Natural Language Engineering **14** (2): 223–251. © 2007 Cambridge University Press
doi:10.1017/S135132490700455X First published online 15 May 2007 Printed in the United Kingdom

Part-of-speech tagging of Modern Hebrew text

ROY BAR-HAIM[1], KHALIL SIMA'AN[2]
and YOAD WINTER[3,4]
[1]*Dept. of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel*
[2]*Institute for Logic, Language and Computation, Universiteit van Amsterdam, Amsterdam, The Netherlands*
[3]*Dept. of Computer Science, Technion, Haifa 32000, Israel*
[4]*Netherlands Institute for Advanced Study, Meijboomlaan 1, 2242 PR Wassenaar, The Netherlands*
e-mail: barhair@cs.biu.ac.il, simaan@science.uva.nl, winter@cs.technion.ac.il

(*Received 6 January 2006; revised and accepted 13 October 2006*)

Abstract

Words in Semitic texts often consist of a concatenation of *word segments*, each corresponding to a part-of-speech (POS) category. Semitic words may be ambiguous with regard to their segmentation as well as to the POS tags assigned to each segment. When designing POS taggers for Semitic languages, a major architectural decision concerns the choice of the atomic input tokens (terminal symbols). If the tokenization is at the word level, the output tags must be complex, and represent both the segmentation of the word and the POS tag assigned to each word segment. If the tokenization is at the segment level, the input itself must encode the different alternative segmentations of the words, while the output consists of standard POS tags. Comparing these two alternatives is not trivial, as the choice between them may have global effects on the grammatical model. Moreover, intermediate levels of tokenization between these two extremes are conceivable, and, as we aim to show, beneficial. To the best of our knowledge, the problem of tokenization for POS tagging of Semitic languages has not been addressed before in full generality. In this paper, we study this problem for the purpose of POS tagging of Modern Hebrew texts. After extensive error analysis of the two simple tokenization models, we propose a novel, linguistically motivated, intermediate tokenization model that gives better performance for Hebrew over the two initial architectures. Our study is based on the well-known hidden Markov models (HMMs). We start out from a manually devised morphological analyzer and a very small annotated corpus, and describe how to adapt an HMM-based POS tagger for both tokenization architectures. We present an effective technique for smoothing the lexical probabilities using an untagged corpus, and a novel transformation for casting the segment-level tagger in terms of a standard, word-level HMM implementation. The results obtained using our model are on par with the best published results on Modern Standard Arabic, despite the much smaller annotated corpus available for Modern Hebrew.

1 Introduction

Part-of-speech (POS) tagging is commonly known as the task of classifying a word in a given input sentence by assigning it a tag from a predefined set of classes that represent syntactic behavior. For languages such as English, word-level POS tagging seems sufficient because words usually correspond to the syntactically relevant POS tag classes. But for various other languages, including Semitic languages such as Modern Hebrew (henceforth Hebrew) and Modern Standard Arabic (henceforth

Arabic), this view is insufficient. In Hebrew and Arabic, the syntactically relevant POS tag classes do not necessarily correspond to words. In the morphology and orthography of Arabic and Hebrew, words are often formed by concatenating smaller parts, which function as free morphosyntactic units, each of which with its own POS tag. One of these word parts is a potentially polymorphemic *stem*, often derived from a root morpheme, a *template/pattern* morpheme, and possibly an inflectional suffix. The other morphemes within Hebrew words are clitics for certain prepositions, conjunctions, definiteness marking, and other parts of speech. We refer to the stem and the clitics within a word as *word segments*, or *segments*.

Since the different segments in a word may belong to a variety of POS tag classes, the classes that are associated with a word can be seen as *compound classes*, each constituting an ordered sequence of POS tags together with the corresponding segments. Consequently, the task of POS tagging for Semitic languages involves word segmentation in addition to POS tag classification. Crucially, a word may be segmented into different candidate sequences of segments, and, in turn, each of the word segments may be ambiguous with respect to the choice of a POS tag. The result is a relatively high degree of ambiguity, aggravated by the absence of vocalization markers (diacritics) in most modern Semitic texts.

Because of the similarity between Hebrew and Arabic at the orthographic, morphological, and syntactic levels, similar POS tagging architectures are expected to be similarly successful for both languages. A major decision in designing unified models of segmentation and POS tagging for Semitic languages concerns the *tokenization level* of the input: the optimal definition of the terminal symbols for the grammatical model. This paper focuses on this problem, which has not been addressed so far in its full generality with respect to Semitic languages.

We adapt the well-known hidden Markov model (HMM) POS tagging architecture (Church 1988; Brants 2000) to suit the specific tokenization problems of Hebrew and other Semitic languages. We employ HMMs because they easily allow joint modeling (Noisy Channel style) of segmentation and POS tagging, and because they have been successfully applied to various tasks and languages (Brants 2000; Nakagawa 2004) with comparable tagsets to the Hebrew tagset that we employ. This setting is general enough for making principal architectural decisions that are also relevant for other platforms of POS tagging besides HMM.

When considering the tokenization level of HMM-based taggers, two simple options present themselves for Semitic languages: tokenization into words and tokenization into word segments. Between these two alternatives, more sophisticated, intermediate levels of tokenization are of course conceivable. When words are considered as the tokens to be tagged, the corresponding classes are compound tokens. This means that a standard HMM implementation can be directly employed, but the Markov model conditionings can be less balanced and much sparser than when the tokens are word segments. On the other hand, when the tokens are segments, their classes are standard syntactic POS tags, but the alternative segmentations must be derived by a separate segmentation process. This implies that the standard HMM-based POS tagger implementation must be extended in a suitable manner. We provide the implementation details of HMM taggers based on the two alternative strategies of tokenization. In particular, for implementing the segment-level tagger using existing tools, we present a novel transformation that casts segment-level tokenization in terms of the standard HMM-based POS tagging architecture.

For dealing with severe sparse data problems that result from the small size of the annotated Hebrew corpus, we employ an effective smoothing technique based on unsupervised training on a large unannotated corpus. Using this smoothing method, we present extensive experiments that show that neither of the two basic tokenization architectures is optimal with regard to both segmentation and POS tag disambiguation. While the segment-level system outperforms the word-level system with regard to tag disambiguation, the latter turns out at least as successful as the former with regard to segmentation. This surprising result hints at the need for an intermediate level that combines statistics from both architectures. Subsequent error analysis of the output of both taggers points at a reoccurring problem, concerning the disambiguation of nominal words with certain prepositional prefixes. Many such words in Hebrew (but not in Arabic) are textually ambiguous for in/definiteness. Covert occurrences of the Hebrew definiteness marker are a major source of inaccuracy, and more seriously so in the POS tagger based on the segment-level tokenization.

To overcome this problem, we present a modification of the basic segment-level POS tagger that treats the definiteness marker as a feature of the noun rather than as a separate word segment. This treatment resonates well with theoretical linguistic considerations (Wintner 2000; Danon 2001), and results in a novel, intermediate-level model, incorporating segment-level input with word-level features into a single model. Extensive experimentation with this intermediate-level tokenization exhibits segmentation and tagging accuracies that are on par with the best reported results for Standard Arabic, despite the much smaller size of the Hebrew annotated corpus.

The paper is structured as follows. Section 2 gives necessary details on Hebrew morphology and POS tagging. Section 3 describes the existing corpora. Section 4 defines 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 5 gives details on the implementation of the architectures we experiment with. Section 6 compares the different levels of tokenization empirically, discusses their limitations, and introduces an improved model that outperforms both of the initial models. Finally, Section 7 discusses related work and summarizes the conclusions of our study for segmentation and POS tagging of Hebrew, in particular, and Semitic languages, in general.

2 Hebrew morphology and POS tagging

The major syntactic categories that are employed for the analysis of Semitic languages are similar to those of many European languages. However, the morphology and orthography of Hebrew, Standard Arabic, and other Semitic languages differ substantially from those of European languages. Semitic languages have rich inflectional systems and a template-based derivational morphology, which are manifested in a large variation of word forms. In addition, Semitic languages often allow grammatical prefixes/proclitics (e.g. conjunctions and prepositions) and suffixes/enclitics (e.g. pronouns) to appear quite productively, which increases the ambiguity of texts in these languages. Moreover, in most modern Semitic texts, vocalization and gemination are not represented, which further increases the ambiguity of word tokens. In this section, we give a brief overview of Hebrew morphology, and describe the POS tagging scheme that is assumed throughout this paper. For more details on the grammar and morphology of Modern Hebrew, see Glinert (1989).

2.1 Morphological analysis of Hebrew

In theoretical, descriptive, and computational work on Semitic languages (e.g. Glinert, 1989; Segal, 2000; Buckwalter, 2002; Watson, 2002), it is commonly assumed that a word may consist of the following elements:

- *Prefixes (or proclitics):* conjunctions, prepositions, complementizers, and the definiteness marker (in a strict well-defined order).
- *Stem:* a potentially polymorphemic part of the word, together with its POS tag and additional information about its template/pattern and root morphemes (see below).
- *Inflectional suffix:* encoding inflectional features on verbs, adjectives, and nouns.
- *Pronominal suffix/enclitic:* pronominal complements with verbs and prepositions, or possessive pronouns with nouns.
- A feature indicating whether a noun, adjective, or numeral is a construct state or an absolute form.[1]

Except for the definiteness marker in Hebrew (see below), these word parts are simply concatenated in a strict order.[2] For instance, the word *wfnpgfnw* ('and that we met', pronounced *ve-she-nifgashnu*)[3] is traditionally analyzed as follows:

Example 1

- *Prefixes:* the conjunction *w* ('and') and the complementizer *f* ('that')
- *Stem:* the verb *npgf* (past tense of 'meet')
- *Inflectional suffix: nw* ('we')

Some parts of this analysis are very robust and do not depend on the tagset or conventional notations. For instance, the identification of the prefixes *w* and *f* as separate segments, with their independent POS tags, is something that any linguistic analysis of Hebrew texts should assume. These morphemes are typical clitics: they

[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] Importantly, the intenal morphology of the stem (see below) is much more complicated, and its analysis in terms of root and template/pattern involves non-concatenative processes.

[3] See Table A1 in Appendix A2 for the Hebrew/Latin transliteration that we use in this work.

Table 1. *Some usages of the Hebrew definiteness marker*

Text	Segmentation	Meaning
hbit	h-bit	'the-house' (unambiguous)
fhbit	f-h-bit	'that-the-house' (unambiguous)
bbit	b(-h)-bit	'in(-the)-house' (ambiguous)
bbit gdwl	b-bit gdwl	'in-house big' (disambiguated: 'in a big house')
bbit hgdwl	b-h-bit h-gdwl	'in-the-house the-big' (disambiguated: 'in the big house')

have an independent syntactic role but are phonologically bound to the stem. Other decisions concerning the annotation scheme, especially concerning the status of the inflectional suffix, are sometimes less clear cut. For instance, in the verbal form *npgfnw*, whether to analyze the pronominal suffix *nw* as a separate word segment or as an inflectional part of the verbal stem may depend on the task at hand. In this work, we analyze inflectional suffixes as part of the stem, which is compatible with analysis of inflectional suffixes as part of the word in English and other languages.

Another non-trivial morphological phenomenon in Hebrew and other Semitic languages concerns the analysis of the templates and patterns for verbs, nouns, and adjectives. For instance, the meaning and inflectional features (tense, number, gender, person) of the verb *npgf*(*nw*) are identified using a set of morphological rules for the Hebrew verbal templates. These rules derive different verbs from one abstract *root* — a sequence of (usually three) consonants. In the case of the verb form *npgf*(*nw*), the root is *pgf*, and the template formation rule $n + \langle\mathsf{root}\rangle$ (traditionally called '*npel*') is responsible for forming the verbal stem *npgf*. Full morphological analysis of a Semitic word should include its root and template/pattern, but the usefulness of this morphological information may depend on the task at hand (see more on this point below).

One crucial problem for morphological disambiguation in Hebrew texts, which plays a special role in this paper, is the possible omission of the definite article. This prefix can be covert, which happens (only) when it appears together with one of three prepositional prefixes ($b =$'in', $l =$'to' or $k =$'like/as'). The presence of a covert definiteness marker can often be recovered using the context. Some examples for the usage of the Hebrew definiteness marker are given in Table 1. Note that the ambiguity here is only due to the lack of vocalization in the modern Hebrew writing system, and ambiguous occurrences of words such as *bbit* have two different pronunciations (*babayit* or *bebayit*). Note further that in Hebrew (as in Arabic), modifying adjectives are marked for definiteness, and their marking should agree with the definiteness of the noun they modify. This facilitates the disambiguation of modified nouns that are orthographically ambiguous with regard to definiteness.

2.2 An annotation scheme for Hebrew POS tagging

From Example 1 above, it is clear that the morphological analysis of a Hebrew word may involve more than one element that has a syntactic role. In addition, the conjunction, the complementizer, and the verb may belong to different constituents in the syntactic structure, and may therefore need to have separate POS tags in any syntactic representation that contains the word *wfnpgfnw*. More generally, for Hebrew POS tagging we need to define the possible word segments and the POS tag that is assigned to any such word segment. By the term *word segmentation*, we henceforth refer to identifying the prefixes, the stem, and various suffixes of the word. By *POS tag disambiguation*, we mean the assignment of a proper POS tag to each of the identified word segments.

The task of segmentation and POS tag disambiguation is closely related to morphological analysis and disambiguation, but the two tasks are different at some respects. First, the internal analysis of the stem can be ignored for the sake of analyzing its POS tag. This is because the root and pattern/template morphemes are phonologically and syntactically bound, and the syntactic function of the stem is derived from them in a complex manner that need not concern applications such as syntactic parsing. Conversely, some syntactic distinctions that are relevant for POS tagging are not crucial for morphological disambiguation. For instance, the clitic *f* ('that') in Hebrew, like its parallel in English, is ambiguous between a complementizer and a relative pronoun. This ambiguity should be resolved by a syntactic parser or a POS tagger, but is not crucial for the internal morphological analysis of Hebrew words. Whether full morphological disambiguation is required in addition to POS tagging depends on the target application. For instance, full morphological disambiguation is important for machine translation and text-to-speech applications for Semitic languages. However, for syntactic parsing, chunking or information extraction, the POS tagging scheme that we describe below may often be sufficient.

For the purpose of our POS tagging scheme, we ignore part of the information that is found in the common morphological scheme shown in Section 2.1. The internal morphological structure of stems is not analyzed, and the POS tag that is assigned to a stem includes no information about its root, template/pattern, inflectional features, and suffixes. Among the suffixes, only pronominal complement suffixes on verbs and prepositions are identified as separate word segments. The construct state/absolute feature 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 below by the segmentation and POS tagging of the word *wfnpgfnw*, as opposed to its traditional morphological analysis in Example 1.

Example 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), as described below. Ignoring part of the morphological information in the analysis allows us to achieve reduced sparseness when training a POS tagging model on the small annotated corpus that is available for Hebrew, but may result in analyses that are not fully disambiguated morphologically. However, it is interesting to note that only 6.6% of the words that were tagged correctly by our system were not fully disambiguated. If one of the possible full analyses is picked randomly in such cases, then approximately 96.4% of the correct analyses can be expanded correctly to a full morphological analysis. Therefore, tagging using our scheme can be significantly helpful as a first stage in full morphological disambiguation. Having said that, in this paper, we concentrate on the tasks of word segmentation and POS tagging, and leave the study of full morphological disambiguation to further research.[4]

2.3 A note on the POS tag annotation scheme for Arabic

Despite the similarity between the writing systems and morphology of Hebrew and Arabic, two differences should be mentioned. First, unlike the Hebrew definiteness marker (*h*), which can be covert, the definiteness marker *al*/*l* is always present in the Arabic writing system. Furthermore, the Arabic treebank (Maamouri *et al.* 2004) does not separate the *al*/*l* prefix from the stem following it. This convention helps POS tagging in Arabic when compared with Hebrew. In contrast, however, pronominal suffixes of nouns and verbs are much more productive in Arabic than in Hebrew. In the Hebrew treebank, unlike the Arabic treebank, pronominal suffixes are scarce on nouns and virtually nonexistent on verbs (cf. Table A6 below). In this respect, POS tagging of nouns and verbs may often be easier than in Arabic.

3 Available corpora

One of the problems of dealing with Hebrew is the lack of large annotated corpora. The Hebrew treebank (Sima'an *et al.* 2001) consists of syntactically annotated sentences taken from articles from the *Ha'aretz* daily newspaper. The treebank version that was used in the current work contains 57 articles, which amount to 1,892 sentences, 35,848 words, and 48,332 word segments.[5] In addition to the manually tagged corpus, we have access to an untagged corpus containing 337,651 words, also originating from *Ha'aretz* newspaper.

To obtain a training and testing corpus for POS tagging, we extracted from the Hebrew treebank a mapping from each word to its analysis as a sequence of POS tagged segments. The simplified tagset that was obtained in this way contains 24 categories, ignoring the gender, number, person, and tense features in the treebank POS tags. Thirteen additional tags are used for various symbols, and for marking beginning and end of sentence. See Appendices B and C for more data and explanations about the tagset.[6]

[4] See Habash and Rambow (2005) and Adler and Elhadad (2006) for recent works that deal with morphological disambiguation and POS tagging using the same model.

[5] Recently, a new version of the Hebrew treebank has become available. For some preliminary results on this bigger dataset that support our main conclusions in this paper, see Section 7.2 below.

[6] The corpus version used in this paper is available at http://www.cs.technion.ac.il/∼barhaim/MorphTagger

4 Architectures for POS tagging Semitic languages

Our segmentation and POS tagging system uses a morphological analyzer that assigns a set of candidate morphological analyses to each word. The morphological analyses of each word are then mapped to a set of candidate POS taggings of the word. Each such candidate tagging consists of a segmentation of the word and a POS tag assigned to each of the resulting segments. The mapping from morphological analyses to candidate POS taggings is defined using the same annotation scheme that was used for the Hebrew treebank (see Section 3). A disambiguator selects from the set of candidate POS taggings a single preferred analysis for each word. In this section, we concentrate on the architectural decisions in devising an optimal disambiguator, given a morphological analyzer.

4.1 Defining the input/output

As indicated in Section 1, word-level and segment-level tokenizations are the simplest options for defining the tokenization level for POS tagging of Hebrew and other Semitic languages. Before implementing these two alternatives and developing further variations, let us see how the choice between them directly affects the terminal and the nonterminal (output) symbols for POS tagging.

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 segment 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-the-house’). 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 $\boldsymbol{a} = [t_1 \ldots t_p]$ for w, it is possible to extend $\boldsymbol{a}$ back to a full analysis $\tilde{\boldsymbol{a}} = [(s_1, t_1) \ldots (s_p, t_p)]$, which includes the segments $s_1 \ldots s_p$ that make out w. This is done by finding a full analysis $\tilde{\boldsymbol{a}}$ in $Analyses(w)$, the set of possible analyses for w, such that the tagging of its segments is the same as the tag sequence in $\boldsymbol{a}$. Except for very rare cases, this match is unique. As an example for the way in which POS tag sequences can be expanded to analysis, consider the word *lmrwt*, which has the following analyses according to our scheme:

(1) *lmrwt*/RB (‘although’)
(2) *l*/IN *mrwt*/NN (‘to authority’, indefinite)
(3) *l*/IN *mrwt*/NNT (‘to the authority of’, construct form)
(4) *l*/IN *h*/H *mrwt*/NN (‘to the authority’, definite)
(5) *l*/IN *mrwt*/JJ (‘to bitter’, feminine, plural)
(6) *l*/IN *mrwt*/JJT (‘to bitter’, construct form, feminine, plural, indefinite)
(7) *l*/IN *h*/H *mrwt*/JJ (‘to the bitter’, feminine, plural, definite)

Table 2. n-*gram counts for each tokenization level*[*].

Level of tokenization	Unigrams	Bigrams	Trigrams	Tokens
Word segment	37	644	**4,109**	39,282
Word	185	**2,539**	10,496	28,738

[*]For each level of tokenization, the n-gram order actually used is marked in boldface.

Removing the segments results in the following set of analyses: {RB, IN-NN, IN-NNT, IN-H-NN, IN-JJ, IN-JJT, IN-H-JJ}. It can be seen that each of these seven tag sequences corresponds to exactly one full analysis.

Word segments (WS)

Assuming a segmentation of the input words, the terminals of the tagger are segments, and the nonterminals are the usual POS tags. Note that in this case, information about how segments combine into words is not directly available for the disambiguator.

Table 2 shows the number of different types (unigrams) found in the training corpus for the word and segment tokenization levels.

It should be stressed that word-level and segment-level tokenizations are two extremes, and the optimal level of tokenization that we propose for Hebrew POS tagging lies somewhere between them. However, the W/WS opposition is useful for the time being for introducing the key concepts of the proposed architecture.

4.2 The probabilistic framework

Let w_1^k be the input sentence, a sequence of words $w_1 \ldots w_k$. With word-level tokenization, the disambiguator aims at finding the analysis sequence $\boldsymbol{a}_1^k$ that has the highest probability given the sentence w_1^k:

$$\arg\max_{\boldsymbol{a}_1^k} P\left(\boldsymbol{a}_1^k | w_1^k\right) = \arg\max_{\boldsymbol{a}_1^k} P\left(w_1^k, \boldsymbol{a}_1^k\right). \tag{1}$$

This is the standard formulation of probabilistic tagging for languages such as English.

Alternatively, with segment-level tokenization, the disambiguator aims at finding a combination of a segmentation s_1^n and a tagging t_1^n for s_1^n, such that their joint probability with the given sentence, w_1^k, is maximized:

$$\arg\max_{(s_1^n, t_1^n) \in ANALYSES(w_1^k)} P\left(w_1^k, s_1^n, t_1^n\right), \tag{2}$$

where $ANALYSES(w_1^k)$ is the set of candidate 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 $\langle s_1^n, t_1^n \rangle$ pairs that constitute the input to the disambiguator were derived from w_1^k, we have $P(w_1^k | s_1^n, t_1^n) = 1$, and thus

$P(w_1^k, s_1^n, t_1^n) = P(t_1^n, s_1^n)$. Therefore, (2) can be simplified as follows:

$$\underset{(s_1^n,t_1^n)\in ANALYSES(w_1^k)}{\arg\max} \quad P\left(s_1^n, t_1^n\right). \tag{3}$$

(1) and (3) can be represented in a unified formula that applies to both word-level and segment-level tokenizations:

$$\underset{(e_1^n,A_1^n)\in ANALYSES(w_1^k)}{\arg\max} \quad P\left(e_1^n, A_1^n\right). \tag{4}$$

In (4), e_1^n represents either a sequence of words or a sequence of segments, depending on the level of tokenization. The nonterminals A_1^n are the respective analyses of these terminals—POS tag sequences or POS tags, respectively. The disambiguator aims at finding the most probable pair $\langle$*terminal sequence, nonterminal sequence*$\rangle$ for the given sentence, whereas in the case of word-tokenization, there is only one possible input terminal sequence for the sentence.

4.3 HMM probabilistic model

The actual probabilistic model used in this work for estimating $P(e_1^n, A_1^n)$ is based on HMMs. HMMs underly many successful POS taggers for different languages, for example DeRose (1988), Church (1988), Cutting *et al.* (1992), Charniak *et al.* (1993), Weischedel *et al.* (1993), and Merialdo (1994) for English, Dermatas and Kokkinakis (1995) for Dutch, English, French, German, Greek, Italian and Spanish, Brants (2000) for German and English, Hakkani-Tür *et al.* (2000) for Turkish, and many more.

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

$$\underset{e_1^n,A_1^n}{\arg\max}\, P\left(e_1^n, A_1^n\right) \approx \underset{e_1^n,A_1^n}{\arg\max} \prod_{i=1}^{n} P(A_i|A_{i-k},\ldots,A_{i-1})P(e_i|A_i). \tag{5}$$

For reasons of data sparseness, the instance models employed here work with $k = 2$ for the segment-level tokenization, and with $k = 1$ for the word-level tokenization. It is crucial at this point to observe that a Markov model at the word-level tokenization employs different statistics than a Markov model of the same order at the segment-level tokenization. Like in the case of English, the Hebrew segment tokenization working with a first-order Markov model conditions a single POS tag over a single preceding POS tag. In contrast, for the Hebrew word-level tokenization, a first-order Markov model will condition a sequence of POS tags for the current word on another sequence of POS tags for the preceding word. For example, consider again the phrase ‘bbit hgdwl’ from Table 1, tagged as *bbit*/IN-H-NN *hgdwl*/H-JJ. First-order Markov model will condition H-JJ on IN-H-NN (i.e. estimate P(H-JJ | IN-H-NN)).

For both models of HMM-based POS tagging, two kinds of probabilities need to be estimated: $P(e_i|A_i)$ (the lexical model) and $P(A_i|A_{i-k},\ldots,A_{i-1})$ (the language model). Because the only manually POS tagged corpus that is available to us for training the HMM is relatively small ($<$ 4% of the *Wall Street Journal WSJ* portion of the Penn treebank), major effort must be dedicated to alleviating the sparseness problems that arise. For smoothing the language model probabilities over nonterminals, 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 in our setting did not appear in the training corpus. Next, we describe a novel method for smoothing the lexical probabilities using an untagged corpus.

4.4 Bootstrapping a better lexical model

We use an unsupervised training method for smoothing the lexical model acquired from the annotated data. For the sake of exposition, we assume word-level tokenization in this subsection. The method used for smoothing the segment-level tagger is very similar.

The smoothing of the lexical probability of a word w given an analysis $\mathbf{a}$, that is, $P(w|\mathbf{a}) = P(w,\mathbf{a})/P(\mathbf{a})$, is accomplished by smoothing the joint probability $P(w,\mathbf{a})$ only, that is, we do not smooth the relative frequency estimate of $P(\mathbf{a})$.[7] For smoothing $P(w,\mathbf{a})$, 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 $\mathbf{em}_{auto}(w,\mathbf{a})$):

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

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

Our unsupervised estimation method for obtaining $\mathbf{em}_{auto}(w,\mathbf{a})$ starts out with a naively smoothed POS tagger, and then tries to reestimate the lexical model probabilities of this POS tagger on an untagged corpus. In fact, this method can be viewed as a single iteration of the Baum–Welch (forward–backward) estimation algorithm (Baum 1972) with minor differences. We apply this unsupervised method to an untagged corpus of 337K words.

The reestimation method starts out with a POS tagger that employs naively smoothed relative frequency estimates of the lexical model probabilities:

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

where $f_{tr}(w)$ is the frequency of w in the training corpus and p_0 is a constant (fixed at 10^{-10} in our experiments). We use the notation P_{basic} to denote the POS tagger that combines a Katz backoff smoothed language model together with the naively smoothed lexical model P_{LM_0}.

The unsupervised reestimation algorithm works in two steps. In the first step (expectation calculation), the model P_{basic} is employed in order to generate a *distribution of alternative analyses* (segment-tag sequences) for each of the sentences in the untagged corpus.[8] This way the untagged (incomplete) corpus is transformed into an

[7] The smoothed probabilities are normalized so that $\sum_w P(w,\mathbf{a}) = P(\mathbf{a})$.
[8] We approximate this distribution by the 300 most probable analyses per sentence.

ambiguously tagged (complete) corpus, containing for every sentence a distribution of analyses (i.e., segment-tag sequences), each weighted with its probability according to the model P_{basic}. Subsequently, in the second step (maximization), the ambiguously tagged, completecorpus is used to reestimate the lexical model probabilities utilizing relative frequency estimation (which in the case of a complete corpus is equivalent to maximum-likelihood estimation). Note that because this is an unsupervised training, the test set sentences may be added to the untagged corpus to ensure that the model assigns non-zero probabilities to the test set words.[9]

This smoothing method can be seen as a variant of the well-known expectation-maximization (EM) method (Elworthy 1994; Merialdo 1994) for smoothing lexical probabilities using unlabeled data. In this approach, we initialize the model with relative frequency estimates from labeled data and reestimate the probabilities by running the EM algorithm on the unlabeled data. Our proposed method differs from this classical approach in the following points: First, the use of a weighting factor to control the influence of the unlabeled data, following (Nigam *et al.* 2000). Second, we perform only one cycle of reestimation. This is justified by Merialdo's finding that additional iterations do not improve accuracy when starting with a relatively small tagged corpus. Finally, unlike standard EM that affects all model parameters, we smooth only the lexical probabilities, while using standard backoff methods to smooth language model probabilities. This is justified by the fact that sparseness is far more severe for lexical probabilities than for language model probabilities. To the best of our knowledge, this aspect of our method of lexical smoothing is novel.

5 Implementing different models for Hebrew POS tagging

After having introduced the basic ingredients of our POS tagging procedure, we move on to their implementation, which involves two subtasks: extracting candidate POS tag sequences from the output of a morphological analyzer and implementing the POS tag disambiguator for each of the two models.

5.1 From morphological analayses to candidate POS taggings

The set of possible morphological analyses was obtained from Segal's morphological analyzer (Segal's 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. Segal's analyses follow the scheme

[9] Adding the test sentences to the untagged corpus for improved reestimation can be implemented efficiently enough that it can be applied during run time for the tagging of running text. Since the expectation calculation step is done independently for each untagged sentence (recall that the probabilistic model used in this step is based solely on the tagged corpus), the expectation can be calculated in an accumulative fashion. For the large untagged corpus, it can be computed offline during training. At run time, the expectation calculation is conducted only for the sentences in the running text, which amounts to running the Viterbi algorithm on the running text sentences (linear time complexity), and taking the N best taggings. Finally, in the maximization step, $\mathbf{em}_{auto}(w, \mathbf{a})$ is calculated for words occuring only in the running text, by combining the probabilities estimated offline from the untagged corpus with the probabilties computed online from the running text.

described in Section 2.1. Translating them to the POS tagging scheme described in Section 2.2 involves three elements:

(1) Translating the POS of the stem to the treebank tagset. Since this tagset is more fine-grained than Segal's, the translation is one-to-many in some cases, thereby increasing the ambiguity. The treebank scheme uses different POS tags to represent some of the morphological features. For example, nouns in the absolute/construct state are represented as NN/NNT, respectively.
(2) Creating separate segments for the prefixes, along with their POS tag. This mapping can be ambiguous as well: according to the treebank tagset, the prefix *f* may be tagged either as a complementizer (COM) or as a relativizer (REL).[10]
(3) Creating suffix segments for pronominal complement suffixes of verbs and prepositions.

Out of vocabulary (OOV) words (words that are not recognized by the morphological analyzer) are assumed to be proper nouns[11] and the morphological analyzer proposes possible segmentations for the word on the basis of the recognition of possible prefixes. We enhanced this to allow analysis as *'h'+common noun* in case the OOV word begins with *h*. Furthermore, we applied a simple heuristic for reducing the number of possible segmentations for OOV words. The idea is that if two OOV words differ only in their (presumed) prefixes, it is more likely that they share the same stem and differ in their prefixes than that they have similar but different stems. For example, if the OOV word is *wlalprd*, 'and to Alfred', the letters *w,l* can *be either prefixes* or part of the stem. Now, if we see another OOV word *malprd, 'from Alfred'*, we may assume with high probability that letters *w,l* are indeed prefixes, and eliminate segmentations in which these letters are part of the stem. The heuristic makes use of a collection of OOV words, extracted from a raw corpus containing 11 million words. The contribution of the OOV heuristic is analyzed in Section 6.6.

5.2 Implementing word-level and segment-level disambiguators

We used the SRI Language Modeling (SRILM) Toolkit (Stolcke 2002) for constructing smoothed language models for tag sequences, and for finding the most probable tag sequence, we used Viterbi decoding (Viterbi, 1967). We created from the small tagged corpus, a backoff language model for tag sequences, using Good-Turing discounting (Good, 1953) with Katz backoff (Katz, 1987).

The construction of word-level (W) HMMs is well studied in the literature and is straightforward in SRILM. This is the standard HMM implementation that works with unambiguous input (a finite sequence of terminals). Clearly, this

[10] For example, in *amr fdn hlx* ('said that—Dan went'), the prefix *f* in *fdn* is tagged as COM. By contrast, in *haif fdn awhv* ('the—man that—Dan likes') the same prefix is tagged as REL. Hebrew morphological analyzers usually do not make this distinction, although it is important for syntactic analysis and for applications such as machine translation from Hebrew to English.

[11] Unlike English, proper nouns are not orthographically marked as such in Hebrew.

implementation is not directly suitable for modeling segment-level HMM taggers, since the disambiguator has to optimize not only the nonterminal sequence but also the terminal sequence (which corresponds to the segmentation).

For modeling the segment-level tagger, we simulate segment tagging by a word tagger, where the terminals are words and the nonterminals are analyses (tag sequences). As explained in Section 4.1, for a given word w and an analysis $\boldsymbol{a} = [t_1 \ldots t_p]$ for w that contains tags without the corresponding word segments, it is possible to extend $\boldsymbol{a}$ to a unique analysis $\tilde{\boldsymbol{a}} = [(s_1, t_1) \ldots (s_p, t_p)]$, where $s_1 \ldots s_p$ are the respective segments. Therefore, at the sentence level, it is easy to move back and forth between segment-level analyses of the form (s_1^n, t_1^n) and word-level analyses of the form $(w_1^k, \boldsymbol{a}_1^k)$. To simulate a segment tagger, we now only have to change the calculation of the probabilities $P(\boldsymbol{a}_i|\boldsymbol{a}_{i-2}, \boldsymbol{a}_{i-1})$ and $P(w_i|\boldsymbol{a}_i)$, so that $P(w_1^k, \boldsymbol{a}_1^k) = P(s_1^n, t_1^n)$ for each $(w_1^k, \boldsymbol{a}_1^k)$ and the corresponding (s_1^n, t_1^n).

The probabilities are calculated as follows. First, we calculate segment-level probabilities: language model probabilities for tag sequences $P_{ws}(t_i|t_{i-2}, t_{i-1})$ and lexical probabilities for segment-tag pairs $P_{ws}(s_i|t_i)$. The next step is to calculate $P(\boldsymbol{a}|\boldsymbol{a}_{i-2}, \boldsymbol{a}_{i-1})$ and $P(w_i|\boldsymbol{a}_i)$:

For $P(\boldsymbol{a}_i|\boldsymbol{a}_{i-2}, \boldsymbol{a}_{i-1})$: Let $\boldsymbol{a}_i = [t_1, \ldots, t_p]$, $\boldsymbol{a}_{i-1} = [y_1, \ldots, y_q]$, and $\boldsymbol{a}_{i-2} = [x_1, \ldots, x_r]$. Let $\boldsymbol{u}$ be the concatenation of $\boldsymbol{a}_{i-2}$, $\boldsymbol{a}_{i-1}$, and $\boldsymbol{a}_i$:

$$\boldsymbol{u} = [u_1, \ldots, u_{r+q+p}], \tag{8}$$

where

$$u_i = \begin{cases} x_i & 1 \le i \le r \\ y_{i-r} & r+1 \le i \le r+q \\ t_{i-r-q} & r+q+1 \le i \le r+q+p \end{cases} \tag{9}$$

Then,

$$P(\boldsymbol{a}_i|\boldsymbol{a}_{i-2}, \boldsymbol{a}_{i-1}) = \prod_{i=r+q+1}^{r+q+p} P_{ws}(u_i|u_{i-2}, u_{i-1}). \tag{10}$$

Unlike standard n-gram backoff models that contain probabilities only for tag sequences found in the corpus, here we need to compute this probability for each possible trigram $\boldsymbol{a}_{i-2}\boldsymbol{a}_{i-1}\boldsymbol{a}_i$, even if no subsequence of this trigram appears in the corpus. Thus, computing all these probabilities offline is inefficient, and will result in a very large model. To save run time and space, these trigram probabilities are derived on demand at run time; for each input sentence, based on the segment-level probabilties (P_{ws}), computed at training time.

For $P(w_i|\boldsymbol{a}_i)$: Let $\tilde{\boldsymbol{a}}_i = [(s_1, t_1) \ldots (s_p, t_p)]$ be the full analysis, obtained from w_i and $\boldsymbol{a}_i$. Then, $P(w_i|\boldsymbol{a}_i)$ is given by :

$$P(w_i|\boldsymbol{a}_i) = \prod_{i=1}^{p} P_{ws}(s_i|t_i). \tag{11}$$

It can be verified that multiplying the probabilities in (10) and (11) together gives the path probability according to the segment HMM model (see Equation 3).

Table 2 shows the average number of unigrams, bigrams, and trigrams in the training set for each tokenization level (this is an average over the five training sets in our cross-validation experiments, described in the next section). The relevant number for the actual *n*-gram size used for each level of tokenization appears in boldface. The table clearly shows the influence of the tokenization level on the data sparseness: word-level models suffer increased data sparseness in comparison with segment-level models. These figures support our choice to use bigrams for the word level and trigrams for segment level, since trigrams seem too sparse for the word level. This decision was also supported empirically, as we found that using trigrams does not improve the accuracy for the word-level tagger.

6 Evaluation: The advantages of a segment-level tagger with a definiteness feature

In this section, we report on an empirical comparison between the two levels of tokenization presented in the previous sections, and study the proper way to handle the Hebrew definiteness marker in the segment-level model. Analysis of the results shows that a segment-level model that treats the definiteness marker as a feature, rather than a separate segment, is advantageous to both of the initial models with word-level and segment-level tokenizations.

We start by introducing the setting for the different experiments, and the results obtained with some baseline models. After evaluating the various tokenization architectures, we present an evaluation of the bootstrapping of lexical model probabilities and the OOV heuristic, which enhance the standard model of HMM tagging. The results show that both of these enhancements are valuable and contribute to the overall accuracy of the best model found.

6.1 Experimental setting

Each architectural configuration was evaluated in five-fold cross-validated experiments. In a train/test split of the corpus, the training set includes 1,598 sentences on average, which amount to 28,738 words and 39,282 word segments. The test set includes 250 sentences.

Per word measures

We measure *segmentation accuracy*—the percentage of words correctly segmented, as well as *tagging accuracy*—the percentage of words that were correctly segmented for which each segment was assigned the correct POS tag. Thus, using this measure a single segmentation/tagging error in one of the segments invalidates the whole word.

Per segment measures

We also report the $F_{\beta=1}$ measure, defined as:

$$F_{\beta=1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{12}$$

Table 3. *Baseline results*

System	Accuracy per word (%)		$F_{\beta=1}$ per segment (%)	
	Tagging	Segmentation	Tagging	Segmentation
Most frequent	82.71 (0.8)	95.08 (0.4)	85.34 (0.9)	95.73 (0.5)
Most frequent with gold segmentation	84.57 (1.1)	100.00 (0.0)	88.43 (0.9)	100.00 (0.0)
Segal	85.22 (2.0)	95.23 (0.4)	87.74 (1.7)	95.67 (0.3)

where *segmentation (tagging) recall* is defined as the length of the longest common subsequence (LCS) of the output (tagged) segment sequence (WSS) and the gold (tagged) WSS, divided by the length of the gold WSS. Similarly, precision is obtained by dividing the LCS length by the length of the output WSS.

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.

6.2 Baseline models

We tested two baseline taggers: one is based on choosing the most frequent analysis and the other is a POS tagger based on a morphological disambiguator developed by Segal's (2000), (see Section 7.1). The most frequent tagger operates at the word level and implements the following simple rule:

$$\hat{\boldsymbol{a}}(w) = \begin{cases} argmax_{\boldsymbol{a} \in Analyses(w)}[\mathbf{rf}_{tr}(w, \boldsymbol{a})] & \text{if } f_{tr}(w) > 0 \\ argmax_{\boldsymbol{a} \in Analyses(w)}[\mathbf{rf}_{tr}(\boldsymbol{a})] & \text{otherwise} \end{cases} \tag{13}$$

where $\mathbf{rf}_{tr}(\boldsymbol{a})$ is the relative frequency of $\boldsymbol{a}$ in the tagged corpus. The most frequent model was also tested on POS tag disambiguation alone, where the input is the correct sequence of segments. In this case, the above formula was applied per segment, rather than per word.

The accuracy of Segal's tagger on our data sets is another reference point. The training corpus was translated to Segal's scheme, and the results were translated back to the treebank's morphological format.[12]

The baseline results are listed in Table 3. The tagging accuracy per word achieved by the most frequent method (82.71%) is relatively low, as compared with the 90% or higher accuracy achieved by similar methods for English (Charniak *et al.* 1993). Table 4 shows some examples for tagging errors. Since an English word corresponds to a word segment in Hebrew, it might be more appropriate to use the $F_{\beta=1}$ measure per segment for the comparison with English. The result (85.34%) is still considerably lower than the equivalent for English. This difference supports our

[12] In case of ambiguity, we chose the most probable translation, estimated from the training corpus.

Table 4. *Some examples for tagging errors with the most frequent tagger.*[*]

Word	Correct (occurrences)	Most frequent (occurrences)	Error count
iwm	*iwm*/NNT (4) ‘day’, construct	*iwm*/NN (5) ‘day’, absolute	8
hia	*hia*/AGR (38) ‘is’, feminine, 3rd person, single	*hia*/PRP (40) ‘she’	8
snj	*snj*/NN (2,400 - for NN) ‘cent’	*snj*/VB (2,690 - for VB) ‘teased’	3

[*]The number of occurrences in the training corpus is given in parentheses. The word ‘snj’ does not appear in the training corpus, therefore the frequencies of ‘NN’ and ‘VB’ are compared.

Table 5. *Level of tokenization – experimental results*

	Accuracy per word (%)		$F_{\beta=1}$ per segment (%)	
System	Tagging	Segmentation	Tagging	Segmentation
W	88.50 (1.0)	96.74 (0.3)	90.69 (0.9)	97.32 (0.3)
WS	89.27 (1.1)	96.55 (0.3)	91.51 (1.0)	97.40 (0.3)
WS+*h*	89.59 (1.1)	97.05 (0.3)	91.59 (1.0)	97.57 (0.3)

conjecture that the task of POS tagging in Hebrew is relatively difficult, given the small annotated corpus we have. The results per segment obtained with the gold segmentation (88.43%) are closer to the results for English.

The tagging accuracy per word of 85.22% obtained for Segal’s tagger is distinctly incompatible with the 96.2% reported in Segal’s (2000), presumably for the following reasons. First, the translation from Segal’s annotation scheme to the treebank format is imperfect. Second, Segal’s scheme is not flexible enough to describe every combination of segments found in Hebrew (and in the corpus). But, presumably, the most important reason for the difference is that Segal ensured that the right analyses for all the test words always appear in the dictionary. In practice, however, for about 4% of the test words, the right analysis does not appear in the dictionary, even after it is complemented with the training corpus.

6.3 Evaluating the word-level and segment-level models

The first two lines in Table 5 detail the results obtained for both word (W) and word segment (WS) levels of tokenization. The tagging accuracy of the segment tagger is considerably better than that achieved by the word tagger (difference of 0.77% with significance level $p = 0.019$). 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:

The segment-level tagger outperforms the word-level tagger in its tagging accuracy since it suffers less from data sparseness. However, it lacks some word-level knowledge that is required for segmentation.

The hypothesis about the relative data sparseness in the word-level model is supported by the number of once occurring terminals at each level: 8,582 at the word level versus 5,129 at the segment level.

6.4 Error analysis and discussion

Motivated by this hypothesis, we next consider what kind of word-level information is required for the segment-level tagger in order to do better in segmentation. Error analysis revealed a very common type of segmentation errors, which was found to be considerably more frequent in segment tagging than in word tagging. This kind of errors involves words such as *bbit* ('in-the-house'/'in-a-house', cf. Table 1), where the definiteness marker *h* may be textually covert. The errors detected miss a covert *h* or wrongly add an unnecessary *h* to the analysis. 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 segment tagging, this type of error was found on average in 1.71% of the words (50% of the segmentation errors). In word tagging, it was found only in 1.37% of the words (42% of the segmentation errors).

Since in Hebrew, there should be agreement between the definiteness status of a noun and its related adjective, definiteness ambiguities of nouns can sometimes be resolved syntactically. For instance, reconsider the disambiguated cases in Table 1:
'bbit hgdwl' implies *b-h-bit* ('in the big house')
'bbit gdwl' implies *b-bit* ('in a big house')

In 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 the contextual information that is not available to the POS tagger.

Lexical statistics can provide valuable information in such situations, since some nouns are more common in their definite form, whereas 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 as *l-h-mmflh*. The stem *mmflh* ('government') was found 25 times in the corpus, out of which only 2 occurrences were indefinite. This strong lexical evidence in favor of *l-h-mmflh* is completely missed by the segment-level tagger, in which segments are assumed to be unrelated. 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-mmflh* 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.

Table 6. *Representation of* l-h-mmflh *in each level of tokenization*

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

Table 7. *WS* + h *model: contribution of bootstrapping and the OOV heuristic*

	Accuracy per word (%)		$F_{\beta=1}$ per word segment (%)	
System	Tagging	Segmentation	Tagging	Segmentation
WS + *h*	87.96 (1.2)	96.09 (0.5)	89.99 (1.1)	96.63 (0.5)
WS + *h* + bootstrapping	89.49 (1.2)	96.76 (0.4)	91.36 (1.0)	97.21 (0.4)
WS + *h* + bootstrapping + OOV	89.59 (1.1)	97.05 (0.3)	91.59 (1.0)	97.57 (0.3)

6.5 Evaluating the WS + h *model*

To better model such situations, we changed the segment-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 WS + h to the resulting model that uses this assumption, which is rather standard in theoretical linguistic studies of Hebrew (Wintner 2000; Danon 2001). The WS + *h* model can be viewed as an intermediate level of tokenization, between segment-level and word-level tokenizations. In Table 6, we demonstrate the different analyses that are obtained by the three models of tokenization.

As shown in Table 5, the WS + *h* model shows remarkable improvement in segmentation (0.50%, $p = 0.0018$) compared with the initial segment-level model (WS). As expected, the frequency of segmentation errors that involve covert definiteness (*h*) dropped from 1.71% of the test words to 1.22%. The adjusted segment tagger also outperforms the word-level tagger in segmentation (by 0.31%, $p = 0.049$). Tagging improved as well, although the significance level is low (0.32%, $p = 0.081$). According to these results, tokenization as in the WS + *h* model is preferable to both plain-segment and plain-word tokenizations.

6.6 The contribution of the individual components

Having determined the optimal level of tokenization, we would now like to measure the contribution of individual components to the overall accuracy of our best model. We focus on the smoothing of the lexical model using untagged data (see Section 4.4), and the heuristic for reducing the number of possible segmentations for OOV words (see Section 5.1). The results are shown in Table 7. The first model uses a baseline lexical model, based on maximum-likelihood estimation from the tagged corpus (the model is similar to LM_0 defined in Section 4.4, except that the level of tokenization here is word segments, rather than words). The results show that bootstrapping (second line) improves tagging by 1.53% and segmentation by 0.67%. The OOV heuristic (third line) further improves the segmentation accuracy by 0.29% and tagging by 0.1%. The improvements are all statistically significant ($p < 0.05$).

7 Discussion

7.1 Related work

We discuss related work on Hebrew, Arabic, and other languages separately.

Hebrew morphological disambiguation: Because of the lack of substantial tagged corpora, most previous corpus-based work on Hebrew focused on inducing probabilities from large unannotated corpora. Similarly to the present work, a morphological analyzer was often used for providing candidate analyses.

Levinger *et al.* (1995) propose a method for choosing the most probable analysis using an unannotated corpus, where each analysis consists of the lemma and a set of morphological features. For each analysis A of a word w, they define a set of "similar words" $SW(A)$. Each word w' in $SW(A)$ corresponds to an analysis A' that 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 a 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. They report accuracy of 86% when choosing the most probable pattern.

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 the "similar words" method, two additional components: correction rules *à la* Brill (1995) and a rudimentary deterministic syntactic parser. See section 6.2 for an empirical study of Segal's system.[13]

An early short version of the present work appeared in Bar-Haim *et al.* (2005).[14] Most recently, Adler and Elhadad (2006) presented new results on unspervised HMM

[13] A related problem in the morphological disambiguation of Semitic languages is root identification (Daya *et al.* 2004). We do not discuss such work here since it is partly independent of POS tagging and syntactic parsing: the root and template/pattern morphemes of the Semitic stem normally form together a unit that is syntactically unanalyzed.

[14] The present paper develops this early version in a number of respects. Beside the more rigorous detail of the novel models and algorithms, the present paper provides the results of new and more thorough experiments, and it covers much better the linguistic background and the related work.

tagging and segmentation, using only a morphological analyzer and a large unlabeled corpus, containing 6 million words (17.6 times bigger than the one we used). Levinger et al.'s "similar words" method was used to obtain initial probabilities. Following Bar-Haim *et al.* (2005), Adler and Elhadad empirically compare word and segment tokenization levels and conclude that segment-level tagging outperforms word-level tagging, which supports our own findings. It is important to note that, as in the case of Segal's reported results (cf. Section 6.2), all of the above results are incomparable to our results (and in most cases, to each other). This is mainly due to differences in choice of tagset, in evaluation methodology and the test set used. Our experience with reevaluating Segal's tagger, as well as the results reported in Adler and Elhadad (2006), show that these differences may have a considerable effect on accuracy. However, over and above accuracy measures, Adler and Elhadad's unsupervised method complements our present work, which augments a manually tagged corpus with an untagged corpus only for smoothing the lexical model. We believe that further research may profitably use the two models within a superior unified framework.

Arabic morphological disambiguation, segmentation, and tagging:

Lee *et al.* (2003) describe a word segmentation system for Arabic that uses a Markov language model over word segments. They start with a seed segmenter that employs 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 very large unsegmented corpus. The reported accuracy of this system is 97.1% (per word).

Diab *et al.* (2004) use Support Vector Machines (SVMs) for the tasks of word segmentation (which is done by classification at the letter level) and POS tagging and Base-Phrase Chunking. Segmentation and POS tagging are done in subsequent steps. For segmentation, they report precision of 99.09% and recall of 99.15%, when measuring *word segments* that were correctly identified. For tagging, Diab *et al.* report accuracy of 95.5%, with a tagset of 24 POS tags. Tagging was applied to segmented words, using the 'gold' segmentation from the annotated corpus.

The tagging architectures that were independently proposed in Habash and Rambow (2005) for Arabic and in Bar-Haim *et al.* (2005) for Hebrew, coincide in two major aspects, which contrasts with previous work on Arabic: first, the use of a morphological analyzer for obtaining the possible analyses, and second, the simultaeous segmentation and POS tagging. Habash and Rambow perform morphological disambiguation by training SVM classifiers separately for each morphological feature, and combining their output via a voting scheme. POS tagging and segmentation are derived from the morphological analysis. Projecting their output to the same tagset used by Diab *et al.* results in tagging accuracy of 97.6% on ATB part 1 (ATB1, also used by Diab *et al.*) and 95.7% on ATB part 2 (ATB2). Like Diab *et al.*, Habash and Rambow also assume gold segmentation. They report segmentation precision and recall of 98.9% and 99.3%, respectively, measured on ATB1. Rogati *et al.* (2003) investigate unsupervised stemming of Arabic using a parallel corpus. The accuracy they achieved is considerably lower than what is achieved with supervised methods.

Other related work

A related problem that has received much attention is the segmentation of a sentence into words in Asian languages such as Chinese and Japanese, where words are not delimited in the text. The approach presented in Nakagawa (2004) makes an interesting comparison to ours. Nakagawa's work builds on two former methods: first, HMM-based joint segmentation and POS tagging, as in our approach. This method, according to Nakagawa, was previously implemented in Japanese segmentation systems, achieving high accuracy in a low computational cost. Second, Nakagawa uses a character-level tagging method (Xue 2003), which performs better for unknown words. These two methods are combined into a single probabilistic model. It seems that our approach for unknown words, which is based on generating the possible segmentations for unknown words according to simple affixation rules, is more suitable for Semitic languages, since the number of segmentations for unknown words is quite small. More importantly, the linguistically motivated tokenization levels we consider, words versus segments, are very different from the word and character levels considered by Nakagawa.

A considerable amount of work has been published on unsupervised or minimally supervised methods for word segmentation in non-Semitic languages, including Goldsmith (2001),Yarowsky and Wicentowski (2000), Schone and Jurafsky (2000), and many others. We do not discuss this work in detail here, since the experience with the processing of Semitic languages has shown that using tagged corpora and morphological analyzers results in a much higher segmentation accuracy, rather similarly to European languages (Habash and Rambow 2005).

7.2 Conclusion

Developing a word segmenter and POS tagger for Hebrew with less than 30,000 annotated words for training is a challenging task, especially given the morphological complexity and high degree of ambiguity in Semitic texts. The difficulty of this task is demonstrated by the low accuracy of a baseline that selects the most frequent tag, relative to the results of the same baseline on tagging English.

In this paper, we started out from two straightforward architectures for segmentation and POS tagging of Hebrew, based on word or word-segment tokenization, and implemented them using standard HMMs. Experimenting with these two models has led us to develop a third architecture (the $\mathrm{WS}+h$ model), which overcomes some of the problems for the two simpler tokenization models. The $\mathrm{WS}+h$ architecture extends the segment-level tokenization by using multi segment nonterminals where it was found to be valuable. In this way, the number of nonterminal types (tag classes) found in the corpus for this model is forty-six, which is much closer to the segment-level model (37 types) than to the word-level model (185 types). We developed a smoothing method that exploits an existing morphological analyzer and

Table 8. *Experiments with gold segmentation and larger training corpus*

	Accuracy per word (%)		$F_{\beta=1}$ per word segment (%)	
System	Tagging	Segmentation	Tagging	Segmentation
WS with gold segmentation	92.64 (0.8)	100.00 (0.0)	94.33 (0.7)	100.00 (0.0)
WS + *h* with 4,500 sentences	90.81 (0.8)	97.21 (0.3)	92.60 (0.6)	97.74 (0.2)

a large untagged corpus for significantly improving the estimates of lexical model probabilities.

The proposed general architecture that we have worked in allows a flexible implementation of the disambiguator, independent of both tagset and language, as long as analyses are represented as a sequence of tagged segments. This allows a quick adaptation of the tagger to other Semitic languages, in particular Arabic. The language-independent tagger, named *MorphTagger*, can be obtained from the Knowledge Center for Processing Hebrew.[15] Among the few other tools available for POS tagging and morphological disambiguation in Hebrew, only Segal's system (Segal's 2000) is freely available. In Section 6.2, we conducted an extensive evaluation showing that, under the same experimental settings, our best architecture achieves an improvement of 1.8% in segmentation accuracy and 4.4% in tagging accuracy over the results of Segal's system.

One of the main sources for tagging errors in our model is the coverage of the morphological analyzer. The analyzer misses the correct analysis in 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 overgeneration as much as possible), is crucial for improving the tagging results.

In a setting comparable to Diab *et al.* (2004) and Habash and Rambow (2005) (including a tagset of a similar size), in which the correct segmentation is given, our tagger achieves (for Hebrew) accuracy per *word segment* of 94.3% (Table 8). This result is close to the result reported by Diab *et al.* and by Habash and Rambow on ATB2, although our result was achieved using a much smaller annotated corpus, and despite the much better coverage of the morphological analyzer used by Habash and Rambow (99.4%).

After this work was completed, a new version of the Hebrew treebank, now containing approximately 4,800 sentences, was released. Preliminary experiments with our best model on this corpus (see Table 8), using 4,500 sentences for training, show a further improvement of 1.22% in tagging accuracy (from 89.59% to 90.81%) and an improvement of 0.16% in segmentation accuracy (from 97.05% to 97.21%). We believe that the additional annotated data will allow us to refine our model, both in terms of accuracy and in terms of coverage, by expanding the tagset with

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

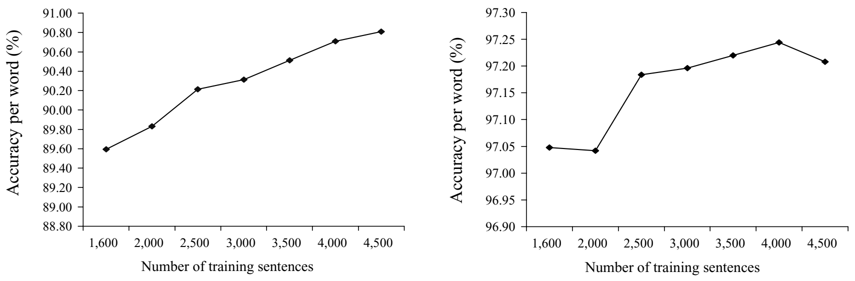

Fig. 1. Learning curves for tagging (left) and segmentation (right).

additional morphosyntactic features such as gender and number, which are prevalent in Hebrew and other Semitic languages. The learning curve in Figure 1 shows the effect of adding more training data on tagging and segmentation accuracy. The curve suggests that there is still room for improvement in tagging accuracy using the current model if the training corpus is further enlarged.

Acknowledgments

We thank Gilad Ben-Avi, Ido Dagan, and Alon Itai for their insightful remarks on major aspects of this work. We also thank Andreas Stolcke for his devoted technical assistance with SRILM. The financial and computational support of the Knowledge Center for Processing Hebrew is gratefully acknowledged. Support by the Institute for Logic, Language and computation (University of Amsterdam), the Computer Science Department (Technion), and the Netherlands Organization for Scientific Research facilitated the cooperation in this research. The first author thanks the Technion for partially funding his part of the research. The third author is grateful for the support of the *Netherlands Institute of Advanced Studies*, where part of this work was carried out.

Appendix A
Transliteration

Table A1 shows the Hebrew/Latin transliteration used in this paper and in our tagger implementation, following the transliteration used in the treebank. It is based on the ISO standard (ISO 1999), with the non-letter symbols @, &, and $ replaced by letters. This is merely due to a technical reason: some software we used required the input to be in English letters only.

Appendix B
Tagset

The tagset used in this work is shown in Table A2. The transcription of symbols is given in Table A3.

Table A1. *Hebrew to Latin transliteration table*

Hebrew	א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ/ך
This paper	a	b	g	d	h	w	z	x	j	i	k
ISO	A	B	G	D	H	W	Z	X	@	I	K
Hebrew	ל	מ/ם	נ/ן	ס	ע	פ/ף	צ/ץ	ק	ר	ש	ת
This paper	l	m	n	s	e	p	c	q	r	f	t
ISO	L	M	N	S	&	P	C	Q	R	$	T

Table A2. *The Hebrew POS tagset*

1.	AGR	Agreement particle
2.	AT	Accusative marker
3.	CC	Coordinating conjunction
4.	CD	Numeral/numeral determiner
5.	COM	Complementizer
6.	DT	Determiner/question word
7.	IN	Preposition
8.	JJ	Adjective
9.	JJT	Construct state adjective
10.	H	Definiteness marker
11.	HAM	Yes/No question word
12.	NN	Noun
13.	NN-H	Noun, definite|definite-genitive
14.	NNP	Proper noun
15.	NNT	Construct state noun
16.	POS	Possessive item
17.	PRP	Personal pronoun
18.	QW	Question/WH word
19.	RB	Adverb or modifier
20.	RBR	Adverb, comparative
21.	REL	Relativizer
22.	VB	Verb (or auxiliary verb), finite
23.	VB-M	Verb, infinite
24.	ZVL	Garbage

Table A3. *Transcription of symbols*[*]

o	%	CLN	:	DOT	.	EXCL	!	ELPS	...
u	”	LRB	(	DASH	–	QM	?		
CM	,	QUOT	”	RRB	)	SCLN	;		

[*]‘o’ and ‘u’ represent transliteraton of symbols that appear as part of tokens. The other symbols are used both as the transcription and the POS tag assigned to symbols that appear in text as separate tokens.

Table A4. *Most frequent word segments*

Word segment	Meaning	Frequency	%
h	‘the’	5,040	10.4
b	‘in’	2,315	4.8
w	‘and’	2,273	4.7
l	‘to’	1,863	3.9
f	‘that’	1,343	2.8

Table A5. *Segments per word*

# Word segments	Count	%
1	19,248	63.82
2	9,479	31.43
3	1,382	4.58
4	53	0.18
Total	30,162	100.00

Appendix C
Figures on the annotated Hebrew corpus

In this section, we present some statistics computed from our tagged training corpus; some of them also involve a morphological analyzer. These statistics characterize some phenomena in Hebrew morphology that are relevant for Hebrew tagging and segmentation.

Most frequent word segments: The most frequent word segments in the corpus are listed in Table A4. As expected, the most common segments are the same function words that are found at the top of word frequency lists in English. The difference is that in Hebrew these word segments appear as prefixes in other words, rather than as separate words.

Number of segments per word: The histogram in Table A5 displays the distribution of words in the corpus according to the number of segments (prefixes, suffixes, and the stem). Punctuation marks were ignored. The results show that, in practice, most Hebrew words are not very complex: more than 95% of them contain only one or two segments, and only 0.18% of the words contain more than three segments. Although not found in our tagged corpus, five-segment words can be found in Hebrew texts (e.g. *w-f-b-h-bit*, ‘and that in the house’), and, in principle, even six-segment words are possible (e.g. *w-f-kf-b-h-bit*, ‘and that when in the house’).

Affixes: In the tagged corpus, there are 10,526 words with prefixes and 1,653 words with suffixes. The distribution of the suffixes among their three categories (verb, preposition, and noun) is given in Table A6. Recall that in the annotation scheme used in this paper, possessive suffixes for nouns are considered part of the stem, rather than separate word segments.

Table A6. *The frequency of pronominal and possessive suffixes in the tagged corpus*

Part of speech (POS)	Count	%	With suffix (POS)	% (of POS)	% (of all)
Verb	4,811	16.0	30	0.6	0.1
Preposition	7,004	23.2	617	8.8	2.0
Noun	11,004	36.4	1006	9.1	3.4

Table A7. *Possible analyses per word*

No. of analyses per word	No. of word tokens	%
1	145,442	43.07
2	75,912	22.48
3	55,489	16.43
4	27,042	8.01
5	18,464	5.47
6	6,527	1.93
7	3,319	0.98
8	2,673	0.79
9	1,364	0.40
10	477	0.14
11	350	0.10
$\geq$12	592	0.18

Ambiguity: The histograms in Tables A7 and A8 illustrate the dimension of segmentation and tagging ambiguity in the untagged Hebrew corpus, with respect to the reduced tagset we use for evaluation. These results were obtained from the untagged corpus (337,651 words) by running Segal's (2000) morphological analyzer (described in Section 5), augmented with the analyses found in the tagged training corpus. The average number of analyses per word was found to be 2.3, and approximately 57% of the word tokens were ambiguous. Note that the actual morphological ambiguity for Hebrew is higher, due to the simplifications made in our scheme, as described above.

Table A8. *Possible segmentations per word*

No. of segmentations per word	No. of word tokens	%
1	268,261	79.45
2	56,074	16.61
3	12,781	3.79
4	532	0.16
5	3	0.00

References

Adler, M. and Elhadad, M. 2006. An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, pp. 665–672. East Stroudsburg, PA: Association for Computational Linguistics.

Bar-Haim, R., Sima'an, K. and Winter, Y. 2005. Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Ann Arbor, pp. 39–46, MI. East Stroudsburg, PA: Association for Computational Linguistics.

Baum, L. 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.*

Brants, T. 2000. TnT: A statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, WA.

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

Buckwalter, T. 2002. *Buckwalter Arabic Morphological Analyzer Version 1.0.* Linguistic Data Consortium (LDC). LDC Catalog No.: LDC2002L49, ISBN:1-58563-257-0.

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

Charniak, E., Hendrickson, C., Jacobson, N. and Perkowitz, M. 1993. Equations for part-of-speech tagging. In *National Conference on Artificial Intelligence*, pp. 784–789.

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

Cutting, D., Kupiec, J., Pedersen, J. and Sibun, P. 1992. A practical part-of-speech tagger. In *Proceedings of the third conference on Applied natural language processing*, Association for Computational Linguistics pp. 133–140.

Danon, G. 2001. Syntactic definiteness in the grammar of Modern Hebrew. *Linguistics* **39**: 1071–1116.

Daya, E., Roth, D. and Wintner, S. 2004. Learning Hebrew roots: machine learning with linguistic constraints. In *Proceedings of EMNLP'04*, Barcelona, Spain, pp. 357–364.

Dermatas, E. and Kokkinakis, G. 1995. Automatic stochastic tagging of natural language texts. *Computational Linguistics* **21**(2): 137–163.

DeRose, S. J. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics* **14**(1): 31–39.

Diab, M., Hacioglu, K. and Jurafsky, D. 2006. Automatic Tagging of arabic text: from raw text to base phrase chunks. In D. M. S. Dumais and S. Roukos (eds), *HLT-NAACL 2004: Short Papers*, Boston, MA, pp. 149–152. East Stroudsburg, PA: Association for Computational Linguistics.

Elworthy, D. 1994. Does Baum-Welch re-estimation help taggers? In *Proceedings of the fourth conference on Applied natural language processing*, Morgan Kaufmann Publishers Inc. pp. 53–58.

Glinert, L. 1989. *The Grammar of Modern Hebrew.* Cambridge, England: Cambridge University Press.

Goldsmith, J. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics* **27**(2): 153–198.

Good, I. J. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika* **40**: 237–264.

Habash, N. and Rambow, O. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting* *of the Association for Computational Linguistics (ACL'05)*, pp. 573–580, Ann Arbor, MI. Association for Computational Linguistics.

Hakkani-Tür, D., Oflazer, K. and Tür, G. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*.

ISO. 1999. Information and documentation – conversion of Hebrew characters into Latin characters – part 3: Phonemic conversion, ISO/FDIS 259-3: (E).

Katz, S. M. 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.

Lee, Y. S., Papineni, K., Roukos, S., Emam, O. and Hassan, H. 2003. Language model based arabic word segmentation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Morristown, NJ, USA, pp. 399–406. East Stroudsburg, PA: Association for Computational Linguistics.

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

Levinger, M. 1992. *Morphological Disambiguation in Hebrew.* Master's thesis, Computer Science Department, Technion, Haifa, Israel. In Hebrew.

Maamouri, M., Bies, A., Buckwalter, T. and Mekki, W. 2004. The Penn Arabic Treebank: building a large-scale annotated Arabic corpus. In *NEMLAR International Conference on Arabic Language Resources and Tools*, Cairo.

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics* **20**(2): 155–171.

Nakagawa, T. 2004. Chinese and japanese word segmentation using word-level and character-level information. In *Proceedings of Coling 2004*, Geneva, Switzerland, pp. 466–472.

Nigam, K., Mccallum, A. K., Thrun, S. and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* **39**(2–3): 103–134.

Rogati, M., McCarley, S. and Yang, Y. 2003. Unsupervised learning of Arabic stemming using a parallel corpus. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, Sapporo, Japan, pp. 391–398.

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

Segal, E., 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

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

Stolcke, A. 2002. SRILM — an extensible language modeling toolkit. In *ICSLP*, Denver, CO, pp. 901–904.

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transaction of Information Theory* **IT-13**(2): 260–269.

Watson, J. C. E. 2002. *The Phonology and Morphology of Arabic*. Oxford University Press, Oxford.

Weischedel, R., Schwartz, R., Palmucci, J., Meteer, M. and Ramshaw, L. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics* **19**(2): 361–382.

Wintner, S. 2000. Definiteness in the Hebrew noun phrase. *Journal of Linguistics* **36**: 319–363.

Xue, N. 2003. Chinese word segmentation as character tagging. *International Journal of Computational Linguistics and Chinese* **8**(1): 29–48.

Yarowsky, D. and Wicentowski, R. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL-2000*, Hong Kong, pp. 207–216.