A New Approach to Lexical Disambiguation of Arabic Text

Rushin Shah
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
rnshah@cs.cmu.edu

Paramveer S. Dhillon, Mark Liberman, Dean Foster, Mohamed Maamouri and Lyle Ungar
University of Pennsylvania
3451 Walnut Street
Philadelphia, PA 19104, USA
{dhillon|myl|ungar}@cis.upenn.edu,
foster@wharton.upenn.edu,
maamouri@ldc.upenn.edu

Abstract

We describe a model for the lexical analysis of Arabic text, using the lists of alternatives supplied by a broad-coverage morphological analyzer, SAMA, which include stable lemma IDs that correspond to combinations of broad word sense categories and POS tags. We break down each of the hundreds of thousands of possible lexical labels into its constituent elements, including lemma ID and part-of-speech. Features are computed for each lexical token based on its local and document-level context and used in a novel, simple, and highly efficient two-stage supervised machine learning algorithm that overcomes the extreme sparsity of label distribution in the training data. The resulting system achieves accuracy of 90.6% for its first choice, and 96.2% for its top two choices, in selecting among the alternatives provided by the SAMA lexical analyzer. We have successfully used this system in applications such as an online reading helper for intermediate learners of the Arabic language, and a tool for improving the productivity of Arabic Treebank annotators.

1 Background and Motivation

This paper presents a methodology for generating high quality lexical analysis of highly inflected languages, and demonstrates excellent performance applying our approach to Arabic. Lexical analysis of the written form of a language involves resolving, explicitly or implicitly, several different kinds of ambiguities. Unfortunately, the usual ways of talking about this process are also ambiguous, and our general approach to the problem, though not unprecedented, has uncommon aspects. Therefore, in order to avoid confusion, we begin by describing how we define the problem.

In an inflected language with an alphabetic writing system, a central issue is how to interpret strings of characters as forms of words. For example, the English letter-string 'winds' will normally be interpreted in one of four different ways, all four of which involve the sequence of two formatives ***wind+s***. The stem 'wind' might be analyzed as (1) a noun meaning something like "air in motion", pronounced [w**I**nd] , which we can associate with an arbitrary but stable identifier like ***wind_n1***; (2) a verb ***wind_v1*** derived from that noun, and pronounced the same way; (3) a verb ***wind_v2*** meaning something like "(cause to) twist", pronounced [wa**I**nd]; or (4) a noun ***wind_n2*** derived from that verb, and pronounced the same way. Each of these "lemmas", or dictionary entries, will have several distinguishable senses, which we may also wish to associate with stable identifiers. The affix '-s' might be analyzed as the plural inflection, if the stem is a noun; or as the third-person singular inflection, if the stem is a verb.

We see this analysis as conceptually divided into four parts: 1) *Morphological analysis*, which recognizes that the letter-string 'winds' might be (perhaps among other things) ***wind/N*** + ***s/PLURAL*** or ***wind/V*** + ***s/3SING***; 2) *Morphological disambiguation*, which involves deciding, for example, that in the phrase "the four winds", 'winds' is probably a plural noun, i.e. ***wind/N*** + ***s/PLURAL***; 3) *Lemma analysis*, which involves recognizing that the stem wind in 'winds' might be any of the four lemmas listed above – perhaps with a further listing of senses or other sub-entries for each of them; and 4) *Lemma disambiguation*, deciding, for example, that the phrase "the four winds" probably involves the lemma ***wind_n1***.

Confusingly, the standard word-analysis tasks in computational linguistics involve various combinations of pieces of these logically-distinguished operations. Thus, "part of speech (POS) tagging" is mainly what we've called "morphological disambiguation", except that it doesn't necessarily require identifying the specific stems and affixes involved. In some cases, it also may require a small amount of "lemma disambiguation", for example to distinguish a proper noun from a common noun. "Sense disambiguation" is basically a form of what we've called "lemma disambiguation", except that the sense disambiguation task may assume that the part of speech is known, and may break down lexical identity more finely than our system happens to do. "Lemmatization" generally refers to a radically simplified form of "lemma analysis" and "lemma disambiguation", where the goal is simply to collapse different inflected forms of any similarly-spelled stems, so that the strings 'wind', 'winds', 'winded', 'winding' will all be treated as instances of the same thing, without in fact making any attempt to determine the identity of "lemmas" in the traditional sense of dictionary entries.

Linguists use the term *morphology* to include all aspects of lexical analysis under discussion here. But in most computational applications, "morphological analysis" does not include the disambiguation of lemmas, because most morphological analyzers do not reference a set of stable lemma IDs. So for the purposes of this paper, we will continue to discuss lemma analysis and disambiguation as conceptually distinct from morphological analysis and disambiguation, although, in fact, our system disambiguates both of these aspects of lexical analysis at the same time.

The lexical analysis of textual character-strings is a more complex and consequential problem in Arabic than it is in English, for several reasons. First, Arabic inflectional morphology is more complex than English inflectional morphology is. Where an English verb has five basic forms, for example, an Arabic verb in principle may have dozens. Second, the Arabic orthographic system writes elements such as prepositions, articles, and possessive pronouns without setting them off by spaces, roughly as if the English phrase "in a way" were written "inaway". This leads to an enormous increase in the number of distinct "orthographic words", and a substantial increase in ambiguity. Third, short vowels are normally omitted in Arabic text, roughly as if English "in a way" were written "nway".

As a result, a whitespace/punctuation-delimited letter-string in Arabic text typically has many more alternative analyses than a comparable English letter-string does, and these analyses have many more parts, drawn from a much larger vocabulary of form-classes. While an English "tagger" can specify the morphosyntactic status of a word by choosing from a few dozen tags, an equivalent level of detail in Arabic would require thousands of alternatives. Similarly, the number of lemmas that might play a role in a given letter-sequence is generally much larger in Arabic than in English.

We start our labeling of Arabic text with the alternative analyses provided by SAMA v. 3.1, the Standard Arabic Morphological Analyzer (Maamouri et al., 2009). SAMA is an updated version of the earlier Buckwalter analyzers (Buckwalter, 2004), with a number of significant differences in analysis to make it compatible with the LDC Arabic Treebank 3-v3.2 (Maamouri et al., 2004). The input to SAMA is an Arabic orthographic word (a string of letters delimited by whitespace or punctuation), and the output of SAMA is a set of alternative analyses, as shown in Table 1. For a typical word, SAMA produces approximately a dozen alternative analyses, but for certain highly ambiguous words it can produce hundreds of alternatives.

The SAMA analyzer has good coverage; for typical texts, the correct analysis of an orthographic word can be found somewhere in SAMA's list of alternatives about **95%** of the time. However, this broad coverage comes at a cost; the list of analytic alternatives must include a long Zipfian tail of rare or contextually-implausible analyses, which collectively are correct often enough to make a large contribution to the coverage statistics. Furthermore, SAMA's long lists of alternative analyses are not evaluated or ordered in terms of overall or contextual plausibility. This makes the results less useful in most practical applications.

Our goal is to rank these alternative analyses so that the correct answer is as near to the top of the list

Token	**Lemma**	**Vocalization**	**Segmentation**	**Morphology**	**Gloss**
yHlm	Halam-u_1	yaHolumu	ya + Holum + u	IV3MS + IV + IV-SUFF_MOOD:I	he / it + dream + [ind.]
yHlm	Halam-u_1	yaHoluma	ya + Holum + a	IV3MS + IV + IV-SUFF_MOOD:S	he / it + dream + [sub.]
yHlm	Halum-u_1	yaHolumo	ya + Holum + o	IV3MS + IV + IV-SUFF_MOOD:J	he / it + be gentle + [jus.]
qbl	qabil-a_1	qabila	qabil + a	PV + PV-SUFF_SUBJ:3MS	accept/receive/approve + he/it [verb]
qbl	qabol_1	qabol	qabol	NOUN	Before

Table 1: Partial output of SAMA for ***yHlm*** and ***qbl***. On average, every token produces more than 10 such analyses

as possible. Despite some risk of confusion, we'll refer to SAMA's list of alternative analyses for an orthographic word as potential *labels* for that word. And despite a greater risk of confusion, we'll refer to the assignment of probabilities to the set of SAMA labels for a particular Arabic word in a particular textual context as *tagging*, by analogy to the operation of a stochastic part-of-speech tagger, which similarly assigns probabilities to the set of labels available for a word in textual context.

Although our algorithms have been developed for the particular case of Arabic and the particular set of lexical-analysis labels produced by SAMA, they should be applicable without modification to the sets of labels produced by any broad-coverage lexical analyzer for the orthographic words of any highly-inflected language.

In choosing our approach, we have been motivated by two specific applications. One application aims to help learners of Arabic in reading text, by offering a choice of English glosses with associated Arabic morphological analyses and vocalizations. SAMA's excellent coverage is an important basis for this help; but SAMA's long, unranked list of alternative analyses for a particular letter-string, where many analyses may involve rare words or alternatives that are completely implausible in the context, will be confusing at best for a learner. It is much more helpful for the list to be ranked so that the correct answer is almost always near the top, and is usually one of the top two or three alternatives.

In our second application, this same sort of ranking is also helpful for the linguistically expert native speakers who do Arabic Treebank analysis. These annotators understand the text without difficulty, but find it time-consuming and fatiguing to scan a long list of rare or contextually-implausible alternatives for the correct SAMA output. Their work is faster and more accurate if they start with a list that is ranked accurately in order of contextual plausibility.

Other applications are also possible, such as vocalization of Arabic text for text-to-speech synthesis, or lexical analysis for Arabic parsing. However, our initial goals have been to rank the list of SAMA outputs for human users.

We note in passing that the existence of set of stable "lemma IDs" is an unusual feature of SAMA, which in our opinion ought to be emulated by approaches to lexical analysis in other languages. The lack of such stable lemma IDs has helped to disguise the fact that without lemma analysis and disambiguation, morphological analyses and disambiguation is only a partial solution to the problem of lexical analysis.

In principle, it is obvious that lemma disambiguation and morphological disambiguation are mutually beneficial. If we know the answer to one of the questions, the other one is easier to answer. However, these two tasks require rather different sets of contextual features. Lemma disambiguation is similar to the problem of word-sense disambiguation – on some definitions, they are identical – and as a result, it benefits from paragraph-level and document-level bag-of-words attributes that help to characterize what the text is "about" and therefore which lemmas are more likely to play a role in it. In contrast, morphological disambiguation mainly depends on features of nearby words, which help to characterize how inflected forms of these lemmas might fit into local phrasal structures.

2 Problem and Methodology

Consider a collection of tokens (observations), t_i, referred to by index $i \in \{1, \ldots, n\}$, where each token is associated with a set of p features, x_{ij}, for the j^{th} feature, and a label, l_i, which is a combination of a lemma and a morphological analysis. We use indicator functions y_{ik} to indicate whether or not the k^{th} label for the i^{th} token is present. We represent the complete set of features and labels for the entire training data using matrix notation as X and Y, respectively. Our goal is to predict the label l (or equivalently, the vector y for a given feature vector x.

A standard linear regression model of this problem would be

$$\mathbf{y} = \mathbf{x}\beta + \epsilon \tag{1}$$

The standard linear regression estimate of β (ignoring, for simplicity the fact that the ys are 0/1) is:

$$\hat{\beta} = (X_{train}^T X_{train})^{-1} X_{train}^T Y_{train} \tag{2}$$

where Y_{train} is an $n \times h$ matrix containing 0s and 1s indicating whether or not each of the h possible labels is the correct label (l_i) for each of the n tokens t_i, X_{train} is an $n \times p$ matrix of context features for each of the n tokens, the coefficients $\hat{\beta}$ are $p \times h$.

However, this is a large, sparse, multiple label problem, and the above formulation is neither statistically nor computationally efficient. Each observation $(\mathbf{x}, \mathbf{y})$ consists of thousands of features associated with thousands of potential labels, almost all of which are zero. Worse, the matrix of coefficients β, to be estimated is large ($p \times h$) and one should thus use some sort of transfer learning to share strength across the different labels.

We present a novel principled and highly computationally efficient method of estimating this multi-label model. We use a two stage procedure, first using a subset (X_{train1}, Y_{train1}) of training data to give a fast approximate estimate of β; we then use a second smaller subset of the training data $(X_{train2}, Y_{train2},)$ to "correct" these estimates in a way that we will show can be viewed as a specialized shrinkage. Our first stage estimation approximates β, but avoids the expensive computation of $(X_{train}^T X_{train})^{-1}$. Our second stage corrects (shrinks) these initial estimates in a manner specialized to this problem. The second stage takes advantage of the fact that we only need to consider those candidate labels produced by SAMA. Thus, only dozens of the thousands of possible labels are considered for each token.

We now present our algorithm. We start with a corpus D of documents d of labeled Arabic text. As described above, each token, t_i is associated with a set of features characterizing its context, computed from the other words in the same document, and a label, $l_i = (\text{lemma}_i, \text{morphology}_i)$, which is a combination of a lemma and a morphological analysis. As described below, we introduce a novel factorization of the morphology into 15 different components.

Our estimation algorithm, shown in Algorithm 1, has two stages. We partition the training corpus into two subsets, one of which (X_{train1}) is used to estimate the coefficients βs and the other of which (X_{train2}) is used to optimally "shrink" these coefficient estimates to reduce variance and prevent overfitting due to data sparsity.

For the first stage of our estimation procedure, we simplify the estimate of the (β) matrix (Equation 2) to avoid the inversion of the very high dimensional ($p \times p$) matrix ($X^T X$) by approximating ($X^T X$) by its diagonal, $\text{Var}(X)$, the inverse of which is trivial to compute; i.e. we estimate β using

$$\hat{\beta} = \text{Var}(X_{train1})^{-1} X_{train1}^T Y_{train1} \tag{3}$$

For the second stage, we assume that the coefficients for each feature can be shrunk differently, but that coefficients for each feature should be shrunk the same regardless of what label they are predicting. Thus, for a given observation we predict:

$$\hat{g}_{ik} = \sum_{j=1}^{p} w_j \hat{\beta}_{jk} x_{ij} \tag{4}$$

where the weights w_j indicate how much to shrink each of the p features.

In practice, we fold the variance of each of the j features into the weight, giving a slightly modified equation:

$$\hat{g}_{ik} = \sum_{j=1}^{p} \alpha_j \beta_{jk}^* x_{ij} \tag{5}$$

where $\beta^* = X_{train1}^T Y_{train1}$ is just a matrix of the counts of how often each context feature shows up with each label in the first training set. The vector α, which we will estimate by regression, is just the shrinkage weights w rescaled by the feature variance.

Note that the formation here is different from the first stage. Instead of having each observation be a token, we now let each observation be a (token, label) pair, but only include those labels that were output by SAMA. For a given token t_i and potential label l_k, our goal is to approximate the indicator function $g(i,k)$, which is 1 if the k^{th} label of token t_i is present, and 0 otherwise. We find candidate labels using a morphological analyzer (namely SAMA), which returns a set of possible candidate labels, say $C(t)$, for each Arabic token t. Our predicted label for t_i is then $\text{argmax}_{k \in C(t_i)} g(i,k)$.

The regression model for learning the weights α_j in the second stage thus has a row for each label $g(i,k)$ associated with a SAMA candidate for each token $i = n_{train1+1} \ldots n_{train2}$ in the second training set. The value of $g(i,k)$ is predicted as a function of the feature vector $z_{ijk} = \beta^*_{jk} x_{ij}$.

The shrinkage coefficients, α_j, could be estimated from theory, using a version of James-Stein shrinkage (James and Stein, 1961), but in practice, superior results are obtained by estimating them empirically. Since there are only p of them (unlike the $p * h$ βs), a relatively small training set is sufficient. We found that regression-SVMs work slightly better than linear regression and significantly better than standard classification SVMs for this problem.

Prediction is then done in the obvious way by taking the tokens in a test corpus D_{test}, generating context features and candidate SAMA labels for each token t_i, and selected the candidate label with the highest score $\hat{g}(i,k)$ that we set out to learn. More formally, The model parameters β^* and α produced by the algorithm allow one to estimate the most likely label for a new token t_i out of a set of candidate labels $C(t_i)$ using

$$k_{pred} = \text{argmax}_{k \in C(t_i)} \sum_{j=1}^{p} \alpha_j \beta^*_{jk} x_{ij} \qquad (6)$$

The most expensive part of the procedure is estimating β^*, which requires for each token in corpus D_1, (a subset of D), finding the co-occurrence frequencies of each label element (a lemma, or a part of the morphological segmentation) with the target token and jointly with the token and with other tokens or characters in the context of the token of interest. For example, given an Arabic token, ***"yHlm"***, we count what fraction of the time it is associated with each lemma (e.g. ***Halam-u_1***), *count(lemma=**Halam-u_1**, token=**yHlm**)* and each segment (e.g. "ya"), *count(segment=**ya**, token=**yHlm**)*. (Of course, most tokens never show up with most lemmas or segments; this is not a problem.) We also find the base rates of the components of the labels (e.g., *count(lemma=**Halam-u_1**)*, and what fraction of the time the label shows up in various contexts, e.g. *count(lemma=**Halam-u_1**, previous token = **yHlm**)*. We describe these features in more detail below.

Algorithm 1 Training algorithm.

```
Input: A training corpus $D_{train}$ of $n$ observations $(X_{train}, Y_{train})$
Partition $D_{train}$ into two sets, $D_1$ and $D_2$, of sizes $n_{train1}$ and $n_{train2} = n - n_{train1}$ observations
// Using $D_1$, estimate $\beta^*$
$\beta^*_{jk} = \sum_{i=1}^{n_{train1}} x_{ij} y_{ik}$ for the $j^{th}$ feature and $k^{th}$ label
// Using $D_2$, estimate $\alpha_j$
// Generate new "features" $Z$ and the true labels $g(i,k)$ for each of the SAMA candidate labels for each of the tokens in $D_2$
$z_{ijk} = \beta^*_{jk} x_{ij}$ for $i$ in $i = n_{train1} + 1 \ldots n_{train2}$
Estimate $\alpha_j$ for the above (feature,label) pairs $(z_{ijk}, g(i,k))$ using Regression SVMs
Output: $\alpha$ and $\beta^*$
```

3 Features and Labels used for Training

Our approach to tagging Arabic differs from conventional approaches in the two-part shrinkage-based method used, and in the choice of both features and labels used in our model. For features, we study both local context variables, as described above, and document-level word frequencies. For the labels, the key question is what labels are included and how they are factored. Standard "taggers" work by doing an n-way classification of all the alternatives, which is not feasible here due to the thousands of possible labels. Standard approaches such as Conditional Random Fields (CRFs) are intractable with so many labels. Moreover, few if any taggers do any lemma disambiguation; that is partly because one must start with some standard inventory of lemmas, which are not available for most languages, perhaps because the importance of lemma disambiguation has been underestimated.

We make a couple of innovations to deal with these issues. First, we perform lemma disambiguation in addition to "tagging". As mentioned above, lemmas and morphological information are not independent; the choice of lemma often influences morphology and vice versa. For example, Table 1 contains two analyses for the word **qbl**. For the first analysis, where the lemma is ***qabil-a_1*** and the gloss is *accept/receive/approve + he/it [verb]*, the word is a verb. However, for the second analysis, where the lemma is ***qabol_1*** and the gloss is *before*, the word is a noun.

Simultaneous lemma disambiguation and tagging introduces additional complexity: An analysis of ATB and SAMA shows that there are approximately 2,200 possible morphological analyses ("tags") and 40,000 possible lemmas; even accounting for the fact that most combinations of lemmas and morphological analyses don't occur, the size of the label space is still in the order of tens of thousands. To deal with data sparsity, our second innovation is to factor the labels. We factor each label l into a set of 16 label elements (**LE**s). These include lemmas, as well as morphological elements such as basic part-of-speech, suffix, gender, number, mood, etc. These are explained in detail below. Thus, since each label l is a set of 15 categorical variables, each $\mathbf{y}$ in the first learning stage is actually a vector with 16 nonzero components and thousands of zeros. Since we do simultaneous estimation of the entire set of label elements, the value $g(i, k)$ being predicted in the second learning phase is 1 if the entire label set is correct, and zero otherwise. We do *not* learn separate models for each label.

3.1 Label Elements (LEs)

The fact that there are tens of thousands of possible labels presents the problem of extreme sparsity of label distribution in the training data. We find that a model that estimates coefficients β^* to predict a single label (a label being in the Cartesian product of the set of label elements) yields poor performance. Therefore, as just mentioned, we factor each label l into a set of label elements (**LE**s), and learn the correlations β^* between features and label elements, rather than features and entire label sets. This reduces, but does not come close to eliminating, the problem sparsity. A complete list of these **LE**s and their possible values is detailed in Table 2.

LE	**Description**
lemma	Lemma
pre1	Closer prefix
pre2	Farther prefix
det	Determiner
pos	Basic POS
dpos	Additional data on basic pos
suf	Suffix
perpos	Person (basic pos)
numpos	Number (basic pos)
genpos	Gender (basic pos)
persuf	Person (suffix)
numsuf	Number (suffix)
gensuf	Gender (suffix)
mood	Mood of verb
pron	Pronoun suffix

Table 2: Label Elements (**LE**s). Examples of additional data on basic POS include whether a noun is proper or common, whether a verb is transitive or not, etc. Both the basic POS and its suffix may have person, gender and number data.

3.2 Features

3.2.1 Local Context Features

We take (t, l) pairs from D_2, and for each such pair generate features Z based on co-occurrence statistics β^* in D_1, as mentioned in Algorithm 2. These statistics include unigram co-occurrence frequencies of each label with the target token and bigram co-occurrence of the label with the token and with other tokens or characters in the context of the target token. We define them formally in Table 3. Let $Z_{baseline}$ denote the set of all such basic features based on the local context statistics of the target token, namely the words and letters preceding and following it. We will use this set to create a baseline model.

Statistic	Description
Freq	$count_{D_1}(t, l)$
PrevWord	$count_{D_1}(t, l, t_{-1})$
NextWord	$count_{D_1}(t, l, t_{+1})$
PreviLetter	$count_{D_1}(t, l, \text{first letter}(t_{-1}))$
NextiLetter	$count_{D_1}(t, l, \text{first letter}(t_{+1}))$
PrevfLetter	$count_{D_1}(t, l, \text{last letter}(t_{-1}))$
NextfLetter	$count_{D_1}(t, l, \text{last letter}(t_{+1}))$

Table 3: Co-occurrence statistics β^*. We use these to generate feature sets for our regression SVMs.

For each label element (**LE**) e, we define a set of features Z_e similar to $Z_{baseline}$; these features are based on co-occurrence frequencies of the particular **LE** e, not the entire label l.

Finally, we define an aggregate feature set Z_{aggr} as follows:

$$Z_{aggr} = Z_{baseline} \quad \bigcup \quad \{Z_e\} \qquad (7)$$

where $e \in \{$*lemma, pre1, pre2, det, pos, dpos, suf, perpos, numpos, genpos, persuf, numsuf, gensuf, mood, pron*$\}$.

3.2.2 Document Level Features

When trying to predict the lemma, it is useful to include not just the words and characters immediately adjacent to the target token, but also the all the words in the document. These words capture the "topic" of the document, and help to disambiguate different lemmas, which tend to be used or not used based on the topic being discussed, similarly to the way that word sense disambiguation systems in English sometimes use the "bag of words" the document to disambiguate, for example a "bank" for depositing money from a "bank" of a river. More precisely, we augment the features for each target token with the counts of each word in the document (the "term frequency" *tf*) in which the token occurs with a given label.

$$Z_{full} = Z_{aggr} \quad \bigcup \quad Z_{tf} \qquad (8)$$

This set Z_{full} is our final feature set. We use Z_{full} to train an SVM model M_{full}; this is our final predictive model.

3.3 Corpora used for Training and Testing

We use three modules of the Penn Arabic Treebank (ATB) (Maamouri et al., 2004), namely ATB1, ATB2 and ATB3 as our corpus of labeled Arabic text, D. Each ATB module is a collection of newswire data from a particular agency. ATB1 uses the Associated Press as a source, ATB2 uses Ummah, and ATB3 uses Annahar. D contains a total of 1,835 documents, accounting for approximately 350,000 words. We construct the training and testing sets D_{train} and D_{test} from D using 10-fold cross validation, and we construct D_1 and D_2 from D_{train} by randomly performing a 9:1 split.

As mentioned earlier, we use the SAMA morphological analyzer to obtain candidate labels $C(t)$ for each token t while training and testing an SVM model on D_2 and D_{test} respectively. A sample output of SAMA is shown in Table 1. To improve coverage, we also add to $C(t)$ all the labels l seen for t in D_1. We find that doing so improves coverage to **98%**. This is an upper bound on the accuracy of our model.

$$C(t) = \text{SAMA}(t) \quad \bigcup \quad \{l|(t, l) \in D_1\} \qquad (9)$$

4 Results

We use two metrics of accuracy: **A1**, which measures the percentage of tokens for which the model assigns the highest score to the correct label or **LE** value (or **E1**= $100 - A1$, the corresponding percentage error), and **A2**, which measures the percentage of tokens for which the correct label or **LE** value is one of the two highest ranked choices returned by the model (or **E2** = $100 - A2$). We test our model M_{full} on D_{test} and achieve **A1** and **A2** scores of 90.6% and 96.2% respectively. The accuracy achieved by our M_{full} model is, to the best of our knowledge, higher than prior approaches have been able to achieve so far for the problem of combined morphological and lemma disambiguation. This is all the more impressive considering that the upper bound on accuracy for our model is **98%** because, as described above, our set of candidate labels is incomplete.

In order to analyze how well different **LE**s can be predicted, we train an SVM model M_e for each **LE** e using the feature set Z_e, and test all such models on D_{test}. The results for all the **LE**s are reported in the form of error percentages **E1** and **E2** in Table 4.

Model	E1	E2	Model	E1	E2
M_{lemma}	11.1	4.9	M_{pre1}	1.9	1.4
M_{pre2}	0.2	0	M_{det}	0.7	0.1
M_{pos}	23.4	4.0	M_{dpos}	10.3	1.9
M_{suf}	7.6	2.5	M_{perpos}	3.0	0.1
M_{numpos}	3.2	0.2	M_{genpos}	1.8	0.1
M_{persuf}	3.2	0.1	M_{numsuf}	8.2	0.5
M_{gensuf}	11.6	0.4	M_{mood}	1.6	1.4
M_{pron}	1.8	0.6	M_{case}	14.7	5.9
$\boldsymbol{M_{full}}$	**9.4**	**3.8**	-	-	-

Table 4: Results of M_e for each **LE** e. **Note:** The results reported are 10 fold cross validation test accuracies and no parameters have been tuned on them.

A comparison of the results for M_{full} with the results for M_{lemma} and M_{pos} is particularly informative. We see that M_{full} is able to achieve a substantially lower **E1** error score (9.4%) than M_{lemma} (11.1%) and M_{pos} (23.4%); in other words, we find that our full model is able to predict lemmas and basic parts-of-speech more accurately than the individual models for each of these elements.

We examine the effect of varying the size of D_2, i.e. the number of SVM training instances, on the performance of M_{full} on D_{test}, and find that with increasing sizes of D_2, **E1** reduces only slightly from 9.5% to 9.4%, and shows no improvement thereafter. We also find that the use of document-level features in M_{lemma} reduces **E1** and **E2** percentages for M_{lemma} by 5.7% and 3.2% respectively.

4.1 Comparison to Alternate Approaches

4.1.1 Structured Prediction Models

Preliminary experiments showed that knowing the predicted labels (lemma + morphology) of the surrounding words can slightly improve the predictive accuracy of our model. To further investigate this effect, we tried running experiments using different structured models, namely CRF (Conditional Random Fields) (Lafferty et al., 2001), (Structured) MIRA (Margin Infused Relaxation Algorithm) (Crammer et al., 2006) and Structured Perceptron (Collins, 2002). We used linear chain CRFs as implemented in MALLET Toolbox (McCallum, 2001) and for Structured MIRA and Perceptron we used their implementations from EDLIN Toolbox (Ganchev and Georgiev, 2009). However, given the vast label space of our problem, running these methods proved infeasible. The time complexity of these methods scales badly with the number of labels; It took a week to train a linear chain CRF for only $\sim$ 50 labels and though MIRA and Perceptron are online algorithms, they also become intractable beyond a few hundred labels. Since our label space contains combinations of lemmas and morphologies, so even after factoring, the dimension of the label space is in the order of thousands.

We also tried a naïve version (two-pass approximation) of these structured models. In addition to the features in Z_{full}, we include the predicted labels for the tokens preceding and following the target token as features. This new model is not only slow to train, but also achieves only slightly lower error rates (1.2% lower **E1** and 1.0% lower **E2**) than M_{full}. This provides an upper bound on the benefit of using the more complex structured models, and suggests that given their computational demands our (unstructured) model M_{full} is a better choice.

4.1.2 MADA

(Habash and Rambow, 2005) perform morphological disambiguation using a morphological analyzer. (Roth et al., 2008) augment this with lemma disambiguation; they call their system MADA. Our work differs from theirs in a number of respects. Firstly, they don't use the two step regression procedure that we use. Secondly, they use only "unigram" features. Also, they do not learn a single model from a feature set based on labels and **LE**s; instead, they combine models for individual elements by using weighted agreement. We trained and tested MADA v2.32 using its full feature set on the same D_{train} and D_{test}. We should point out that this is not an exact comparison, since MADA uses the older Buckwalter morphological analyzer.[1]

4.1.3 Other Alternatives

Unfactored Labels: To illustrate the benefit obtained by breaking down each label l into

[1]A new version of MADA was released very close to the submission deadline for this conference.

LEs, we contrast the performance of our M_{full} model to an SVM model $M_{baseline}$ trained using only the feature set $Z_{baseline}$, which only contains features based on entire labels, those based on individual **LE**s.

Independent lemma and morphology prediction: Another alternative approach is to predict lemmas and morphological analyses separately. We construct a feature set $Z_{lemma'} = Z_{full} - Z_{lemma}$ and train an SVM model $M_{lemma'}$ using this feature set. Labels are then predicted by simply combining the results predicted independently by M_{lemma} and $M_{lemma'}$. Let M_{ind} denote this approach.

Unigram Features: Finally, we also consider a context-less approach, i.e. using only "unigram" features for labels as well as **LE**s. We call this feature set Z_{uni}, and the corresponding SVM model M_{uni}.

The results of these various models, along with those of M_{full} are summarized in Table 5. We see that M_{full} has roughly half the error rate of the state-of-the-art MADA system.

Model	**E1**	**E2**
$M_{baseline}$	13.6	9.1
M_{ind}	18.7	6.0
M_{uni}	11.6	6.4
M_{cheat}	8.2	2.8
MADA	**16.9**	**12.6**
$\boldsymbol{M_{full}}$	**9.4**	**3.8**

Table 5: Percent error rates of alternative approaches. **Note:** The results reported are 10 fold cross validation test accuracies and no parameters have been tuned on them. We used same train-test splits for all the datasets.

5 Related Work

(Hajic, 2000) show that for highly inflectional languages, the use of a morphological analyzer improves accuracy of disambiguation. (Diab et al., 2004) perform tokenization, POS tagging and base phrase chunking using an SVM based learner. (Ahmed and Nürnberger, 2008) perform word-sense disambiguation using a Naive Bayesian model and rely on parallel corpora and matching schemes instead of a morphological analyzer. (Kulick, 2010) perform simultaneous tokenization and part-of-speech tagging for Arabic by separating closed and open-class items and focusing on the likelihood of possible stems of open-class words. (Mohamed and Kübler, 2010) present a hybrid method between word-based and segment-based POS tagging for Arabic and report good results. (Toutanova and Cherry, 2009) perform joint lemmatization and part-of-speech tagging for English, Bulgarian, Czech and Slovene, but they do not use the two step estimation-shrinkage model described in this paper; nor do they factor labels. The idea of joint lemmatization and part-of-speech tagging has also been discussed in the context of Hungarian in (Kornai, 1994).

A substantial amount of relevant work has been done previously for Hebrew. (Adler and Elhadad, 2006) perform Hebrew morphological disambiguation using an unsupervised morpheme-based HMM, but they report lower scores than those achieved by our model. Moreover, their analysis doesn't include lemma IDs, which is a novelty of our model. (Goldberg et al., 2008) extend the work of (Adler and Elhadad, 2006) by using an EM algorithm, and achieve an accuracy of 88% for full morphological analysis, but again, this does not include lemma IDs. To the best of our knowledge, there is no existing research for Hebrew that does what we did for Arabic, namely to use simultaneous lemma and morphological disambiguation to improve both. (Dinur et al., 2009) show that prepositions and function words can be accurately segmented using unsupervised methods. However, by using this method as a preprocessing step, we would lose the power of a *simultaneous solution* for these problems. Our method is closer in style to a CRF, giving much of the accuracy gains of simultaneous solution, while being about 4 orders of magnitude easier to train.

We believe that our use of factored labels is novel for the problem of simultaneous lemma and morphological disambiguation; however, (Smith et al., 2005) and (Hatori et al., 2008) have previously made use of features based on parts of labels in CRF models for morphological disambiguation and word-sense disambiguation respectively. Also, we note that there is a similarity between our two-stage machine learning approach and log-linear models in machine translation that break the data in two parts, estimating log-probabilities of generative models from one part, and discriminatively re-weighting the models using the second part.

6 Conclusions

We introduced a new approach to accurately predict labels consisting of both lemmas and morphological analyses for Arabic text. We obtained an accuracy of over 90% – substantially higher than current state-of-the-art systems. Key to our success is the factoring of labels into lemma and a large set of morphosyntactic elements, and the use of an algorithm that computes a simple initial estimate of the coefficient relating each contextual feature to each label element (simply by counting co-occurrence) and then regularizes these features by shrinking each of the coefficients for each feature by an amount determined by supervised learning using only the candidate label sets produced by SAMA.

We also showed that using features of word n-grams is preferable to using features of only individual tokens of data. Finally, we showed that a model using a full feature set based on labels as well as factored components of labels, which we call label elements (**LE**s) works better than a model created by combining individual models for each **LE**. We believe that the approach we have used to create our model can be successfully applied not just to Arabic but also to other languages such as Turkish, Hungarian and Finnish that have highly inflectional morphology. The current accuracy of of our model, getting the correct answer among the top two choices 96.2% of the time is high enough to be highly useful for tasks such as aiding the manual annotation of Arabic text; a more complete automation would require that accuracy for the single top choice.

Acknowledgments

We woud like to thank everyone at the Linguistic Data Consortium, especially Christopher Cieri, David Graff, Seth Kulick, Ann Bies, Wajdi Zaghouani and Basma Bouziri for their help. We also wish to thank the anonymous reviewers for their comments and suggestions.

References

Meni Adler and Michael Elhadad. 2006. An Unsupervised Morpheme-Based HMM for Hebrew Morphological Disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*.

Farag Ahmed and Andreas Nürnberger. 2008. Arabic/English Word Translation Disambiguation using Parallel Corpora and Matching Schemes. In *Proceedings of EAMT'08*, Hamburg, Germany.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer version 2.0.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of EMNLP'02*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic Tagging of Arabic text: From Raw Text to Base Phrase Chunks. In *Proceedings of the 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL'04)*.

Elad Dinur, Dmitry Davidov, and Ari Rappoport. 2009. Unsupervised Concept Discovery in Hebrew Using Simple Unsupervised Word Prefix Segmentation for Hebrew and Arabic. In *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*.

Kuzman Ganchev and Georgi Georgiev. 2009. Edlin: An Easy to Read Linear Learning Framework. In *Proceedings of RANLP'09*.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start)*. In *Proceedings of ACL'08*.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of ACL'05*, Ann Arbor, MI, USA.

Jan Hajic. 2000. Morphological Tagging: Data vs. Dictionaries. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*.

Jun Hatori, Yusuke Miyao, and Jun'ichi Tsujii. 2008. Word Sense Disambiguation for All Words using Tree-Structured Conditional Random Fields. In *Proceedings of COLing'08*.

W. James and Charles Stein. 1961. Estimation with Quadratic Loss. In *Proceedings of the Fourth Berkeley*

Symposium on Mathematical Statistics and Probability, Volume 1.

András Kornai. 1994. On Hungarian morphology (Linguistica, Series A: Studia et Dissertationes 14). *Linguistics Institute of Hungarian Academy of Sciences, Budapest*.

Seth Kulick. 2010. Simultaneous Tokenization and Part-of-Speech Tagging for Arabic without a Morphological Analyzer. In *Proceedings of ACL'10*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML'01*, pages 282–289.

Mohamed Maamouri, Ann Bies, and Tim Buckwalter. 2004. The Penn Arabic Treebank: Building a Large Scale Annotated Arabic Corpus. In *Proceedings of NEMLAR Conference on Arabic Language Resources and Tools*.

Mohamed Maamouri, David Graff, Basma Bouziri, Sondos Krouna, and Seth Kulick. 2009. LDC Standard Arabic Morphological Analyzer (SAMA) v. 3.0.

Andrew McCallum, 2001. *MALLET: A Machine Learning for Language Toolkit*. Software available at http://mallet.cs.umass.edu.

Emad Mohamed and Sandra Kübler. 2010. Arabic Part of Speech Tagging. In *Proceedings of LREC'10*.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. In *Proceedings of ACL'08*, Columbus, Ohio, USA.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-Based Morphological Disambiguation with Random Fields*. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Kristina Toutanova and Colin Cherry. 2009. A Global Model for Joint Lemmatization and Part-of-Speech Prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*, pages 486–494.