Self-Organizing Maps

Self-Organizing Maps (SOM) is an unsupervised learning scheme neural network, which was invented by Kohonen (1999). It was originally developed to project multi-dimensional vectors on a reduced dimensional space. Self-organizing systems can have many kinds of structures, a common one consists of an input layer and an output layer, with feed-forward connections from input to output layers and full connectivity (connections between all neurons) in the output layer.

A SOM is provided with a set of rules of a local nature (a signal affects neurons in the immediate vicinity of the current neuron), enabling it to learn to compute an input-output pairing with specific desirable properties. The learning process consists of repeatedly modifying the synaptic weights of the connections in the system in response to input (activation) patterns and in accordance to prescribed rules, until a final configuration develops. Commonly both the weights of the neuron closest matching the inputs and the weights of its neighbourhood nodes are increased. At the beginning of the training the neighbourhood (where input patterns cluster depending on their similarity) can be fairly large and then be allowed to decrease over time.

2.2 Neural network-based text classification

Neural networks have been widely used in text classification, where they can be given terms and having the output nodes represent categories. Ruiz and Srinivasan (1999) utilize an hierarchical array of backpropagation neural networks for (nonlinear) classification of MEDLINE records, while Ng et al. (1997) use the simplest (and linear) type of ANN classifier, the perceptron. Nonlinear methods have not been shown to add any performance to linear ones for text categorization (Sebastiani, 2002).

SOMs have been used for information access since the beginning of the 90s (Lin et al., 1991). A SOM may show how documents with similar features cluster together by projecting the N-dimensional vector space onto a two-dimensional grid. The radius of neighbouring nodes may be varied to include documents that are weaker related. The most elaborate experiments of using SOMs for document classification have been undertaken using the WEBSOM architecture developed at Helsinki University of Technology (Honkela et al., 1997; Kohonen et al., 2000). WEBSOM is based on a hierarchical two-level SOM structure, with the first level forming histogram clusters of words. The second level is used to reduce the sensitivity of the histogram to small variations in document content and performs further clustering to display the document pattern space.

A Self-Organizing Map is capable of simulating new data sets without the need of retraining itself when the database is updated; something which is not true for Latent Semantic Indexing, LSI (Deerwester et al., 1990). Moreover, LSI consumes ample time in calculating similarities of new queries against all documents, but a SOM only needs to calculate similarities versus some representative subset of old input data and can then map new input straight onto the most similar models without having to recompute the whole mapping.

The SOM model preparation passes through the processes undertaken by the LSI model and the classical vector space model (Salton and McGill, 1983). Hence those models can be taken as particular cases of the SOM, when the neighbourhood diameter is maximized. For instance, one can calculate the LSI model's similarity measure of documents versus queries by varying the SOM's neighbourhood diameter, if the training set is a singular value decomposition reduced vector space. Tambouratzis et al. (2003) use SOMs for categorizing texts according to register and author style and show that the results are equivalent to those generated by statistical methods.

3 Processing Amharic

Ethiopia with some 70 million inhabitants is the third most populous African country and harbours more than 80 different languages.[3] Three of these are dominant: Oromo, a Cushitic language spoken in the South and Central parts of the country and written using the Latin alphabet; Tigrinya, spoken in the North and in neighbouring Eritrea; and Amharic, spoken in most parts of the country, but predominantly in the Eastern, Western, and Central regions. Both Amharic and Tigrinya are Semitic and about as close as are Spanish and Portuguese (Bloor, 1995),

3.1 The Amharic language and script

Already a census from 1994[4] estimated Amharic to be mother tongue of more than 17 million people, with at least an additional 5 million second language speakers. It is today probably the second largest language in Ethiopia (after Oromo). The Constitution of 1994 divided Ethiopia into nine fairly independent regions, each with its own nationality language. However, Amharic is the language for countrywide communication and was also for a long period the principal literal language and medium of instruction in primary and secondary schools in the country, while higher education is carried out in English.

Amharic and Tigrinya speakers are mainly Orthodox Christians, with the languages drawing common roots to the ecclesiastic Ge'ez still used by the Coptic Church. Both languages are written using the Ge'ez script, horizontally and left-to-right (in contrast to many other Semitic languages). Written Ge'ez can be traced back to at least the 4th century A.D. The first versions of the script included consonants only, while the characters in later versions represent consonant-vowel (CV) phoneme pairs. In modern written Amharic, each syllable pat-

[3] How many languages there are in a country is as much a political as a linguistic issue. The number of languages of Ethiopia and Eritrea together thus differs from 70 up to 420, depending on the source; however, 82 (plus 4 extinct) is a common number.

[4] Published by Ethiopia's Central Statistal Authority 1998.

Order	1	2	3	4	5	6	7
C \ V	/ə/	/u/	/i/	/ɐ/	/e/	/ɨ/	/o/
/s/	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
/m/	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ

Table 1: The orders for ስ (/s/) and ም (/m/)

tern comes in seven different forms (called *orders*), reflecting the seven vowel sounds. The first order is the basic form; the other orders are derived from it by more or less regular modifications indicating the different vowels. There are 33 basic forms, giving 7*33 syllable patterns, or *fidEls*.

Two of the base forms represent vowels in isolation (ዐ and አ), but the rest are for consonants (or semivowels classed as consonants) and thus correspond to CV pairs, with the first order being the base symbol with no explicit vowel indicator (though a vowel is pronounced: C+/ə/). The sixth order is ambiguous between being just the consonant or C+/ɨ/. The writing system also includes 20 symbols for labialised velars (four five-character orders) and 24 for other labialisation. In total, there are 275 *fidEls*. The sequences in Table 1 (for ስ and ም) exemplify the (partial) symmetry of vowel indicators.

Amharic also has its own numbers (twenty symbols, though not widely used nowadays) and its own punctuation system with eight symbols, where the space between words looks like a colon ፡, while the full stop, comma and semicolon are ።, ፣ and ፤. The question and exclamation marks have recently been included in the writing system. For more thorough discussions of the Ethiopian writing system, see, for example, Bender et al. (1976) and Bloor (1995).

Amharic words have consonantal roots with vowel variation expressing difference in interpretation, making stemming a not-so-useful technique in information retrieval (no full morphological analyser for the language is available yet). There is no agreed upon spelling standard for compounds and the writing system uses multitudes of ways to denote compound words. In addition, not all the letters of the Amharic script are strictly necessary for the pronunciation patterns of the language; some were simply inherited from Ge'ez without having any semantic or phonetic distinction in modern Amharic: there are many cases where numerous symbols are used to denote a single phoneme, as well as words that have extremely different orthographic form and slightly distinct phonetics, but the same meaning. As a result of this, lexical variation and homophony is very common, and obviously deteriorates the effectiveness of Information Access systems based on strict term matching; hence the basic idea of this research: to use the approximative matching enabled by self-organizing map-based artificial neural networks.

3.2 Test data and preprocessing

In our SOM-based experiments, a corpus of news items was used for text classification. A main obstacle to developing applications for a language like Amharic is the scarcity of resources. No large corpora for Amharic exist, but we could use a small corpus of 206 news articles taken from the electronic news archive of the website of the Walta Information Center (an Ethiopian news agency). The training corpus consisted of 101 articles collected by Saba (Amsalu, 2001), while the test corpus consisted of the remaining 105 documents collected by Theodros (GebreMeskel, 2003). The documents were written using the Amharic software VG2 Main font.

The corpus was matched against 25 queries. The selection of documents relevant to a given query, was made by two domain experts (two journalists), one from the Monitor newspaper and the other from the Walta Information Center. A linguist from Gonder College participated in making consensus of the selection of documents made by the two journalists. Only 16 of the 25 queries were judged to have a document relevant to them in the 101 document training corpus. These 16 queries were found to be different enough from each other, in the content they try to address, to help map from document collection to query contents (which were taken as class labels). These mappings (assignment) of documents to 16 distinct classes helped to see retrieval and classification effectiveness of the ANN model.

The corpus was preprocessed to normalize spelling and to filter out stopwords. One preprocessing step tried to solve the problems with non-standardised spelling of compounds, and that the same sound may be represented with two or more distinct but redundant written forms. Due to the systematic redundancy inherited from the Ge'ez, only about 233 of the 275 *fidEls* are actually necessary to

Sound pattern	Matching Amharic characters
/sə/	ሰ, ሠ
/rə/	ጸ, ፀ
/hə/	ሀ, ሃ, ሐ, ሓ, ኀ, ኃ
/iə/	አ, ኣ, ዐ, ዓ

Table 2: Examples of character redundancy

represent Amharic. Some examples of character redundancy are shown in Table 2. The different forms were reduced to common representations.

A negative dictionary of 745 words was created, containing both stopwords that are news specific and the Amharic text stopwords collected by Nega (Alemayehu and Willett, 2002). The news specific common terms were manually identified by looking at their frequency. In a second preprocessing step, the stopwords were removed from the word collection before indexing. After the preprocessing, the number of remaining terms in the corpus was 10,363.

4 Text retrieval

In a set of experiments we investigated the development of a retrieval system using Self-Organizing Maps. The term-by-document matrix produced from the entire collection of 206 documents was used to measure the retrieval performance of the system, of which 101 documents were used for training and the remaining for testing. After the preprocessing described in the previous section, a weighted matrix was generated from the original matrix using the log-entropy weighting formula (Dumais, 1991). This helps to enhance the occurrence of a term in representing a particular document and to degrade the occurrence of the term in the document collection. The weighted matrix can then be dimensionally reduced by Singular Value Decomposition, SVD (Berry et al., 1995). SVD makes it possible to map individual terms to the concept space.

A query of variable size is useful for comparison (when similarity measures are used) only if its size is matrix-multiplication-compatible with the documents. The pseudo-query must result from the global weight obtained in weighing the original matrix to be of any use in ranking relevant documents. The experiment was carried out in two versions, with the original vector space and with a reduced one.

4.1 Clustering in unreduced vector space

In the first experiment, the selected documents were indexed using 10,363 dimensional vectors (i.e., one dimension per term in the corpus) weighted using log-entropy weighting techniques. These vectors were fed into an Artificial Neural Network that was created using a SOM lattice structure for mapping on a two-dimensional grid. Thereafter a query and 101 documents were fed into the ANN to see how documents cluster around the query.

For the original, unnormalised (unreduced, 10,363 dimension) vector space we did not try to train an ANN model for more than 5,000 epochs (which takes weeks), given that the network performance in any case was very bad, and that the network for the reduced vector space had its apex at that point (as discussed below).

Those documents on the node on which the single query lies and those documents in the immediate vicinity of it were taken as being relevant to the query (the neighbourhood was defined to be six nodes). Ranking of documents was performed using the cosine similarity measure, on the single query versus automatically retrieved relevant documents. The eleven-point average precision was calculated over all queries. For this system the average precision on the test set turned out to be 10.5%, as can be seen in the second column of Table 3.

The table compares the results on training on the original vector space to the very much improved ones obtained by the ANN model trained on the reduced vector space, described in the next section.

Recall	Original vector	Reduced vector
0.00	0.2080	0.8311
0.10	0.1986	0.7621
0.20	0.1896	0.7420
0.30	0.1728	0.7010
0.40	0.0991	0.6888
0.50	0.0790	0.6546
0.60	0.0678	0.5939
0.70	0.0543	0.5300
0.80	0.0403	0.4789
0.90	0.0340	0.3440
1.00	0.0141	0.2710
Average	0.1052	0.5998

Table 3: Eleven-point precision for 16 queries

4.2 Clustering in SVD-reduced vector space

In a second experiment, vectors of numerically indexed documents were converted to weighted matrices and further reduced using SVD, to infer the need for representing co-occurrence of words in identifying a document. The reduced vector space of 101 pseudo-documents was fed into the neural net for training. Then, a query together with 105 documents was given to the trained neural net for simulation and inference purpose.

For the reduced vectors a wider range of values could be tried. Thus 100, 200, ..., 1000 epochs were tried at the beginning of the experiment. The network performance kept improving and the training was then allowed to go on for 2000, 3000, ..., 10,000, 20,000 epochs thereafter. The average classification accuracy was at an apex after 5,000 epochs, as can been seen in Figure 1.

The neural net with the highest accuracy was selected for further analysis. As in the previous model, documents in the vicinity of the query were ranked using the cosine similarity measure and the precision on the test set is illustrated in the third column of Table 3. As can be seen in the table, this system was effective with 60.0% eleven-point average precision on the test set (each of the 16 queries was tested).

Thus, the performance of the reduced vector space system was very much better than that obtained using the test set of the normal term document matrix that resulted in only 10.5% average precision. In both cases, the precision of the training set was assessed using the classification accuracy which shows how documents with similar features cluster together (occur on the same or neighbouring nodes).

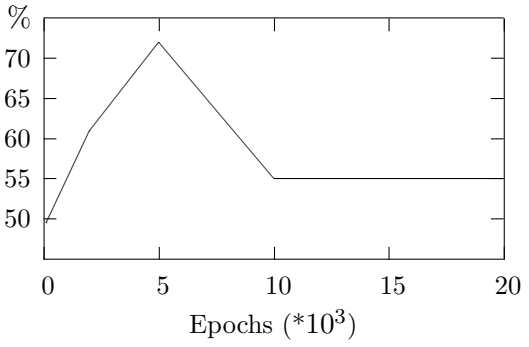

Figure 1: Average network classification accuracy

5 Document Classification

In a third experiment, the SVD-reduced vector space of pseudo-documents was assigned a class label (query content) to which the documents of the training set were identified to be more similar (by experts) and the neural net was trained using the pseudo-documents and their target classes. This was performed for 100 to 20,000 epochs and the neural net with best accuracy was considered for testing.

The average precision on the training set was found to be 72.8%, while the performance of the neural net on the test set was 69.5%. A matrix of simple queries merged with the 101 documents (that had been used for training) was taken as input to a SOM-model neural net and eventually, the 101-dimensional document and single query pairs were mapped and plotted onto a two-dimensional space. Figure 2 gives a flavour of the document clustering.

The results of this experiment are compatible with those of Theodros (GebreMeskel, 2003) who used the standard vector space model and latent semantic indexing for text categorization. He reports that the vector space model gave a precision of 69.1% on the training set. LSI improved the precision to 71.6%, which still is somewhat lower than the 72.8% obtained by the SOM model in our experiments. Going outside Amharic, the results can be compared to the ones reported by Cai and Hofmann (2003) on the Reuters-21578 corpus[5] which contains 21,578 classified documents (100 times the documents available for Amharic). Used an LSI approach they obtained document average precision figures of 88–90%.

In order to locate the error sources in our experiments, the documents missed by the SOM-based classifier (documents that were supposed to be clustered on a given class label, but were not found under that label), were examined. The documents that were rejected as irrelevant by the ANN using reduced dimension vector space were found to contain only a line or two of interest to the query (for the training set as well as for the test set). Also within the test set as well as in the training set some relevant documents had been missed for unclear reasons.

Those documents that had been retrieved as relevant to a query without actually having any relevance to that query had some words that co-occur

[5] Available at `www.daviddlewis.com/resources`

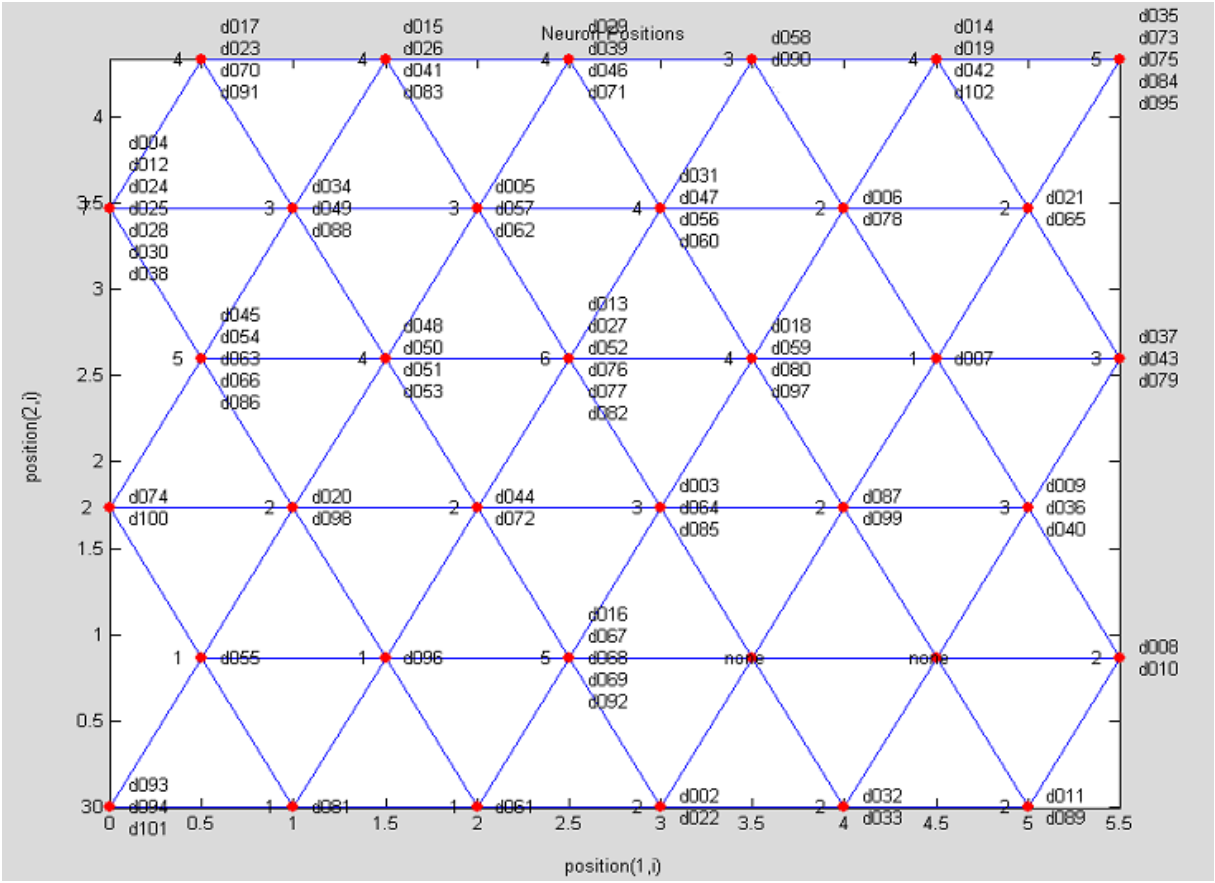

Figure 2: Document clustering at different neuron positions

with the words of the relevant documents. Very important in this observation was that documents that could be of some interest to two classes were found at nodes that are the intersection of the nodes containing the document sets of the two classes.

6 Summary and Conclusions

A set of experiments investigated text retrieval of selected Amharic news items using Self-Organizing Maps, an unsupervised learning neural network method. 101 training set items, 25 queries, and 105 test set items were selected. The content of each news item was taken as the basis for document indexing, and the content of the specific query was taken for query indexing. A term–document matrix was generated and the occurrence of terms per document was registered. This original matrix was changed to a weighted matrix using the log-entropy scheme. The weighted matrix was further reduced using SVD. The length of the query vector was also reduced using the global weight vector obtained in weighing the original matrix.

The ANN model using unnormalised vector space had a precision of 10.5%, whereas the best ANN model using reduced dimensional vector space performed at a 60.0% level for the test set. For this configuration we also tried to classify the data around a query content, taken that query as class label. The results obtained then were 72.8% for the training set and 69.5% for the test set, which is encouraging.

7 Acknowledgments

Thanks to Dr. Gashaw Kebede, Kibur Lisanu, Lars Asker, Lemma Nigussie, and Mesfin Getachew; and to Atelach Alemu for spotting some nasty bugs.

The work was partially funded by the Faculty of Informatics at Addis Ababa University and the ICT support programme of SAREC, the Department for Research Cooperation at Sida, the Swedish International Development Cooperation Agency.

References

Nega Alemayehu and Peter Willett. 2002. Stemming of Amharic words for information retrieval. *Literary and Linguistic Computing*, 17(1):1–17.

Atelach Alemu, Lars Asker, and Gunnar Eriksson. 2003a. An empirical approach to building an Amharic treebank. In *Proc. 2nd Workshop on Treebanks and Linguistic Theories*, Växjö University, Sweden.

Atelach Alemu, Lars Asker, and Mesfin Getachew. 2003b. Natural language processing for Amharic: Overview and suggestions for a way forward. In *Proc. 10th Conf. Traitement Automatique des Langues Naturelles*, Batz-sur-Mer, France, pp. 173–182.

Atelach Alemu, Lars Asker, Rickard Cöster, and Jussi Karlgren. 2004. Dictionary-based Amharic–English information retrieval. In *5th Workshop of the Cross Language Evaluation Forum*, Bath, England.

Saba Amsalu. 2001. The application of information retrieval techniques to Amharic. MSc Thesis, School of Information Studies for Africa, Addis Ababa University, Ethiopia.

Marvin Bender, Sydney Head, and Roger Cowley. 1976. The Ethiopian writing system. In Bender et al., eds, *Language in Ethiopia*. Oxford University Press.

Michael Berry, Susan Dumais, and Gawin O'Brien. 1995. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595.

Thomas Bloor. 1995. The Ethiopic writing system: a profile. *Journal of the Simplified Spelling Society*, 19:30–36.

Lijuan Cai and Thomas Hofmann. 2003. Text categorization by boosting automatically extracted concepts. In *Proc. 26th Int. Conf. Research and Development in Information Retrieval*, pp. 182–189, Toronto, Canada.

John Cowell and Fiaz Hussain. 2003. Amharic character recognition using a fast signature based algorithm. In *Proc. 7th Int. Conf. Image Visualization*, pp. 384–389, London, England.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Susan Dumais. 1991. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23(2):229–236.

Sisay Fissaha and Johann Haller. 2003. Application of corpus-based techniques to Amharic texts. In *Proc. MT Summit IX Workshop on Machine Translation for Semitic Languages*, New Orleans, Louisana.

Jane Furzey. 1996. Enpowering socio-economic development in Africa utilizing information technology. A country study for the United Nations Economic Commission for Africa, University of Pennsylvania.

Theodros GebreMeskel. 2003. Amharic text retrieval: An experiment using latent semantic indexing (LSI) with singular value decomposition (SVD). MSc Thesis, School of Information Studies for Africa, Addis Ababa University, Ethiopia.

Timo Honkela, Samuel Kaski, Krista Lagus, and Teuvo Kohonen. 1997. WEBSOM — Self-Organizing Maps of document collections. In *Proc. Workshop on Self-Organizing Maps*, pp. 310–315, Espoo, Finland.

Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojärvi, Jukka Honkela, Vesa Paatero, and Antti Saarela. 2000. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585.

Teuvo Kohonen. 1999. *Self-Organization and Associative Memory*. Springer, 3 edition.

Xia Lin, Dagobert Soergel, and Gary Marchionini. 1991. A self-organizing semantic map for information retrieval. In *Proc. 14th Int. Conf. Research and Development in Information Retrieval*, pp. 262–269, Chicago, Illinois.

Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proc. 20th Int. Conf. Research and Development in Information Retrieval*, pp. 67–73, Philadelphia, Pennsylvania.

Miguel Ruiz and Padmini Srinivasan. 1999. Hierarchical neural networks for text categorization. In *Proc. 22nd Int. Conf. Research and Development in Information Retrieval*, pp. 281–282, Berkeley, California.

David Rumelhart, Geoffrey Hinton, and Ronald Williams. 1986. Learning internal representations by error propagation. In Rumelhart and McClelland, eds, *Parallel Distributed Processing*, vol 1. MIT Press.

Gerard Salton and Michael McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

George Tambouratzis, N. Hairetakis, S. Markantonatou, and G. Carayannis. 2003. Applying the SOM model to text classification according to register and stylistic content. *Int. Journal of Neural Systems*, 13(1):1–11.

Daniel Yacob. 2005. Developments towards an electronic Amharic corpus. In *Proc. TALN 12 Workshop on NLP for Under-Resourced Languages*, Dourdan, France, June (to appear).

Arabic Diacritization Using Weighted Finite-State Transducers

Rani Nelken and **Stuart M. Shieber**
Division of Engineering and Applied Sciences
Harvard University
33 Oxford St.
Cambridge, MA 02138
{nelken,shieber}@deas.harvard.edu

Abstract

Arabic is usually written without short vowels and additional diacritics, which are nevertheless important for several applications. We present a novel algorithm for restoring these symbols, using a cascade of probabilistic finite-state transducers trained on the Arabic treebank, integrating a word-based language model, a letter-based language model, and an extremely simple morphological model. This combination of probabilistic methods and simple linguistic information yields high levels of accuracy.

Introduction

Most semitic languages in both ancient and contemporary times are usually written without short vowels and other diacritic marks, often leading to potential ambiguity. While such ambiguity only rarely impedes proficient speakers, it can certainly be a source of confusion for beginning readers and people with learning disabilities (Abu-Rabia, 1999). The problem becomes even more acute when people are required to actively generate diacritized script, as used for example in poetry or children's books. Diacritization is even more problematic for computational systems, adding another level of ambiguity to both analysis and generation of text. For example, full vocalization is required for text-to-speech applications, and has been shown to improve speech-recognition perplexity and error rate (Kirchoff et al., 2002).

We present a system for Arabic diacritization[1] in Modern Standard Arabic (MSA) using weighted finite-state transducers. The system is constructed using standardly available finite-state tools, and encodes only minimal morphological knowledge, yet achieves very high levels of accuracy. While the methods described in this paper are applicable to additional semitic languages, the choice of MSA was motivated by the availability of the Arabic Treebank, distributed by the Linguistic Data Consortium (LDC), a sizable electronic corpus of diacritized text, which we could use for training and testing. Such resources are rare for other semitic languages.

This paper is structured as follows. In Section 1, we describe the task, including a brief introduction to Arabic diacritization, and the corpus we used. In Section 2, we describe the design of our system, which consists of a trigram word-based language model, augmented by two extensions: an extremely simple morphological analyzer and a letter-based language model, which are used to address the data sparsity problem. In Section 3, we report the system's experimental evaluation. We review related work in Section 4, and close with conclusions and directions for future research in Section 5.

1 The task

The Arabic vowel system consists of 3 short vowels and 3 long vowels, as summarized in Ta-

[1] We distinguish between vocalization—the restoration of vowel symbols—and diacritization, which includes additionally the restoration of a richer system of diacritic marks, as explained below.

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 79–86,
Ann Arbor, June 2005.

Short vowels				
Vocalized		Unvoc.		Pronounc.
Arab.	*Tran.*	*Arab.*	*Tran.*	
ـُ	u	-	-	/u/
ـَ	a	-	-	/a/
ـِ	i	-	-	/i/
Long vowels				
ـُو	uw	و	w	/u:/
ـَا	aA(1),(2)	ا	A	/a:/
ـَى	aY	ى	Y(3)	/a:/
ـِي	iy	ي	y	/i:/
Doubled case endings				
ـٌ	N	-	-	/un/
ـً	F	-	-	/an/
ـًا	AF	ا	A	/an/
ـٍ	K	-	-	/in/

(1) aA may also appear as A even in vocalized text.
(2) In some lexical items, aA is written as ‘; in which case it is dropped in undiacritized text.
(3) Y and y can appear interchangeably in the corpus (Buckwalter, 2004).

Table 1: Arabic vowels

ble 1.[2] Short vowels are written as symbols either above or below the letter in diacritized text, and dropped altogether in undiacritized text. Long vowels are written as a combination of a short vowel symbol, followed by a vowel letter; in undiacritized text, the short vowel symbol is dropped. Arabic also uses vowels at the end of words to mark case distinctions, which include both the short vowels and an additional doubled form ("tanween").

Diacritized text also contains two syllabification marks: بّ (trans.: ~) denoting doubling of the preceding consonant (in this case, ب), and بْ (trans.: o), denoting the lack of a vowel.

The Arabic glottal stop ("hamza") deserves special mention, as it can appear in several different forms in diacritized text, enumerated in

[2]Throughout the paper we follow Buckwalter's transliteration of Arabic into 7-bit ASCII characters (2002a).

Arab.	*Tran.*	*Arab.*	*Tran.*
أ	>	آ	|
إ	<	ء	'
ؤ	&	آ	{
ئ	}		

Table 2: Arabic glottal stop

Table 2. In undiacritized text, it appears either as A or as one of the forms in the table.

We used the LDC's Arabic Treebank of diacritized news stories (Part 2). The corpus consists of 501 news stories collected from Al-Hayat for a total of 144,199 words. In addition to diacritization, the corpus contains several types of annotations. After being transliterated, it was morphologically analyzed using the Buckwalter morphological analyzer (2002b). Buckwalter's system provides a list of all possible analyses of each word, including a full diacritization, and a part-of-speech (POS) tag. From this candidate list, annotators chose a single analysis. Afterwards, clitics, which are prevalent in Arabic were separated and marked, and a full parse-tree was manually generated. For our purposes, we have stripped all the POS tagging, to retain two versions of each file—diacritized and undiacritized, which we use for both training and evaluation as will be explained below.

An example sentence fragment in Arabic is given in Figure 1 in three forms: undiacritized, diacritized without case endings, and with case endings.

رد الأمين العام لجامعة الدول العربية عمرو موسى...
رَدَّ الأَمِين العَامّ لِجَامِعَة الدُوَل العَرَبِيَّة عَمْرو مُوسَى...
رَدَّ الأَمِينُ العَامُّ لِجَامِعَةِ الدُوَلِ العَرَبِيَّةِ عَمْرو مُوسَى...

Figure 1: Example sentence fragment

The transliteration and translation are given in Figure 2. We follow precisely the form that appears in the Arabic treebank.[3]

[3]The diacritization of this example is (strictly speaking) incomplete with respect to the diacritization of the

```
rd     Al>myn    AlEAm   ljAmEp     Aldwl    AlErbyp        Emrw   mwsY
rad~a Al>amiyn  AlEAm~  lijAmiEap  Alduwal  AlEarabiy~ap  Eamorw muwsaY
rad~a Al>amiynu AlEAm~u lijAmiEapu Alduwali AlEarabiy~api Eamorw muwsaY
```

"Arab League Secretary General Amr Mussa replied..."

Figure 2: Transliteration and translation of the sentence fragment

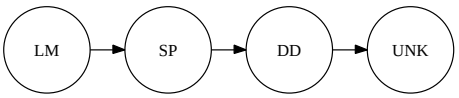

Figure 3: Basic model

2 System design

To restore diacritization, we have created a generative probabilistic model of the process of losing diacritics, expressed as a finite-state transducer. The model transduces fully diacritized Arabic text, weighted according to a language model, into undiacritized text. To restore diacritics, we use Viterbi decoding, a standard algorithm for efficiently computing the best path through an automaton, to reconstruct the maximum likelihood diacritized word sequence that would generate a given undiacritized word sequence.

The model is constructed as a composition of several weighted finite-state transducers (Pereira and Riley, 1997). Transducers are extremely well-suited for this task as their closure under composition allows complex models to be efficiently and elegantly constructed from modular implementations of simpler models.

The system is implemented using the AT&T FSM and GRM libraries (Mohri et al., 2000; Allauzen et al., 2003), which provide a collection of useful tools for constructing weighted finite-state transducers implementing language models.

2.1 Basic model

Our cascade consists of the following transducers, illustrated in Figure 3.

Language model (LM) A standard trigram language model of Arabic diacritized words. For smoothing purposes, we use Katz backoff (Katz, 1987). We learn weights for the model from a training set of diacritized words. This is the only component of the basic model for which we learn such weights. During decoding, these weights are utilized to choose the most probable word sequence that could have generated the undiacritized text. The model also includes special symbols for unknown words, ⟨UNK⟩, and for numbers, ⟨NUM⟩, as explained below.

Spelling (SP) A spelling transducer that transduces a word into its component letters. This is a technical necessity since the language model operates on word tokens and the following components operate on letter tokens. For instance the single token `Al>amiyn` is transduced to the sequence of tokens `A,l,>,a,m,i,y,n`.

Diacritic drop (DD) A transducer for dropping vowels and other diacritics. The transducer simply replaces all short vowel symbols and syllabification marks with the empty string, ϵ. In addition, this transducer also handles the multiple forms of the glottal stop (see Section 1). Rather than encoding any morphological rules on when the glottal stop receives each form, we merely encode the generic availability of these various alternatives, as transductions. For instance, DD includes the option of transducing { to `A`, without any information on when such transduction should take place.

Unknowns (UNK) Due to data sparsity, a test input may include words that did not appear in the training data, and will thus be unknown to the language model. To handle such words, we add a transducer, UNK, that transduces ⟨UNK⟩, into a stochastic sequence of arbitrary letters. During decoding, the letter se-

determiner `Al` and the letter immediately following it. For instance, in `Alduwal`, the `d` should actually have been doubled, yielding `Ald~uwal`. The treebank consistently does not diacritize `Al`, and we adhere to its conventions in both training and testing.

quence is fixed, and since it has no possible diacritization in the model, Viterbi decoding would choose ⟨UNK⟩ as its most likely generator.

UNK plays a similar purpose in handling numbers. UNK transduces ⟨NUM⟩ to a stochastically generated sequence of digits. In the training data, we replace all numbers with ⟨NUM⟩. On encountering a number in a test input, the decoding algorithm would replace the number with ⟨NUM⟩. As a post-processing step, we replace all occurrences of ⟨UNK⟩ and ⟨NUM⟩ with the original input word/number.

2.2 Handling clitics

Arabic contains numerous clitics, which are appended to words, either as prefixes or as suffixes, including the determiner, conjunctions, some prepositions, and pronouns. Clitics pose an important challenge for an n-gram model, since the same word with a different clitic combination would appear to the model as a separate token. We thus augment our basic model with a transducer for handling clitics.

Handling clitics using a rule-based approach is a non-trivial undertaking (Buckwalter, 2002b). In addition to cases of potential ambiguity between letters belonging to a clitic and letters belonging to a word, clitics might be iteratively appended, but only in some combinations and some orderings. Buckwalter maintains a dictionary not only of all prefixes, stems, and suffixes, but also keeps a separate dictionary entry for each allowed combination of diacritized clitics. Since, unlike Buckwalter's system, we are interested just in the most probable clitic separation rather than the full set of analyses, we implement only a very simple transducer, and rely on the probabilistic model to handle such ambiguities and complexities.

From a generative perspective, we assume that the hypothetical original text from which the model starts is not only diacritized, but also has clitics separated. We augment the model with a transducer, Clitic Concatenation (CC), which non-deterministically concatenates clitics to words. CC scans the letter stream; on encountering a potential prefix, CC can non-deterministically append it to the following

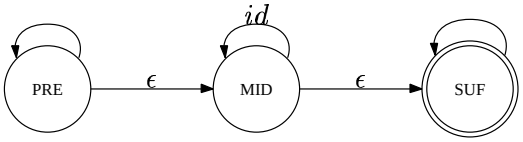

Figure 4: Clitic concatenation

word, merely by transducing the space following it to ϵ. This is done iteratively for each prefix. After concatenating prefixes, CC can non-deterministically decide that it has reached the main word, which it copies. Finally, it concatenates suffixes symmetrically to prefixes. For instance, on encountering the letter string `w Al >myn`, CC might drop the spaces to generate `wAl>myn` ("and the secretary").

The transducer implementation of CC consists of three components, depicted schematically in Figure 4. The first component iteratively appends prefixes. For each of a fixed set of prefixes, it has a set of states for identifying the prefix and dropping the trailing space. A non-deterministic jump moves the transducer to the middle component, which implements the identity function on letters, copying the putative main word to the output. Finally, CC can non-deterministically jump to the final component, which appends suffixes by dropping the preceding space.

By design, CC provides a very simple model of Arabic clitics. It maintains just a list of possible prefixes and suffixes, but encodes no information about stems or possible clitic orderings, potentially allowing many ungrammatical combinations. We rely on the probabilistic language model to assign such combinations very low probabilities.[4]

We use the following list of (undiacritized) clitics: (cf. Diab et al. (2004) who use the same set with the omission of `s`, and `ny`):

prefixes: `b` (by/with), `l` (to), `k` (as), `w` (and), `f` (and), `Al` (the), `s` (future);

suffixes: `y` (my/mine), `ny` (me), `nA` (our/ours), `k` (your/yours), `kmA` (your/yours

[4]The only special case of multiple prefix combinations that we explicitly encode is the combination of l+Al (to + the) which becomes ll, by dropping the A.

masc. dual), km (your/yours masc. pl.), knA (your/yours fem. dual), kn (your/yours fem. pl.), h (him/his), hA (her/hers), hmA (their/theirs masc. dual), hnA (their/theirs fem. dual), hm (their/theirs masc. pl.), hn (their/theirs fem. pl).

We integrate CC into the cascade by composing it after DD, and before UNK. Thus, clitics appear in their undiacritized form. Our model now assumes that the diacritized input text has clitics separated. This requires two changes to our method. First, training must now be performed on text in which clitics are separated. This is straightforward since clitics are tagged in the corpus. Second, in the undiacritized test data, we keep clitics intact. Running Viterbi decoding on the augmented model would not only diacritize it, but also separate clitics. To generate grammatical Arabic, we reconnect the clitics as a post-processing step. We use a greedy strategy of connecting each prefix to the following word, and each suffix to the preceding word.[5]

While our handling of clitics helps overcome data sparsity, there is also a potential cost for decoding. Clitics, which are, intuitively speaking, less informative than regular words, are now treated as lexical items of "equal stature". For instance, a bigram model may include the collocation Al>amiyn AlEAm~ (the secretary general). Once clitics are separated this becomes Al >amiyn Al EAm~. A bigram model would no longer retain the connection between each of the main words, >amiyn and EAm~, but only between them and the determiner Al, which is potentially less informative.

Figure 5 shows an example transduction through the word-based model, where for illustration purposes, we assume that Aldwl is an unknown word.

2.3 Letter model for unknown words

To diacritize unknown words, we trained a letter-based 4-gram language model of Arabic words, LLM, on the letter sequences of words in the training set. Composing LLM with the vowel-drop transducer, DD, yields a probabilistic generative model of Arabic letter and diacritization patterns, including for words that were never encountered in training.

In principle, we could use the letter model as an alternative model of the full text, but we found it more effective to use it selectively, only on unknown words. Thus, after running the word-based language model, we extract all the words tagged as ⟨UNK⟩ and run the letter-based model on them. Here is an example transduction:

Diacritized	Alduwal
LLM	(Weighted)
DD	(Diacritics dropped) Aldwl

We chose not to apply any special clitic handling for the letter-based model. To see why, consider the alternative model that would include CC. Since LLM is unaware of word tokens, there is no pressure on the decoding algorithm to split the clitics from the word, and clitics may therefore be incorrectly vocalized.

3 Experiments

We randomly split the set of news articles in each of the two parts of the Arabic treebank into a training and held-out testing set of sizes 90% and 10% respectively. We trained both the word-based and the letter-based language models on the diacritized version of the training set. We then ran Viterbi decoding on the undiacritized version of the testing set, which consists of a total of over 14,000 words. As a baseline, we used a unigram word model without clitic handling, constructed using the same transducer technology. We ran two batches of experiments—one in which case endings were stripped throughout the training and testing data, and we did not attempt to restore them, and one in which case markings were included.

Results are reported in Table 3. For each model, we report two measures: the word error rate (WER), and the diacritization error rate (DER), i.e., the proportion of incorrectly restored diacritics.

Surprisingly, a trigram word-based language

[5] The only case that requires special attention is ka which can be either a prefix (meaning "as") or a suffix (meaning "your/yours" masc.). The greedy strategy always chooses the suffix meaning. We correct it by comparison with the input text.

Diacritized	`rad~a Al >amiyn Al EAm~ li jAmiEap` ⟨UNK⟩
LM	(Weighted, but otherwise unchanged)
SP	(Change in token resolution from words to letters)
DD	(Diacritics dropped) `rd Al >myn Al EAm l jAmEp`
CC	(Clitics concatenated) `rd Al>myn AlEAm ljAmEp`
UNK	(⟨UNK⟩ becomes `Aldwl`) `rd Al>myn AlEAm ljAmEp Aldwl`

Figure 5: Example transduction

Model	*without case*		*with case*	
	WER	*DER*	*WER*	*DER*
Baseline	15.48%	17.33%	30.39%	24.03%
3-gram word	14.64%	16.9%	28.42%	23.34%
3-gram word + CC	8.49%	9.32%	24.22%	15.36%
3-gram word + CC + 4-gram letter	7.33%	6.35%	23.61%	12.79%

Table 3: Results on the Al-Hayat corpus

model yields only a modest improvement over the baseline unigram model. The addition of a clitic connection model and a letter-based language model leads to a marked improvement in both WER and DER. This trend is repeated for both variants of the task—either with or without case endings. Including case information naturally yields proportionally worse accuracy. Since case markings encode higher-order grammatical information, they would require a more powerful grammatical model than offered by finite-state methods.

To illustrate the system's performance, here are some decodings made by the different versions of the model.

Basic model

- `An`, `<in~a`, and `>an~a`, three versions of the word "that", which may all appear as `An` in undiacritized text, are often confused. As Buckwalter (2004) notes, the corpus itself is sometimes inconsistent about the use of `<in~a` and `>an~a`.
- Several of the third-person possessive pronoun clitics can appear either with a `u` or an `i`, for instance, the third person singular masculine possessive can appear as either `hu` or `hi`. The correct form depends on the preceding letter and vowel (including the case vowels). Part of the tradeoff of treating clitics as independent lexical items is that the word-based model is ignorant of the letter preceding a suffix clitic.

Clitic model

- `wstkwn` was correctly decoded to `wa sa takuwnu`, which after post-processing becomes `wasatakuwnu` ("and [it] shall be").

Letter model

- `AstfzAz` correctly decoded to {`isotifozAz` ("instigation"). This example is interesting, since a morphological analysis would deterministically predict this diacritization. The probabilistic letter model was able to correctly decode it even though it has no explicit encoding of such knowledge.
- Non-Arabic names are obviously problematic for the model. For instance `bwrtlAnd` was incorrectly decoded to `buwrotilAnoda` rather than `buwrotlAnod` (Portland), but note that some of the diacritics were correctly restored. Al-Onaizan and Knight (2002) proposed a transducer for modeling the Arabic spelling of such names for the purpose of translating from Arabic. Such a model could be seamlessly integrated into our architecture, for improved accuracy.

4 Related work

Gal (2002) constructed an HMM-based bigram model for restoring vowels (but not additional diacritics) in Arabic and Hebrew. For Arabic, the model was applied to the Qur'an, a corpus of about 90,000 words, achieving 14% WER. The word-based language model component of our system is very similar to Gal's HMM. The very flexible framework of transducers allows us to easily enhance the model with our simple but effective morphology handler and letter-based language model.

Several commercial tools are available for Arabic diacritization, which unfortunately we did not have access to. Vergyri and Kirchhoff (2004) evaluated one (unspecified) system on two MSA texts, reporting a 9% DER without case information, and 28% DER with case endings.

Kirchoff et al. (2002) focuses on vocalizing transcripts of oral conversational Arabic. Since conversational Arabic is much more free-flowing, and prone to dialect and speaker differences, diacritization of such transcripts proves much more difficult. Kirchoff et al. started from a unigram model, and augmented it with the following heuristic. For each unknown word, they search for the closest known unvocalized word to it according to Levenshtein distance, and apply whatever transformation that word undergoes, yielding 16.5% WER. Our letter-based model provides an alternative method of generalizing the diacritization from known words to unknown ones.

Vergyri and Kirchhoff (2004) also handled conversational Arabic, and showed that some of the complexity inherent in vocalizing such text can be offset by combining information on the acoustic signal with morphological and contextual information. They treat the latter problem as an *unsupervised* tagging problem, where each word is assigned a tag representing one of its possible diacritizations according to Buckwalter's morphological analyzer (2002b). They use Expectation Maximization (EM) to train a trigram model of tag sequences. The evaluation shows that the combined model yields a significant improvement over just the acoustic model.

5 Conclusions and future directions

We have presented an effective probabilistic finite-state architecture for Arabic diacritization. The modular design of the system, based on a composition of simple and compact transducers allows us to achieve high levels of accuracy while encoding extremely limited morphological knowledge. In particular, while our system is aware of the existence of Arabic clitics, it has no explicit knowledge of how they can be combined. Such patterns are automatically learned from the training data. Likewise, while the system is aware of different orthographic variants of the glottal stop, it encodes no explicit rules to predict their distribution.

The main resource that our method relies on is the existence of sufficient quantities of diacritized text. Since semitic languages are typically written without vowels, it is rare to find sizable collections of diacritized text in digital form. The alternative is to diacritize text using a combination of manual annotation and computational tools. This is precisely the process that was followed in the compilation of the Arabic treebank, and similar efforts are now underway for Hebrew (Wintner and Yona, 2003).

In contrast to morphological analyzers, which usually provide only an unranked list of all possible analyses, our method provides the most probable analysis, and with a trivial extension, could provide a ranked n-best list. Reducing and ranking the possible analyses may help simplify the annotator's job. The burden of requiring large quantities of diacritized text could be assuaged by iterative bootstrapping—training the system and manually correcting it on corpora of increasing size.

As another future direction, we note that occasionally one may find a vowel or two, even in otherwise undiacritized text fragments. This is especially true for extremely short text fragments, where ambiguity is undesirable, as in banners or advertisements. This raises an interesting optimization problem—what is the least number of vowel symbols that are required in

order to ensure an unambiguous reading, and where should they be placed? Assuming that the errors of the probabilistic model are indicative of the types of errors that a human might make, we can use this model to predict where disambiguating vowels would be most informative. A simple change to the model described in this paper would make vowel drop optional rather than obligatory. Such a model would then be able to generate not only fully unvocalized text, but also partially vocalized variants of it. The optimization problem would then become one of finding the partially diacritized text with the minimal number of vowels that would be least ambiguous.

Acknowledgments

We thank Ya'akov Gal for his comments on a previous version of this paper. This work was supported in part by grant IIS-0329089 from the National Science Foundation.

References

Salim Abu-Rabia. 1999. The effect of Arabic vowels on the reading comprehension of second- and sixth-grade native Arab children. *Journal of Psycholinguist Research*, 28(1):93–101, January.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in Arabic texts. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 34–46, Philadelphia, July. Association for Computational Linguistics.

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'2003)*, pages 40–47.

Tim Buckwalter. 2002a. Arabic transliteration table. http://www.qamus.org/transliteration.htm.

Tim Buckwalter. 2002b. Buckwalter Arabic morphological analyzer version 1.0. Linguistic Data Consortium, catalog number LDC2002L49 and ISBN 1-58563-257-0.

Tim Buckwalter. 2004. Issues in Arabic orthography and morphology analysis. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 31–34, Geneva, Switzerland, August 28th. COLING.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In Susan Dumais, Daniel Marcu, and Salim Roukos, editors, *HLT-NAACL 2004: Short Papers*, pages 149–152, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Ya'akov Gal. 2002. An HMM approach to vowel restoration in Arabic and Hebrew. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 27–33, Philadelphia, July. Association for Computational Linguistics.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–4001, March.

Katrin Kirchoff, Jeff Bilmes, Sourin Das, Nicolae Duta, Melissa Egan, Gang Ji, Feng He, John Henderson, Daben Liu, Mohamed Noamany, Pat Schone, Richard Schwarta, and Dimitra Vergyri. 2002. Novel approaches to Arabic speech recognition: report from the 2002 Johns-Hopkins summer workshop. Technical report, Johns Hopkins University.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

Fernando C. N. Pereira and Michael Riley. 1997. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Devices for Natural Language Processing*. MIT Press, Cambridge, MA.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic diacritization of Arabic for acoustic modeling in speech recognition. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 66–73, Geneva, Switzerland, August 28th. COLING.

Shuly Wintner and Shlomo Yona. 2003. Resources for processing Hebrew. In *Proceedings of the MT Summit IX Workshop on Machine Translation for Semitic Languages*, New Orleans, September.

An Integrated Approach for Arabic-English Named Entity Translation

Hany Hassan
IBM Cairo Technology Development Center
Giza - Egypt
P.O. Box 166 Al-Ahram
hanyh@eg.ibm.com

Jeffrey Sorensen
IBM T.J. Watson Research Center
Yorktown Heights
NY 10598
sorenj@us.ibm.com

Abstract

Translation of named entities (NEs), such as person names, organization names and location names is crucial for cross lingual information retrieval, machine translation, and many other natural language processing applications. Newly named entities are introduced on daily basis in newswire and this greatly complicates the translation task. Also, while some names can be translated, others must be transliterated, and, still, others are mixed. In this paper we introduce an integrated approach for named entity translation deploying phrase-based translation, word-based translation, and transliteration modules into a single framework. While Arabic based, the approach introduced here is a unified approach that can be applied to NE translation for any language pair.

1 Introduction

Named Entities (NEs) translation is crucial for effective cross-language information retrieval (CLIR) and for Machine Translation. There are many types of NE phrases, such as: person names, organization names, location names, temporal expressions, and names of events. In this paper we only focus on three categories of NEs: person names, location names and organization names, though the approach is, in principle, general enough to accommodate any entity type.

NE *identification* has been an area of significant research interest for the last few years. NE translation, however, remains a largely unstudied problem. NEs might be phonetically transliterated (e.g. persons names) and might also be mixed between phonetic transliteration and semantic translation as the case with locations and organizations names.

There are three distinct approaches that can be applied for NE translation, namely: a transliteration approach, a word based translation approach and a phrase based translation approach. The transliteration approach depends on phonetic transliteration and is only appropriate for out of vocabulary and completely unknown words. For more frequently used words, transliteration does not provide sophisticated results. A word based approach depends upon traditional statistical machine translation techniques such as IBM Model1 (Brown et al., 1993) and may not always yield satisfactory results due to its inability to handle difficult many-to-many phrase translations. A phrase based approach could provide a good translation for frequently used NE phrases though it is inefficient for less frequent words. Each of the approaches has its advantages and disadvantages.

In this paper we introduce an integrated approach for combining phrase based NE translation, word based NE translation, and NE transliteration in a single framework. Our approach attempts to harness the advantages of the three approaches while avoiding their pitfalls. We also introduce and evaluate a new approach for aligning NEs across parallel corpora, a process for automatically extracting new NEs translation phrases, and a new transliteration approach. As is typical for statistical MT, the system requires the availability of general parallel corpus and Named Entity identifiers for the NEs of interest.

Our primary focus in this paper is on translating NEs out of context (i.e. NEs are extracted and translated without any contextual clues). Although

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 87–93,
Ann Arbor, June 2005.

this is a more difficult problem than translating NEs in context, we adopt this approach because it is more generally useful for CLIR applications.

The paper is organized as follows, section 2 presents related work, section 3 describes our integrated NE translation approach, section 4 presents the word based translation module, the phrase based module, the transliteration module, and system integration and decoding, section 5 provides the experimental setup and results and finally section 6 concludes the paper.

2 Related Work

The Named Entity translation problem was previously addressed using two different approaches: Named Entity phrase translation (which includes word-based translation) and Named Entity transliteration. Recently, many NE phrase translation approaches have been proposed. Huang et al. (Huang et al., 2003) proposed an approach to extract NE trans-lingual equivalences based on the minimization of a linearly combined multi-feature cost. However this approach used a bilingual dictionary to extract NE pairs and deployed it iteratively to extract more NEs. Moore (Moore, 2003), proposed an approach deploying a sequence of cost models. However this approach relies on orthographic clues, such as strings repeated in the source and target languages and capitalization, which are only suitable for language pairs with similar scripts and/or orthographic conventions.

Most prior work in Arabic-related transliteration has been developed for the purpose of machine translation and for Arabic-English transliteration in particular. Arbabi (Arbabi et al., 1998) developed a hybrid neural network and knowledge-based system to generate multiple English spellings for Arabic person names. Stalls and Knight (Stalls and Knight, 1998) introduced an approach for Arabic-English back transliteration for names of English origin; this approach could only back transliterate to English the names that have an available pronunciation. Al-Onaizan and Knight (Al-Onaizan and Knight, 2002) proposed a spelling-based model which directly maps English letter sequences into Arabic letter sequences. Their model was trained on a small English Arabic names list without the need for English pronunciations. Although this method does not require the availability of English pronunciation, it has a serious limitation because it does not provide a mechanism for inserting the omitted short vowels in Arabic names. Therefore it does not perform well with names of Arabic origin in which short vowels tend to be omitted.

3 Integrated Approach for Named Entity Translation

We introduce an integrated approach for Named Entity (NE) translation using phrase based translation, word based translation and transliteration approaches in a single framework. Our unified approach could handle, in principle, any NE type for any languages pair.

The level of complication in NE translation depends on the NE type, the original source of the names, the standard *de facto* translation for certain named entities and the presence of acronyms. For example persons names tend to be phonetically transliterated, but different sources might use different transliteration styles depending on the original source of the names and the idiomatic translation that has been established. Consider the following two names:

"جاك شيراك : jAk $yrAk" → "Jacques Chirac"
"جاك سترو :jAk strw" → "Jack Straw"

Although the first names in both examples are the same in Arabic, their transliterations should be different. One might be able to distinguish between the two by looking at the last names. This example illustrates why transliteration may not be good for frequently used named entities. Transliteration is more appropriate for unknown NEs.

For locations and organizations, the translation can be a mixture of translation and transliteration. For example:

شركة مايكروسوفت :$rkp mAykrwswft → Microsoft Company
القدس : Alqds → Jerusalem
مطار طوكيو : mTAr Tokyw → Tokyo Airport

These examples highlight some of the complications of NE translation that are difficult to overcome using any phrase based, word based or transliteration approach independently. An approach that integrates phrase and word based translation with transliteration in a systematic and flexible framework could provide a more complete solution to the problem.

Our system utilizes a parallel corpus to separately acquire the phrases for the phrase based sys-

tem, the translation matrix for the word based system, and training data for the transliteration system. More details about the three systems will be presented in the next section. Initially, the corpus is automatically annotated with NE types in the source and target languages using NE identifiers similar to the systems described in (Florian et al., 2004) for NE detection.

4 Translation and Transliteration Modules

4.1 Word Based NE Translation

- **Basic multi-cost NE Alignment**

We introduce a novel NE alignment technique to align NEs from a parallel corpus that has been automatically annotated with NE types for source and target languages. We use IBM Model1, as introduced in (Brown et. al, 1993), with a modified alignment cost. The cost function has some similarity with the multi-cost aligning approach introduced by Huang (Huang et al. 2003) but it is significantly different. The cost for aligning any source and target NE word is defined as:

$$C = \lambda_1 p(w_e \mid w_f) + \lambda_2 Ed(w_e, w_f) + \lambda_3 Tag(w_e, w_f)$$

Where: w_e and w_f are the target and source words respectively and λ_1, λ_2 and λ_3 are the cost weighting parameters.

The first term $p(w_e \mid w_f)$ represents the translation log probability of target word (w_e) given the source word (w_f). The second term $Ed(w_e, w_f)$ is length-normalized phonetic based edit distance between the two words. This phonetic-based edit distance employs an Editex style (Zobel and Dart, 1996) distance measure, which groups letters that can result in similar pronunciations, but doesn't require that the groups be disjoint, and can thus reflect the correspondences between letters with similar pronunciation more accurately. The Editex distance (d) between two letters a and b is:

$d(a,b)$ = 0 if both are identical
= 1 if they are in the same group
= 2 otherwise

The Editex distance between two words is the summation of Editex distance between their letters and length-normalized edit distance is:

$$Ed(w_e, w_f) = \log(1 - \frac{d(w_e, w_f)}{\max(|w_e|, |w_f|)})$$

where $d(w_e, w_f)$ is the "Editex" style edit distance and $\max(|w_e|, |w_f|)$ is the maximum of the two lengths for the source and target, normalizing the edit distance.

The Editex edit distance is deployed between English words and "romanized" Arabic words with a grouping of similar consonants and a grouping of similar vowels. This helps in identifying the correspondence between rare NEs during the alignment. For example, consider two rare NE phrases that occur once in the training:

"وقد استدعى وزير الخارجية الياباني نوبوتاكا ماشيمورا السفير الصيني وانج يي"

"*wqd AstdEY wzyr AlxArjyp AlyAbAny nwbwtAkA mA$ymwrA Alsfyr AlSyny wAnj yy*"

"*Japanese Foreign Minister Nobutaka Machimura has summoned the Chinese ambassador Wang Yee*"

Thus the task of the alignment technique is to align
نوبوتاكا :nwbwkAtA → Nobutaka
ماشيمورا : mA$ymwrA → Machimura
وانج :wAng → Wang
يي :yy → Yee

If a pure Model-1 alignment was used, then the model would have concluded that all words could be aligned to all others with equal probability. However, the multi-cost alignment technique could align two named entities using a single training sample. This approach has significant effect in correctly aligning rare NEs.

The term $Tag(w_e, w_f)$ in the alignment cost function is the NE type cost which increases the alignment cost when the source and target words are annotated with different types and is zero otherwise.

The parameters of the cost function ($\lambda_1, \lambda_2, \lambda_3$) can be tuned according to the NE category and to frequency of a NE. For example, in the case of person's names, it might be advantageous to use a larger λ_2 (boosting the weight of transliteration).

- **Multi-cost Named Entity Alignment by Content Words Elimination**

In the case of organization and location names; many content words, which are words other than the NEs, occur in the NE phrases. These content words might be aligned incorrectly to rare NE words. A two-phase alignment approach is deployed to overcome this problem. The first phase is aligning the content words using a content-word-only translation matrix. The successfully aligned content words are removed from both the source and target sentences. In the second phase, the remaining words are subsequently aligned using the multi-cost alignment technique described in the previous section. This two-phase approach filters out the words that might be incorrectly aligned using the single phase alignment techniques. Thus the alignment accuracy is enhanced; especially for organization names since organization names used to contain many content words.

The following example illustrates the technique, consider two sentences to be aligned and to avoid language confusion let's assume symbolic sentences by denoting:

- Wsi: content words in the source sentence.
- NEsi: the Named Entity source words.
- Wti: the content words in the target sentence.
- NEti: the Named Entity target words.

The source and target sentences are represented as follows:

Source: *Ws1 Ws2 NEs1 NEs2 Ws3 Ws4 Ws5*

Target: *Wt1 Wt2 Wt3 NEt1 NEt2 NEt3 Wt4 NEt4*

After the first phase is applied, the remaining not aligned words might look like that:

Source: *NEs1 NEs2 Ws4 Ws5*

Target: *Wt3 NEt1 NEt2 NEt3 NEt4*

The example clarify that the elimination of some content words facilitates the task of NEs alignment since many of the words that might lead to confusion have been eliminated.

As shown in the above example, different mismatched identification of NEs could result from different identifiers. The "Multi-cost Named Entity Alignment by Content Words Elimination" technique helps in reducing alignment errors due to identification errors by reducing the candidate words for alignment and thus reducing the aligner confusion.

4.2 Phrase Based Named Entity Translation

For phrase-based NE translation, we used an approach similar to that presented by Tillman (Tillmann, 2003) for block generation with modifications suitable for NE phrase extraction. A block is defined to be any pair of source and target phrases. This approach starts from a word alignment generated by HMM Viterbi training (Vogel et. Al, 1996), which is done in both directions between source and target. The intersection of the two alignments is considered a high precision alignment and the union is considered a low precision alignment. The high precision alignments are used to generate high precision blocks which are further expanded using low precision alignments. The reader is referred to (Tillmann, 2003) for detailed description of the algorithm.

In our approach, for extracting NE blocks, we limited high precision alignments to NE phrases of the same NE types. In the expansion phase, the multi-cost function described earlier is used. Thus the blocks are expanded based on a cost depending on the type matching cost, the edit distance cost and the translation probability cost.

To explain this procedure, consider the following sentences pair:

وقد استدعى وزير الخارجية الياباني نوبوتاكا ماشيمورا السفير الصيني وانج يي"

"wqd AstdEY wzyr AlxArjyp AlyAbAny nwbwtAkA mA$ymwrA Alsfyr AlSyny wAnj yy"

"Japanese Foreign Minister Nobutaka Machimura has summoned the Chinese ambassador Wang Yee

The underlined words are the words that have been identified by the NE identifiers as person names. In the Arabic sentence, the identifier missed the second name of the first Named Entity (*mA$ymwrA)* and did not identify the word as person name by mistake. The high precision block generation technique will generate the following two blocks:

نوبوتاكا(nwbwtAkA): Nobutaka

وانج يي :(wAnj yy) Wang Yee

The expansion technique will try to expand the blocks on all the four possible dimensions (right and left of the blocks in the target and source) of each block. The result of the expansion will be:

نوبوتاكا ماشيمورا *(nwbwtAkA mA$ymwrA) : Nobutaka Machimura*

Therefore, the multi-cost expansion technique enables expansions sensitive to the translation probability and the edit distance and providing a mechanism to overcome NE identifiers errors.

4.3 Named Entity Transliteration

NE transliteration is essential for translating Out Of Vocabulary (OOV) words that are not covered by the word or phrase based models. As mentioned earlier, phonetic and orthographic differences between Arabic and English make NE transliteration challenging.

We used a block based transliteration method, which transliterates sequence of letters from the source language to sequence of letters in the target language. These source and target sequences construct the blocks which enables the modeling of vowels insertion. For example, consider Arabic name "شكري $kry," which is transliterated as "Shoukry." The system tries to model bi-grams from the source language to n-grams in the target language as follows:

$k → shouk

kr→ kr

ry → ry

To obtain these block translation probabilities, we use the translation matrix, generated in section 4.1 from the word based alignment models. First, the translation matrix is filtered out to only preserve highly confident translations; translations with probabilities less than a certain threshold are filtered out. Secondly, the resulting high confident translations are further refined by calculating phonetic based edit distance between both romanized Arabic and English names. Name pairs with an edit distance greater than a predefined threshold are also filtered out. The remaining highly confident name pairs are used to train a letter to letter translation matrix using HMM Viterbi training (Vogel et al., 1996).

Each bi-gram of letters on the source side is aligned to an n-gram of letters sequence on the target side, such that vowels have very low cost to be aligned to NULL. The block probabilities are calculated and refined iteratively for each source and target sequences. Finally, for a source block *s* and a target block *t*, the probability of *s* being translated as *t* is the ratio of their co-occurrence and total source occurrence:

$$P(t \mid s) = N(t,s)/N(s).$$

The resulting block translation probabilities and the letter to letter translation probabilities are combined to construct a Weighted Finite State Transducer (WFST) for translating any source sequence to a target sequence.

Furthermore, the constructed translation WFST is composed with two language models (LM) transducers namely a letter trigram model and a word unigram model. The trigram letter based LM acts to provide high recall results while the word based unigram LM acts for providing high precisin results.

4.4 System Integration and Decoding

The three constructed models in the steps above, namely phrase-based NE translation, word-based translation, and transliteration, are used to generate hypotheses for each source NE phrase. We used a dynamic programming beam search decoder similar to the decoder described by Tillman (Tillmann, 2003).

We employed two language models that were built from NE phrases extracted from monolingual target data for each NE category under consideration. The first language model is a trigram language model on NE phrases. The second language model is a class based language model with a class for unknown NEs. Every NE that do exist in the monolingual data but out of the vocabulary of the phrase and word translation models are considered unknown. This helps in correctly scoring OOV hypothesis produced by the transliteration module.

5 Experimental Setup

We test our system for Arabic to English NE translation for three NE categories, namely names of persons, organizations, and locations. The system has been trained on a news domain parallel corpus containing 2.8 million Arabic words and 3.4 million words. Monolingual English data was annotated with NE types and the extracted named entities were used to train the various language models described earlier.

We manually constructed a test set as follows:

Category	No. of Phrases	No. of Words
Person names	803	1749
Organization names	312	867
Location names	345	614

The BLEU score (Papineni et al., 2002) with a single reference translation was deployed for evaluation. BLEU-3 which uses up to 3-grams is deployed since three words phrase is a reasonable length for various NE types. Table 1 reports the results for person names; the baseline system is a general-purpose machine translation system with relatively good Bleu score.

System	Bleu Score
Base line	0.2942
Word based only	0.3254
Word + Phrase	0.4620
Word + Phrase + Transliteration	0.5432

Table 1: Person Names Results

Table 2 reports the bleu score for Location category with the same three systems presented before with persons:

System	Bleu Score
Base line	0.2445
Word based only	0.3426
Word + Phrase	0.4721
Word + Phrase + Transliteration	0.4983

Table 2: Locations Names Results

Table 3 reports the bleu score for Organization category with the same three systems presented before:

System	Bleu Score
Base line	0.2235
Word based only	0.2541
Word + Phrase	0.3789
Word + Phrase + Transliteration	0.3876

Table 3: Organizations Names Results

Figure 1, illustrates various BLEU scores for various categories. The results indicate that phrase based translation provided enhancement for all NE types, while transliteration proved more effective for person names.

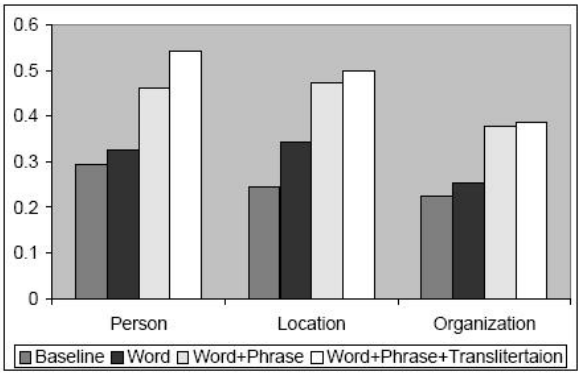

Figure 1: Various BLEU scores for various categories

It is also worth mentioning that evaluating the system using a single reference has limitations; many good translations are considered wrong because they do not exist in the single reference.

6 Conclusion and Future Work

We have presented an integrated system that can handle various NE categories and requires the regular parallel and monolingual corpora which are typically used in the training of any statistical machine translation system along with NEs identifier. The proposed approach does not require any costly special resources, lexicons or any type of annotated data.

The system is composed of multiple translation modules that give flexibility for different named entities type's translation requirements. This gives a great flexibility that enables the system to handle NEs of any type.

We will evaluate the effect of the system on CLIR and MT tasks. We will also try to investigate new approaches for deploying NE translation in general phrase based MT system.

Acknowledgment

We would like to thank Salim Roukos and Kishore Papineni for several invaluable suggestions and guidance. We would like also to thank Christoph Tillmann for help with various components.
We would like also to thank Kareem Darwish for his invaluable help in editing this paper.

References

Yaser Al-Onaizan and Kevin Knight. 2002. Machine Transliteration of Names in Arabic Text. In Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics, 19(2):263–311.

Radu Florian, Hany Hassan, Abraham Ittycheriah, H. Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, Salim Roukos: A Statistical Model for Multilingual Entity Detection and Tracking. HLT-NAACL 2004: 1-8

Fei Huang, Stephan Vogel and Alex Waibel, Automatic Extraction of Named Entity Translingual Equivalence Based on Multi-feature Cost Minimization, in the Proceedings of the 2003 Annual Conference of the Association for Computational Linguistics (ACL'03), Workshop on Multilingual and Mixed-language Named Entity Recognition, July, 2003

Leah Larkey, Nasreen AbdulJaleel, and Margaret Connell, What's in a Name? Proper Names in Arabic Cross-Language Information Retrieval. CIIR Technical Report, IR-278,2003.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of machine translation. In Proc. of the 40th Annual Conf. of the Association for Computational Linguistics (ACL 02), pages 311–318, Philadelphia, PA, July.

Bonnie G. Stalls and Kevin Knight.. Translating Names and Technical Terms in Arabic Text. In Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages. 1998.

Christoph Tillmann,. A Projection Extension Algorithm for Statistical Machine Translation. In Proc of Empirical Methods in Natural Language Processing, 2003

Stefan Vogel, Hermann Ney, and Christoph Tillmann.. HMM Based Word Alignment in Statistical Machine Translation. In Proc. of the 16th Int. Conf. on Computational Linguistics (COLING), 1996

J. Zobel and P. Dart, Phonetic String Matching: Lessons from Information Retrieval. SIGIR Forum, special issue:166--172, 1996

Author Index