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 conditioning, 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 morpheme 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:

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

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:

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

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:

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

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

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

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:

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

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 result 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.