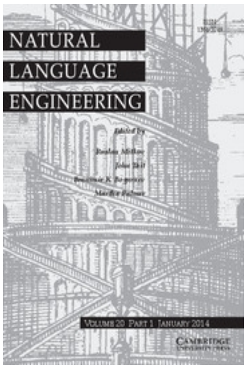

Morphological disambiguation of Hebrew: a case study in classifier combination

GENNADI LEMBERSKY, DANNY SHACHAM and SHULY WINTNER

Natural Language Engineering / Volume 20 / Issue 01 / January 2014, pp 69 - 97
DOI: 10.1017/S1351324912000216, Published online: 26 July 2012

Natural Language Engineering **20** (1): 69–97. © Cambridge University Press 2012
doi:10.1017/S1351324912000216

Morphological disambiguation of Hebrew: a case study in classifier combination

GENNADI LEMBERSKY, DANNY SHACHAM
and SHULY WINTNER
Department of Computer Science, University of Haifa, Haifa, Israel
e-mails: glembers@campus.haifa.ac.il,
danny@shach.am,
shuly@cs.haifa.ac.il

(*Received 30 November 2011; revised 27 June 2012; accepted 1 July 2012; first published online 26 July 2012*)

Abstract

Morphological analysis and disambiguation are crucial stages in a variety of natural language processing applications, especially when languages with complex morphology are concerned. We present a system which disambiguates the output of a morphological analyzer for Hebrew. It consists of several simple classifiers and a module that combines them under the constraints imposed by the analyzer. We explore several approaches to classifier combination, as well as a back-off mechanism that relies on a large unannotated corpus. Our best result, around 83 percent accuracy, compares favorably with the state of the art on this task.

1 Introduction

Morphological analysis and disambiguation are crucial pre-processing steps for a variety of natural language processing applications, from search and information extraction to machine translation. For languages with complex morphology, these are non-trivial processes. This paper presents a morphological disambiguation module for Hebrew which uses a sophisticated combination of classifiers to rank the analyses produced by a morphological analyzer.[1] This work has a twofold contribution: First, our system achieves approximately 83 percent accuracy on the full disambiguation task, paving the way to more accurate processing of Hebrew texts. More generally, we reinforce the utility of combining the predictions of simple classifiers under linguistically motivated constraints; the insight gained from these experiments will

[1] This paper is a completely revised and significantly extended version of Shacham and Wintner's paper (Shacham, D., and Wintner, W. June 2007. Morphological disambiguation of Hebrew: a case study in classifier combination. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning (EMNLP-CoNLL 2007)*, Prague, Czech Republic. Stroudsburg, PA: Association for Computational Linguistics). In particular, the disambiguation module was completely redesigned and re-implemented; the annotated corpus we use is new; we added a few classifiers to further improve the accuracy of the module; the baseline is better defined; we report on several new experiments; and the entire error analysis section is new.

be useful for other applications of machine learning to complex (morphological and other) problems.

In the remainder of this section we discuss the complexity of Hebrew morphology, the challenge of morphological disambiguation and related work. We describe our methodology in Section 2: we use basic, naïve classifiers to predict some components of the analysis, and then combine them to predict a consistent result (Section 3). We analyze the errors of the system in Section 4 and conclude with suggestions for future work.

1.1 Linguistic Background

Hebrew morphology is rich and complex.[2] The major word formation machinery is root-and-pattern, and inflectional morphology is highly productive and consists of prefixes, suffixes and circumfixes. Nouns, adjectives and numerals inflect for number (singular, plural and, in rare cases, also dual) and gender (masculine or feminine). In addition, all these three types of nominals have two phonologically- and morphologically distinct forms, known as the *absolute* and *construct* states. In the standard orthography, approximately half of the nominals appear to have identical forms in both states, a fact which substantially increases the ambiguity. In addition, nominals take possessive pronominal suffixes, which inflect for number, gender and person: *spr+h→sprh* 'her book', *spr+km→sprkm* 'your book' *etc*. As expected, these processes involve certain morphological alternations, as in *mlkh+h→mlkth* 'her queen', *mlkh+km→mlktkm* 'your queen'.

Verbs inflect for number, gender and person (first, second and third) and also for a combination of tense and aspect, which is traditionally analyzed as having the values past, present, future, imperative and infinite. Verbs can also take pronominal suffixes, which in this case are interpreted as direct objects, but such constructions are rare in contemporary Hebrew of the registers we are interested in. In some frozen cases, verbs can also take nominative pronominal suffixes, especially in the infinitive: *bw'+h→bw'h* 'her coming', *$wb+km→$wbkm* 'your return'. These can arguably be analyzed as deverbal nouns with genitive suffixes. A peculiarity of the Hebrew verbs is that the participle form can not only be used as a present tense but also as a noun or an adjective.

These matters are complicated further due to two sources: First, the standard Hebrew orthography leaves most of the vowels unspecified. It does not explicate *[a]* and *[e]*, does not distinguish between *[o]* and *[u]* and leaves many of the *[i]* vowels unspecified. Furthermore, the single letter *w* is used both for the vowels *[o]* and *[u]*, and for the consonant *[v]*, whereas *i* is similarly used both for the vowel *[i]* and for the consonant *[y]*. On top of that, the script dictates that many particles, including four of the most frequent prepositions (*b* 'in', *k* 'as', *l* 'to' and *m* 'from'), the definite article *h* 'the', the coordinating conjunction *w* 'and' and some subordinating conjunctions

[2] To facilitate readability, we use a straightforward transliteration of Hebrew using ASCII characters, where the characters (in Hebrew alphabetic order) are: abgdhwzxviklmnsypcqr$t.

Table 1. *The analyses of the form* $*bth*. *Each analysis specifies a unique ID for the lemma, the lemma itself, its part-of-speech, number, gender, person, tense, nominal status, definiteness, prefixes and suffixes*

#	ID+lemma		POS	Num	Gen	Per	Tns	Stat	Def	Pref	Suf
1.	17280	$*bt*	noun	sing	fem	–	–	abs	no		h
2.	1379	*bt*	noun	sing	fem	–	–	abs	no	$	h
3.	19130	*bth*	noun	sing	fem	–	–	abs	no	$	
4.	19804	*th*	noun	sing	masc	–	–	abs	yes	$+*b*+*h*	
5.	19804	*th*	noun	sing	masc	–	–	abs	no	$+*b*	
6.	19804	*th*	noun	sing	masc	–	–	cons	no	$+*b*	
7.	1541	$*bh*	verb	sing	fem	3	past	–	–		
8.	9430	$*bt*	verb	sing	fem	3	past	–	–		

(such as $ ‘that’ and *k*$ ‘when’), all attach to the words that immediately follow them.

There is only one definite article in Hebrew, *h [ha-]*, which does not inflect and is attached (pre-nominally) to words. When a definite nominal is prefixed by one of the prepositions *b*, *k* or *l*, the definite article *h* is assimilated with the preposition and the resulting form becomes ambiguous as to whether or not it is definite. Thus, *bth* can be read either as *b*+*th* ‘in tea’ or as *b*+*h*+*th* ‘in the tea’. Subsequently, a form such as $*bth* can be read as a lexeme (the verb ‘capture’, third person singular feminine past), as $+*bth* ‘that+field’, $+*b*+*th* ‘that+in+tea’, $*bt*+*h* ‘her sitting’ or even as $+*bt*+*h* ‘that her daughter’.

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

1.2 Hebrew morphological analysis

Our departure point in this work is the MILA morphological analyzer Itai and Wintner (2008), a wide coverage, linguistically motivated morphological analyzer of Hebrew, which is based on a finite-state morphological grammar (Yona and Wintner, 2008) but is reimplemented in Java (Wintner, 2008). The output that the analyzer produces for the form $*bth* is illustrated in Table 1. In general, it includes the part-of-speech (POS) as well as the sub-category, where applicable, along with several POS-dependent features such as number, gender, tense, nominal state, definiteness *etc*. Table 2 shows the relevant attributes for each major POS category.

Table 2. *Part-of-speech (POS) categories and their possible attributes*

POS	Attributes
Adjective	Gender, Number, Status, Definiteness, suffix
Adverb	Gender, Number, Person
Conjunction	
Copula	Tense, Gender, Number, Person, suffix
Existential	Tense, Gender, Number, Person, Definiteness
Foreign	
Interjection	
Interrogative	Gender, Number, Person
Modal	Tense, Gender, Number, Person
Multi-word expression(MWE)	Gender, Number, Definiteness
Negation	Definiteness
Noun	Gender, Number, Status, Definiteness, suffix
Number expression	
Numeral	Gender, Number, Status, Definiteness, suffix
Participle	Gender, Number, Status, Person, Definiteness, Type
Preposition	Gender, Number, Person, suffix
Pronoun	Gender, Number, Person, Definiteness, Type
Proper name	Gender, Number, Definiteness
Punctuation	
Quantifier	Status, Definiteness, suffix
Title	Gender, Number, Definiteness
Unknown	Definiteness
Verb	Tense, Gender, Number, Person, suffix
WPrefix	Gender, Number, Definiteness

1.3 Hebrew morphological disambiguation

Morphological analyzers produce, for each word, all its valid analyses, independently of the context in which the word occurs. Morphological *disambiguation* is the task of ranking the grammatically valid analyses of each word, taking into account the context and producing the most likely analysis. This task is reminiscent of POS tagging, but is more challenging for languages with rich morphology (see Section 1.4).

The first stochastic approaches to Hebrew morphology could not rely on the existence of an annotated corpus. Hence, the primary task of such works was to approximate the *morpho-lexical probability* of each word in a given text (Levinger, Ornan and Itai 1995). Given a text T with n words $w_1, \ldots, w_n$, for each morphologically ambiguous word w_i whose analyses are $A_1, \ldots, A_k$, there is one analysis, $A_r \in \{A_1, \ldots A_k\}$, which is the correct analysis in context. The morpho-lexical probability of an analysis A_i for some word w is the estimate of the conditional probability $P(A_i|w)$ from a given corpus:

$$P(A_i \mid w) = \frac{\text{Number of times } A_i \text{ is the correct analysis of } w}{\text{Number of occurrences of } w}$$

This probability is independent of the context, which implies that an algorithm for computing it will necessarily select the same analysis for a given word in all contexts.

Approximation of the morpho-lexical probabilities from an untagged corpus is the main task of Levinger *et al.* (1995), who use the Avgad morphological analyzer (Bentur, Angel and Segev 1992). The main idea of their approach is to count not only the ambiguous word itself but also all the members of the set of its *similar words*, the underlying assumption being that similar words have similar frequencies but may be unambiguous. Similarity is defined as sharing the same lexical entry, but differing in at least one morphological feature. The rules for defining the set of similar words for each word *w* are pre-defined and are manually constructed. For example, an indefinite noun is similar to its definite counterpart; nouns with an attached pronominal clitic are similar to (the same) nouns with other pronominal clitics *etc*. The reported results show very good performance in many of the cases, reasonable performance in others and some cases of very poor decisions. Note, however, that morphological disambiguation in this work is completely independent of context.

A similar approach is used by Carmel and Maarek (1999). To overcome the problem of data sparseness, this system makes decisions based on the frequencies of *morphological patterns* associated with the analyses of input words, rather than the analyses themselves. The patterns consist of parts of speech and other morphological information, but essentially not the lexeme. Here, too, contextual information is not taken into account, and the system only prunes relatively unlikely candidates independent of where they occur using some pre-defined threshold. For evaluation, a set of 16,000 words were manually annotated. Accuracy is defined as the number of words for which the output of the system includes the correct analysis. At a threshold of 0, accuracy is 98 percent (very unlikely analyses are filtered out always, so 100 percent cannot be achieved) and the proportion of words with a single analysis is 62 percent. At a threshold of 0.5, accuracy is 86 percent (74 percent for ambiguous words) and 100 percent of the words are assigned a single analysis.

The first work which uses stochastic contextual information for morphological disambiguation in Hebrew is Segal (1999): Texts are analyzed using the morphological analyzer of Segal (1997); then, each word in a text is assigned its most likely analysis, defined by probabilities computed from a small tagged corpus. In the next phase the system corrects its own decisions by using short context (one word to the left and one to the right of the target word). The corrections are also automatically learned from the tagged corpus (using transformation-based learning). In the last phase, the analysis is corrected by the results of a syntactic analysis of the sentence. The reported results are excellent: 96.2 percent accuracy. More reliable tests, however, reveal accuracy of 85.5 percent only (Lembersky 2003: 85). Furthermore, the performance of the program is unacceptable (the reported running time on 'two papers' is thirty minutes).

The early works described above were all hampered by the lack of annotated corpora for Hebrew. More recently, however, such corpora have become available through a treebank project (Sima'an *et al.*, 2001) and the Knowledge Center for Processing Hebrew (Itai and Wintner, 2008), facilitating the application of advanced machine-learning techniques to the problem of morphological disambiguation.

Bar-Haim, Sima'an and Winter (2005, 2008) describe a segmenter and POS-tagger for Hebrew based on Hidden Markov Models (HMMs). This disambiguation model for segmentation and part-of-speech is based on word segments (morphemes) rather than words. Word segments are referred to as identifying the prefixes, the stem and suffixes of the word. This work starts out from a morphological analyzer and a very small morphologically annotated corpus. A model whose terminal symbols are word segments is shown to be advantageous over a word-level model for the task of POS tagging. Hence, a morpheme-level model is proposed, where words are first divided into separate morpheme segments: prefixes, stem and suffix. Then a proper POS tag is assigned to each of these morphemes. A way to extend the morpheme sequence back to a full analysis is also shown. The definiteness morpheme is treated as a possible feature of morpheme terminals, and the definiteness marker's POS tag is prefixed to the POS tag of the stem. This way data sparseness is reduced. However, this method lacks some word-level knowledge that is required for segmentation, and hence a part of information that is usually found in Hebrew morphological analyses is ignored. The internal morphological structure of stems is not analyzed, and the POS tag assigned to stems includes no information about their root, pattern, inflectional features and suffixes. The accuracy of this segmenter is reported for both tagging (a tagset of twenty-one POS tags is used) and for segmentation, and is 90.51 percent for the former and 96.74 percent for the latter.

Unfortunately, this tagger/segmenter is not enough for full morphological disambiguation. Some features produced by the analyzer are not used by the tagger, but are needed for full disambiguation. Consider, for example, the word *$bth* (Table 1): Analyses 5 and 6 differ only in status. Such words are ambiguous in a way that is not solved by the segmenter, since it cannot distinguish between different lexical IDs or the different values of the status attribute. Consequently, the version of the tagger that was made available to us resulted in a much lower accuracy on the full disambiguation task (only 75.73 percent).

Adler and Elhadad (2006) present an unsupervised stochastic model, where the only resource used is a morphological analyzer (see Section 1.5). A morpheme-based model must learn both segmentation and tagging in parallel in an unsupervised manner, since the output sequence used for the learning and searching process is uncertain. Reported results on a large-scale corpus (6 M words) with fully unsupervised learning are 92.32 percent for POS tagging and 88.5 percent for full morphological disambiguation. Due to various technical difficulties, including inconsistent corpus versions, we were unable to reproduce these results on our corpus. Furthermore, our work addresses certain subtle distinctions between different morphological analyses that Adler and Elhadad (2006) ignore; in other words, their system retains some ambiguity that we resolve. These include the subtype of proper names (we distinguish between names of persons, locations, organizations *etc.*); the subtype of pronouns (personal, demonstrative, reflexive *etc.*); the function of participles (which can function as nouns, adjectives or verbs); and distinguishing between two analyses that only differ in their lemma. On the other hand, Adler and Elhadad (2006) provide analyses for out-of-vocabulary tokens, which our task does not handle (Section 2.1).

A supervised approach to morphological disambiguation of *Arabic* is presented by Habash and Rambow (2005). This work shows that the use of a morphological analyzer and learning classifiers for individual morphological features improve the results of both tokenizing and POS tagging for Arabic.[3] Habash and Rambow (2005) developed a supervised morphological disambiguator, based on a training corpus of two sets of 120 K words, which combines several classifiers. Every morphological feature is predicted separately and then combined into a full disambiguation result. The accuracy of the disambiguator is 94.8–96.2 percent (depending on the test corpus). One thing to note, though, is the high accuracy they report for the baseline of each classifier (96.6–99.9 percent, depending on the classifier). The baseline of the whole disambiguating process is 87.3–92.1 percent (depending on the test corpus). Although Habash and Rambow (2005) use a similar approach to the one presented in the present work, and the number of tags is similar to the number of tags in Hebrew, the differences between Hebrew and Arabic still induce significant differences in the results. Furthermore, our training corpus is far smaller than the ones available for Arabic.

1.4 The challenge of disambiguation

The most common and successful techniques for POS tagging in English use HMM (Marshall, 1983; Cutting *et al.*, 1992), Transformation-based learning (Brill, 1995) or more sophisticated learning algorithms (Roth and Zelenko, 1998). With a baseline of 90–91 percent accuracy, it is not very surprising that the best POS taggers for English achieve an accuracy of 96–97 percent (Manning and Schütze, 1999, Section 10). Crucially, the size of the tagset for English is very small: the Penn Treebank tagset has forty-five tags (including punctuation), and other proposed tagsets (e.g., the Brown Corpus one) are all smaller than one hundred tags.

Stochastic POS taggers for English take advantage of the high baseline and the relatively small tagset. They also benefit from the relatively fixed word order of English and from the fact that many of the frequent functional words (prepositions, determiners *etc.*) are unambiguous; such words can be used as *anchors* to disambiguate their neighbors. Morphological disambiguation of Hebrew is a much more complex endeavor due to the following factors:

Segmentation. A single token in Hebrew can actually be a sequence of more than one lexical item. Consider again the token *$bth* discussed above (Table 1). It can be segmented in the following ways, each inducing a completely different sequence of POS tags:

[3] This reinforces results for Czech and other languages reported by Hajič and Hladká (1998) and Hajič (2000), who first proposed this approach for disambiguation of morphologically complex languages.

\$*bth*	'captured'	VBD
\$*+bth*	'that+field'	IN+NN
\$*+b+th*	'that+in+tea'	IN+IN+NN
\$*+b+h+th*	'that+in+the+tea'	IN+IN+DT+NN
\$*bt+h*	'her sitting'	PRP\$+NN or PRP\$+VBG
\$*+bt+h*	'that her daughter'	IN+PRP\$+NN

In practice, most tokens in a running Hebrew text have only one possible segmentation. In particular, the length of a prefix sequence is rarely greater than 1.

Large tagset. In contrast to the limited tagset of standard POS tagging approaches in English, the number of potential tags in a language such as Hebrew (where the POS, morphological features and prefix and suffix particles are considered) is huge. The morphological analyzer that we use in this work produces twenty-seven different parts of speech, some with subcategories: five values for the number feature (including disjunctions of values), three for gender, three for person, eight for tense and three for nominal status. Possessive pronominal suffixes can have eighteen different values, and prefix particle sequences can theoretically have hundreds of different forms, although in practice only a few dozens are witnessed in a standard corpus. Clearly, not all the combinations of these values are possible, and many values are extremely rare (e.g., the value 'dual' of the feature 'number'). Still, the number of tags observed in large corpora is in the thousands. Thus, Adler (2007) determines the number of tags in the Hebrew corpus he used to be 3,600.

Ambiguity. Due to the rich morphology and the problems of the orthography, Hebrew is highly ambiguous. In a particular corpus of 40,000 word tokens, the average number of analyses per word token was found to be 2.1, while 55 percent of the tokens were ambiguous (Levinger *et al.* 1995: 385). Our analyzer outputs on average approximately three analyses per word token, and 63 percent of the tokens are ambiguous. Excluding punctuation, the average number of analyses per token is 3.4, and some words have as many as thirty-seven analyses. Furthermore, in many cases two or more alternative analyses share the same part-of-speech, and in some cases two or more analyses are completely identical, except for their lexeme. This is the case of \$*bth* (see Table 1), whose verb analysis is in fact ambiguous with respect to the lexeme (which can be the third person feminine singular past tense of 'capture, take as captive' or of 'go on strike'), and whose noun analysis is ambiguous too ('her sitting' versus 'her Saturday'). Such phenomena make morphological disambiguation of Hebrew, in some extreme cases, closer to the problem of word sense disambiguation than to standard POS tagging.

Anchors. Anchors, which are often function words, are almost always morphologically ambiguous in Hebrew. For example, the most frequent token in Hebrew is *at* which is almost always a preposition (the accusative marker), but sometimes a personal pronoun (second person feminine singular) and, very rarely, a noun ('spade'). Similarly, the frequent genitive preposition \$l 'of', in some of its inflections, is highly ambiguous. Many of the function words which help boost the performance of English POS tagging are actually prefix particles which add to the ambiguity in Hebrew (e.g., the definite article *h* 'the' or the coordinating conjunction *w* 'and').

Word order. Word order in Hebrew is relatively free, and in any case much freer than in English.

1.5 Direct classification

While the number of possible analyses is potentially huge, several constraints hold among the values of different components of the analysis. For example, the *tense* feature is only relevant if the POS is verb, whereas *definiteness* is relevant only for nominals (nouns, adjectives, numerals and pronouns). As another example, the number of prefix particle sequences is theoretically large, but in practice no more than three or four prefixes are observed, with very strict (syntactically motivated) constraints on their order. Therefore, it may be possible to define a relatively small number of complex POS tags which uniquely cover the set of valid analyses.

Such a strategy is proposed by Adler and Elhadad (2006); it is based on the following two observations: first, a valid sequence of prefix particles in Hebrew can be at most of length 6 (characters),[4] and when its length is known, its segmentation is uniquely determined; second, if it is known that the word has a pronominal suffix, then this suffix can almost always be uniquely determined from the surface form of the word. Thus, a POS tag for Hebrew should consist of three components: an indication of the number of characters (0 to 6), which comprise the prefix sequence; a tag for the lexeme; and an indication of whether or not a pronominal suffix is present. The tag of the lexeme must capture the main and secondary part-of-speech, and a collection of features such as number, gender, person, tense, status *etc.* As an example of this strategy, consider again the case of $bth (see Table 1), whose analyses can be tagged as follows:

Form	Meaning	Prefix	Tag	Suffix
$*bth*	'captured'	0	V+Sg+F+3rd+Past	–
$+*bth*	'that+field'	1	N+Sg+F+indef	–
$+*b*+*th*	'that+in+tea'	2	N+Sg+M+indef	–
$+*b*+*h*+*th*	'that+in+the+tea'	2	N+Sg+F+det	–
$*bt*+*h*	'her sitting'	0	N+Sg+F+indef	+
$+*bt*+*h*	'that her daughter'	1	N+Sg+F+indef	+

This encoding can then be used with standard POS tagging techniques to uniquely disambiguate a morphologically analyzed corpus. This work was further extended by Adler *et al.* (2008a), who focus on obtaining the correct analysis for out-of-vocabulary items, a task that we do not address in the present work. However, the direct classification approach has two shortcomings. First, the number of tags

[4] While six-character prefixes are theoretically possible, evidence from large corpora reveals that prefix sequences are very rarely longer than three letters.

remains very high: with seven possible values for the location of the prefix sequence (0–6), approximately fifty tags for the lexeme and two values for the suffix, the tag set has 700 members. For languages with more complex morphology than Hebrew, in particular agglutinative languages, this number can reach thousands of categories, thereby reducing the potential of standard classification methods to perform accurately. Second, this encoding does not suffice to uniquely disambiguate the corpus in the face of analyses which differ only in the lexeme, or in the subtype of POS categories, as we have seen above (Section 1.3). We therefore opt for the more complex, yet more promising technique of combining classifiers.

1.6 Classifier combination

Making natural language decisions often involves assigning values to sets of variables where complex and expressive dependencies can influence, or even dictate, what assignments are possible. This is clearly the case for the problem of morphological disambiguation: one can learn relatively accurate classifiers for various components of the analysis, such as the main POS, or the correct segmentation of the word form, or whether or not the form is definite. Obviously, such decisions are highly interdependent, and in particular, the combination of the predictions of the classifiers must respect the constraints imposed by the morphological analyzer, namely that the resulting analysis is indeed one of the analyses produced by the analyzer. As an example, consider again the form *$bth*. A POS classifier can predict that the main POS tag is VBD, whereas a segmentation classifier can predict that the initial $ is a prefix; the two predictions are inconsistent, given the valid analyses produced by the analyzer. Of course, if the classifiers not only predict a single candidate but rather rank all possible candidates, a more sophisticated method of combining their results can yield predictions which are consistent with the constraints imposed by the analyzer.

Efficient solutions have been given to problems of this kind under some restrictions. One class of solutions decouples the process of learning estimators to variables' values from the inference process that is applied later, to derive a global assignment to the set of variables of interest. A second class of solutions incorporates the dependencies among the variables into the learning process, and directly induces estimators that yield a (good) global assignment. HMMs, conditional models (McCallum, Freitag and Pereira 2000; Punyakanok and Roth 2001) and a large number of dynamic programming schemes used in the context of sequential predictions fall into the first category. Conditional Random Fields (Lafferty, McCallum and Pereira 2001) and other discriminative learning methods with Markov assumptions are in the second category.

In this work we use a different solution. We decouple the multi-class classification task into several simple processes and then combine the results under the constraints imposed by a linguistically informed morphological analyzer. This approach was shown to be successful for identifying the roots of Semitic words (Daya, Roth and Wintner 2004, 2008) and for the Arabic morphological disambiguation (Habash and Rambow, 2005). Combining machine learning with limited linguistic knowledge was shown to produce state-of-the-art results on difficult morphological tasks.

2 Methodology

2.1 Task

Our task in this work is to induce a consistent ranking on the analyses produced by the morphological analyzer. Similar to Hajič (2000), Habash and Rambow (2005) and Daya *et al.* (2004, 2008), we approach the problem of morphological disambiguation as a complex classification task. We train a classifier for each of the attributes that can contribute to the disambiguation of the analyses produced by the analyzer (e.g., POS, tense, state). Each classifier predicts a small set of possible values and hence can be highly accurate. In particular, the basic classifiers do not suffer from problems of data sparseness. Of course, each simple classifier cannot fully disambiguate the output of the analyzer, but it does induce a ranking on the analyses. Then, we combine the outcomes of the simple classifiers to produce a consistent ranking which induces a linear order on the analyses.

Note that each classifier ranks only the options that are valid for the target word. If the analyses of a given word specify, for example, only two values for parts of speech: *noun* and *verb*, the POS classifier only ranks those two for this word and gives a 0 score to the other POS values. In this way, the classifiers induce a ranking on the analyses themselves. For example, ranking the two POS values of the analyses of Table 1 directly induces a ranking of those analyses, where analyses whose main part of speech is noun (analyses 1–6) are ranked the same, as are analyses with main POS verb (7–8).

Note also that the task does not involve induction of tags on all words in a text; if a word is unknown to the morphological analyzer, it is up to the analyzer to suggest an analysis for it (its current policy is to suggest proper noun analyses, by stripping potential prefixes from the word form). Our task is only to rank the output of the analyzer.

2.2 Corpora

For training and evaluation, we manually annotated a corpus of 3,989 sentences (72,062 tokens) taken from the daily newspaper *HaAretz*. The annotation consists of selecting the correct analysis produced by the analyzer, or an indication that no such analysis exists. The corpus was annotated by two annotators, and a trained lexicographer consolidated the two annotations and resolved any inconsistencies. Most of the corpus (3,325 sentences; 59,870 tokens) is used for training and the remaining part (664 sentences; 12,192 tokens, of which 2,734 do not occur in the training corpus) is for development.

Tokens for which the lexicographers determined that the correct analysis is not produced by the analyzer are disregarded during the experiments; such tokens reflect limitations of the analyzer, but do not affect the task of disambiguation. The corpus also includes very few tokens for which no analysis was produced, again reflecting

Table 3. *Statistics of training corpus*

Number of tokens	59,870
Number of types	18,418
Number of out-of-vocabulary tokens	556
Number of tokens with no correct analysis	287
Number of tokens with no analysis	4
Degree of ambiguity	3.0

Table 4. *Histogram of number of analyses in the training corpus*

Number of analyses	1	2	3	4	5	6	7
Number of tokens	22,318	11,279	7,358	6,302	4,278	2,769	1,847
Percentage	37.3	18.8	12.3	10.5	7.1	4.6	3.1
Number of analyses	8	9	10	11	12	13	>13
Number of tokens	1,300	739	613	281	260	208	314
Percentage	2.2	1.2	1.0	0.5	0.4	0.3	0.5

some problem of the analyzer. Finally, when the analyzer encounters an unknown word, it typically assigns it a proper name analysis, but indicates the fact that no lexical entry exists for this token; we refer to such tokens at out-of-vocabulary (OOV). Some statistics on the training corpus are given in Tables 3–5.

For testing, we also manually annotated a small corpus of one hundred sentences (1,537 tokens, of which 395 do not occur in the training corpus) from the same newspaper (but a very different period). Finally, we also use a large (32 M tokens) *un*annotated corpus, this time of various sources (Itai and Wintner, 2008), to obtain word- and analysis-frequency statistics (see Section 3.4).[5]

2.3 Classification Method

In all the experiments described in this paper we use SNoW (Roth, 1998) as the learning environment, with *winnow* as the update rule. SNoW is a multi-class classifier that is specifically tailored for learning in domains in which the potential number of information sources (features) taking part in decisions is very large, of which NLP is a principal example. It works by learning a sparse network of linear functions over a pre-defined or incrementally learned feature space. SNoW has been already used successfully as the learning vehicle in a large collection of natural language related tasks, including POS tagging, shallow parsing, information extraction tasks *etc*., and compared favorably with other classifiers (Roth, 1998; Punyakanok and Roth, 2001; Florian, 2002). Typically, SNoW is used as a classifier, and predicts using a winner-take-all mechanism over the activation values of the

[5] The annotated corpora, as well as the disambiguation module, are freely available and are distributed by the Knowledge Center for Processing Hebrew, `http://www.mila.cs.technion.ac.il/`.

Table 5. *Distribution of parts of speech in the training corpus*

POS	Adjective	Adverb	Conjunction	Copula	Existential
# tokens	4,344	1,884	1,680	847	296
Percentage	7.3	3.2	2.8	1.4	0.5
POS	foreign	interjection	interrogative	modal	MWE
# tokens	1	18	245	380	3,328
Percentage	0.0	0.0	0.4	0.6	5.6
POS	negation	noun	number exp.	numeral	participle
# tokens	555	16,901	36	1,610	2,148
Percentage	0.9	28.4	0.1	2.7	3.6
POS	preposition	pronoun	proper name	punctuation	quantifier
# tokens	4,816	1,479	3,634	9,845	529
Percentage	8.1	2.5	6.1	16.5	0.9
POS	title	unknown	verb	wprefix	
# tokens	51	59	4,725	100	
Percentage	0.1	0.1	7.9	0.2	

target classes. However, in addition to the prediction, it provides a reliable confidence level in the prediction, which enables its use in an inference algorithm that combines predictors to produce a coherent inference.

2.4 Baseline

For evaluation we consider only the words that have at least one correct analysis in the annotated corpus. Our measure of evaluation is *accuracy*, defined as the ratio of the number of words classified correctly to the total number of words in the test corpus that have a correct analysis. First, we report the performance of simple classifiers tested on a development subset with which we tune several parameters. We then report the results of classifier combination, obtained by training on the entire training corpus and testing on the development and the test corpora. In addition, we perform ten-fold cross-validation runs on the entire training corpus and report the average of ten runs.

The *baseline* tag of a token *w* is an ID of the most prominent analysis of all the occurrences of *w* in the training corpus. This is the analysis that was annotated as the correct one for this token more times than any other analysis. Table 6 illustrates this idea: Suppose a token *w* has three analyses, and suppose that *w* occurs ten times in the training corpus. The first analysis was annotated as the correct one in three instances, the second analysis in four and the last analysis in three. Then the baseline tag of *w* is the ID of the second analysis. The baseline tag of a token also induces a baseline tag for every feature of this token, like part-of-speech, gender, person *etc*. In the example of Table 6, the baseline POS of *w* is verb, the baseline gender is masculine and the baseline person is second.

Table 6. *An example of the baseline tag*

ID	Correct	POS	Gender	Person
1	3	verb	fem.	2nd
2	**4**	**verb**	**masc.**	**2nd**
3	3	noun	fem.	n/a

If w does not occur in the training corpus, we back off and select the baseline tag of each feature of w from the corpus independent of the word w. That is, if a noun is more prominent in the corpus than a verb, then a noun will be chosen as the baseline tag of the POS feature of w. It is more difficult to induce the baseline tag (i.e., the most prominent analysis ID) of such unseen tokens. We rank each analysis based on a number of its features that have the most prominent values. The analysis that receives the highest score is then selected as the baseline tag of w. For example, suppose that the most prominent value of part of speech in the corpus is noun, the gender is feminine and the person is n/a (not applicable). In this case, the first analysis in the example of Table 6 scores one point (for gender), the second analysis scores zero point and the third one scores three points (POS, gender and person). Consequently, in this case the third analysis is chosen as the baseline tag of w. We now define these notions formally.

Let w be a word token (e.g., *\$bth*), and let $\#w$ be the number of occurrences of w in the training corpus. Assume that the morphological analyzer produces k analyses for (each occurrence of) w (e.g., Table 1 shows $k = 8$ for $w = \$bth$). Let f_i be a feature of the analysis, $1 \leq i \leq F$. Let f_i^j, where $1 \leq j \leq k$, be the value assigned by the jth analysis to feature f_i. Again using the example of Table 1, f_1 is part of speech and while f_1^3 is *noun*, f_1^7 is *verb*.

Let A_w^j be the number of occurrences of w for which the jth analysis is annotated as the correct one. For example, if $w = \$bth$, then $A_{\$bth}^3$ is the number of occurrences of *\$bth* in the corpus in which the third analysis is annotated as the correct one. For each feature f_i, $1 \leq i \leq F$, let B_i^j be the number of times f_i^j is the correct value of f_i in the training corpus. Crucially, B_i^j is independent of w. For example, if f_1 is the feature POS, then B_1^7 is the number of tokens in the corpus for which the value of the POS feature is *verb*.

For predicting the value of some feature f_i, the *baseline tag* of some token w is

$$\hat{t}(f_i, w) = \begin{cases} f_i^{\hat{j}}, \text{ where } \hat{j} = argmax_j A_w^j & \text{if } \#w > 0 \\ f_i^{\hat{j}}, \text{ where } \hat{j} = argmax_j B_i^j & \text{if } \#w = 0 \end{cases}$$

That is, if w occurs in the training corpus, the most frequent value $\hat{j}$ associated with f_i in the training corpus is the baseline tag; if w does not occur in the training corpus, the baseline tag is the most frequent value of f_i in the corpus, independent of w. For example, if f_1 is the feature POS, then the baseline tag of unseen words is the most frequent part of speech in the corpus, namely *noun* (see Table 5).

As for the analysis ranking task, again if w occurs in the training corpus, the baseline analysis is the most frequent analysis assigned to w. If w does not occur in the corpus, however, we define a voting mechanism among all the features that are involved in the analysis of w. Let C_w^j be the score function of analysis j of some token w, given by

$$C_w^j = \sum_{1 \le i \le F} \Lambda(f_i^j), \text{ where } \Lambda(f_i^j) = \begin{cases} 1 & \text{if } \hat{t}(f_i, w) = f_i^j \\ 0 & \text{otherwise} \end{cases}$$

Thus, for the ranking task, the baseline analysis of some token w is

$$\hat{T}(w) = \begin{cases} argmax_j A_w^j & \text{if } \#w > 0 \\ argmax_j C_w^j & \text{if } \#w = 0 \end{cases}$$

3 Disambiguation by classification

We decouple the task of morphological disambiguation by defining several basic classifiers. The predictions of the classifiers are then combined to yield a consistent prediction.

3.1 Simple classifiers

The simple classifiers are all built in the same way. These are trained on feature vectors that are generated from the annotated output of the morphological analyzer. We define several classifiers for the attributes of the morphological analyses. Note that some attributes do not apply for all the analyses (see Table 2). For example, the attribute ‘tense’ is inapplicable for an analysis whose main part of speech is noun. We therefore add a value of ‘irrelevant’ for inapplicable attributes. Also, in some cases the corpus does not have a value for some feature for which such a value *is* necessary. In such cases, the corpus usually lists the value as *unspecified*; we account for these values as well. We describe each of the classifiers below.

Part-of-speech (POS). The (POS) classifier predicts the main part of speech of the word. The possible values of this attribute are ‘adjective’, ‘adverb’, ‘conjunction’, ‘copula’, ‘existential’, ‘foreign’, ‘interjection’, ‘interrogative’, ‘modal’, ‘MWE’, ‘negation’, ‘noun’, ‘number expression’, ‘numeral’, ‘participle’, ‘preposition’, ‘pronoun’, ‘proper name’, ‘punctuation’, ‘quantifier’, ‘title’, ‘unknown’, ‘verb’ and ‘wprefix’.

Segmentation. The segmentation classifier predicts the index of the beginning of the base. This classifier ranks seven options [0–6], where 6 is the length of the longest possible prefix sequence.

Gender. The gender classifier ranks five values: ‘masculine’, ‘feminine’, ‘masculine and feminine’, ‘unspecified’ and ‘irrelevant’.

Number. The number classifier ranks seven values: ‘singular’, ‘plural’, ‘dual’, ‘singular and plural’, ‘dual and plural’, ‘unspecified’ and ‘irrelevant’.

Person. The person classifier ranks six values: ‘first’, ‘second’, ‘third’, ‘any’, ‘unspecified’ and ‘irrelevant’.

Has properties. This is a binary classifier distinguishing between categories that are atomic (e.g., conjunction, interjection, interrogative, negation, punctuation and quantifier) and categories whose words have attributes (e.g., noun, adjective, adverb, auxiliary verb, particle, preposition, pronoun, proper name and verb).

Tense. The tense classifier ranks nine values: 'past', 'present', 'beinoni', 'future', 'imperative', 'infinitive', 'bare infinitive', 'unspecified' and 'irrelevant'.[6]

Definite article. The definiteness classifier ranks four values: 'definite', 'indefinite', 'unspecified' and 'irrelevant'. It identifies both implicit and explicit forms of definiteness.

Status. The status classifier ranks five values: 'absolute', 'construct', 'absolute and construct', 'unspecified' and 'irrelevant'.

Pronoun type. The pronoun type classifier ranks eight values, reflecting the different sub-categories of Hebrew pronouns: 'personal', 'demonstrative', 'relativizer', 'impersonal', 'reflexive', 'interrogative', 'unspecified' and 'irrelevant'.

Participle type. The participle classifier ranks seven values, reflecting the function of the participle in the sentence: 'noun', 'noun or adjective', 'noun or verb', 'noun or adjective or verb', 'adjective', 'verb' and 'irrelevant'.

These classifiers are (almost) sufficient for full disambiguation, see Section 3.4.

3.2 Training the simple classifiers

3.2.1 Feature set

Each word in the training corpus induces features that are generated for itself and its immediate neighbors (the 'window'). These features are generated from the output of the morphological analyzer. For each word in the window, we generate the following features:

- The main part-of-speech (twenty-five options)
- Number (seven options)
- Gender (five options)
- Person (six options)
- Tense (nine options)
- Status (five options)
- Definite article (four options)
- Relevant features. In order to emphasize the value of 'irrelevant' in some of the attributes, we created an additional feature for those attributes that are not applicable to all the parts of speech (see Table 2). These features are binary features whose values are 'relevant' and 'irrelevant'. For example, tense is not relevant for an analysis whose part of speech is noun. The features are 'is the number relevant' (this is the disjunction of all the values of 'Number'

[6] The term *tense* is used in its Hebrew traditional sense here, referring to a combination of tense and aspect. Traditionally, the values of this feature are past, present, future, imperative, bare infinitive and to-infinitive. The annotated corpus we used had occurrences of both 'present' and 'beinoni' as the values indicating present tense, so we allow both values, and treat them as different. This should probably be cleaned in the corpus, and will result in improved accuracy. See also Adler *et al.* (2008b).

Table 7. *Accuracy of all basic classifiers, varying window sizes*

Classifier	0–0	0–1	0–2	1–0	1–1	1–2	2–0	2–1	2–2
POS	90.99	92.24	92.28	92.57	92.46	**92.61**	92.14	92.13	92.10
Gender	92.86	93.90	93.38	93.47	**94.14**	94.09	93.76	93.98	94.00
Number	94.00	**95.02**	94.64	94.35	94.69	94.53	94.29	94.91	94.37
Person	94.41	95.00	94.75	94.59	**95.05**	94.74	94.86	94.91	94.76
Tense	97.79	98.01	97.99	98.00	**98.16**	98.06	98.08	98.11	98.03
Definiteness	86.98	89.35	89.10	87.95	**89.53**	89.48	88.20	89.48	89.21
Status	90.84	92.62	92.23	91.68	**93.10**	92.87	91.71	92.91	92.14
Segmentation	98.21	98.38	98.33	**98.40**	98.38	98.36	98.39	98.31	98.28
Has properties	99.47	99.40	99.38	99.32	**99.49**	99.42	99.30	99.45	99.36
Pronoun type	98.69	98.97	99.18	99.19	**99.36**	99.24	99.16	99.31	99.19
Participle type	98.03	98.05	98.02	98.05	98.06	98.05	**98.07**	98.06	98.01

except 'irrelevant'), 'is the gender relevant', 'is the person relevant', 'is the tense relevant', 'is the status relevant' and 'is the definite article relevant'.

- All the prefixes (twelve options). Each of the possible prefixes of Hebrew (such as 'm', '$', 'm$' *etc.*) is a feature; this feature indicates if an analysis has one of those prefixes.
- Number–gender–person combination of the suffix. As described in Section 1.1, nominals take pronominal suffixes which inflect for number, gender and person. This inflection is combined into a single value (there can be 210 options; 5 Gender $\times$ 7 Number $\times$ 6 Person).
- Does the word have a suffix? (two options)
- Complex POS (a feature that combines the part of speech of the prefix, base and suffix) (200 options). This feature represents the full POS of a surface form, and is created from the cartesian product of twenty-five values of POS, four values of the prefix POS ('conjunction', 'preposition', 'determiner' and 'relativizer/subordinator') and an indication of whether the word has a suffix.
- The surface form of the word (as many values as the size, in types, of the training corpus).
- The lemma of the base (approximately 21,000 values, the size of our lexicon). To reduce data sparseness, we consider not only the surface form of the word but also its lemma.

3.2.2 Determining the window size

Each of the supervised classifiers can be configured in several ways. First we need to determine the size of the training and testing window. We experimented with several window sizes, up to two words before and after the current word. Table 7 depicts the best window size for each classifier. (We used the development set in these experiments.)

3.2.3 Determining feature values in sequential prediction

Our task is sequential in nature: Determining the part of speech of the current word obviously depends on the part of speech of its immediate neighbors. This information is known during training, but not at actual testing time. Therefore, one must decide how to handle this uncertainty.

One question is whether or not to use the correct analysis of the target and its adjacent words when generating feature vectors. The correct analysis is available during training, but not during testing. Obviously, the correct analysis can improve the accuracy of the prediction; but using it for training creates a discrepancy between the training process and the way features are generated for testing. In order to keep training and testing feature vectors consistent, we therefore use instead the most prominent feature values, determined by the same criteria that are used for baseline computation (see Section 2.4) in both testing and training. Experimenting with different approaches yielded worse results.

Another question involves the proper treatment of words that are not present in the training material (unseen items). For such words, which are again only present at test time, the only possible solution is to generate feature vectors that are independent of the word itself. For example, when a value for the feature POS is generated for some unseen word w, this value is determined to be the most frequent part-of-speech in the training corpus, independent of w. Referring back to Section 2.4, the question is whether to use A^j_w or C^j_w for computing $\hat{T}(w)$. Again, this entails a discrepancy between the way features are generated for training and testing. To overcome this problem, we experiment with three ways to generate features at training:

(1) Generate features based on the actual words in the text (since no unseen items exist during training).
(2) Generate features based on the actual words for a randomly determined portion of the training corpus, and treat all other tokens as unseen. We chose 25 percent as the portion of the text treated as unseen, reflecting the ratio of unseen items in our corpus.
(3) Generate features as if *all* words in the training and testing corpus are unseen.

The results of all these experiments are shown in Table 8. We report the accuracy of each classifier and the reduction in error rate (ERR), compared to the baseline, which is defined as in Section 2.4.

Clearly, most classifiers perform much better than the baseline. Specifically, on the important task of POS tagging we demonstrate decent accuracy of 93.1 percent and reduction of almost 40 percent of the errors compared with the baseline. High gains are clearly evident for other classifiers as well, the only exception being the segmentation classifier, for which the baseline performs slightly better. We do not have any explanation for this phenomenon.

Interestingly, for six out of eleven classifiers, the third method outperforms the first one even though it uses less accurate information during both training and testing. This effect is enhanced when *only* unseen items are considered. Table 9 displays exactly the same evaluation, constrained only to unseen items. The third method is the best for ten out of eleven classifiers.

Table 8. *Accuracy and error-rate reduction of simple classifiers, compared with the baseline, reflecting three methods for feature generation during training*

	Baseline	Method 1		Method 2		Method 3	
	Acc	Acc	ERR	Acc	ERR	Acc	ERR
POS	88.80	92.61	34.02	**93.11**	38.48	93.09	38.30
Gender	93.29	94.14	12.67	94.59	19.37	**95.14**	27.57
Number	94.04	95.02	16.44	95.13	18.29	**95.33**	21.64
Person	94.59	95.05	8.50	**95.27**	12.57	94.74	2.77
Tense	97.84	**98.16**	14.81	98.11	12.50	97.80	–
Definite article	87.72	89.53	14.74	**90.35**	21.42	90.16	19.87
Status	91.31	93.10	20.60	93.41	24.17	**93.58**	26.12
Segmentation	**98.48**	98.40	–	98.41	–	97.58	–
Has properties	99.49	99.49	0.00	99.48	–	**99.52**	5.88
Pronoun type	99.09	**99.36**	29.67	99.25	17.58	99.25	17.58
Participle type	98.03	98.06	1.52	**98.08**	2.54	98.06	1.52

Table 9. *The same evaluation as in Table 8, limited to unseen items only*

	Baseline	Method 1		Method 2		Method 3	
	Acc	Acc	ERR	Acc	ERR	Acc	ERR
POS	81.16	87.42	33.23	88.66	39.81	**89.98**	46.82
Gender	86.54	88.55	14.93	92.14	41.60	**94.04**	55.72
Number	89.54	91.59	19.60	93.23	35.28	**95.17**	53.82
Person	91.26	92.14	10.07	92.94	19.22	**93.75**	28.49
Tense	95.50	96.09	13.11	96.20	15.56	**96.82**	29.33
Definite article	75.20	79.37	16.81	82.52	29.52	**83.98**	35.40
Status	84.82	88.22	22.40	89.21	28.92	**90.01**	34.19
Segmentation	95.98	96.16	4.48	96.34	8.96	**96.78**	19.90
Has properties	99.85	**99.85**	0.00	99.78	–	99.74	–
Pronoun type	99.63	**99.85**	59.46	99.82	51.35	**99.85**	59.46
Participle type	94.55	94.51	–	94.59	0.01	**94.84**	5.32

These results also demonstrate quite clearly that the classifiers perform much worse on unseen words than on seen ones. This is not surprising; it stems from the fact that unseen words are much less common in the text than known words, and the distribution of their POS classes is quite different from that of known words. For example, unseen words rarely belong to closed POS classes, like punctuation or preposition. On the other hand, words in closed classes comprise more than 30 percent of the text and are usually not ambiguous. Table 10 depicts the distribution of major POS classes among seen and unseen tokens in the development corpus, exhibiting significant differences. In particular, almost half of the unseen tokens are nouns, and 14 percent of them are verbs, compared to one-quarter and 5.54 percent, respectively, of the known tokens. On the other hand, more than 20 percent of the known tokens, but none of the unseen ones, are punctuation.

Table 10. *Part-of-speech distribution in the development corpus and the accuracy of the POS classifier: known versus unseen tokens*

	Known		Unseen	
POS	Rate	Accuracy	Rate	Accuracy
Noun	23.14	95.31	46.31	94.47
Verb	5.59	92.81	14.23	90.49
Adjective	6.03	87.19	10.39	85.21
Proper Name	4.24	89.28	11.60	88.64
Punctuation	20.52	100.00	0.04	100.00
Preposition	10.23	96.93	0.80	95.45

We also report in Table 10 the performance of the POS classifier on specific POS classes, for known words and unseen ones. The classifier performs much better on the closed POS classes, like punctuation and prepositions, than on open classes, in particular adjectives and proper names. Since the closed POS classes are much more common among known words, this explains the gap between known and unseen words in the overall performance of the POS classifier.

Based on these observations, we conclude that the third training scenario performs best, especially on the more difficult cases. Also, it does not rely on token-specific information, such as the most prominent analysis, and therefore it is more robust. This is then the method we adopt in this work.

3.3 Combination of classifiers

Classifier combination is based on the fact that linguistic knowledge is incorporated into the analyzer. Each analysis is created using a set of morphological rules, and thus the attributes of a specific analysis satisfy those morphological constraints. Each classifier predicts a single morphological feature, which induces a rank of the entire list of analyses.

To combine the predictions of the simple classifiers, we accumulate the scores of all the classifiers. We experiment with three types of combinations: naïve, linear and hierarchical.

Naïve combination. We use the confidence level of each SNoW classifier as the score. We accumulate the scores of all the classifiers, yielding a single score per analysis, thereby inducing a rank on the list of analyses.

Linear combination. We treat the problem of combining the outcomes of the classifiers as a linear regression problem. We define the error as the ratio of wrongly disambiguated tokens to the size of the corpus. Each classifier is a variable in a linear function. The coefficients are induced in a procedure called Powell's search (Powell, 1964), which is a common method for finding the minimum of a function of several variables without computing derivatives. We use the development corpus to optimize weights.

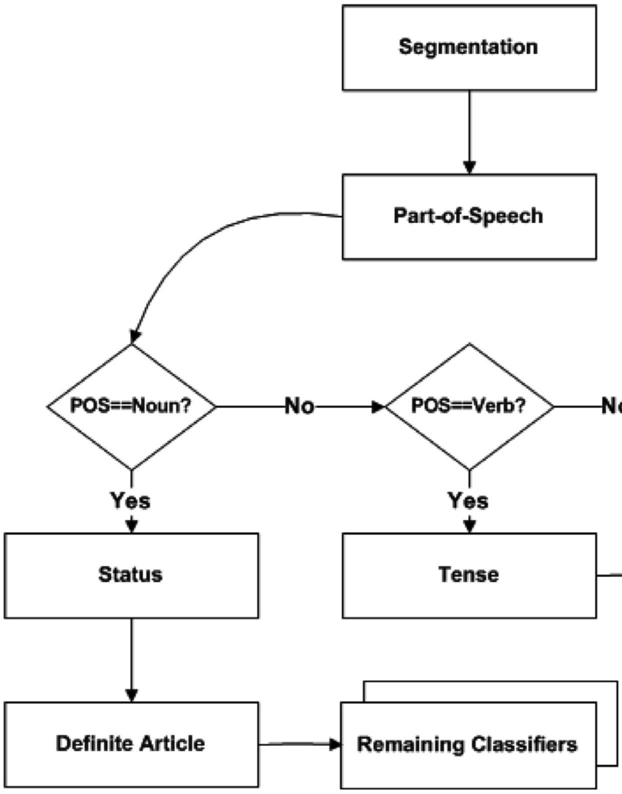

Fig. 1. Hierarchical combination.

Hierarchical combination. The classifiers are applied one-by-one in a predefined order that is based on linguistic knowledge. Each step reduces the level of remaining ambiguity until a token is fully disambiguated or there are no more classifiers. Figure 1 shows the order in which the classifiers are applied.

3.4 Backing off to unsupervised classification

None of the combination strategies discussed above can guarantee a single, unique top-ranked analysis. This is because in some cases two or more analyses with identical morphological features differ only in their lexical ID. Distinguishing between such analyses requires word sense disambiguation rather than morphological disambiguation (Section 1.4). Solving this problem is beyond the scope of this work, and therefore we use a simplistic approach to disambiguate such analyses. We train an additional unsupervised classifier that chooses the most frequent lexical item. The frequencies are computed from a large morphologically analyzed (but not annotated or disambiguated) corpus (Section 2); see also Levinger *et al.* (1995) and Carmel and Maarek (1999).

Given a word with two analyses that only differ in their lexical ID, such as analyses 7 and 8 in Table 1, we break the tie by preferring the analysis reflecting the lexical ID that is more frequent in the unannotated (larger) corpus. The motivation

Table 11. *Results: accuracy of full disambiguation without backoff*

Task	Baseline	Naïve	Linear	Hierarchical
Development set	82.12	83.35	**83.89**	82.08
Ten-fold cross-validation	79.02	**82.73**	82.51	81.40
Test set	81.91	**80.74**	80.85	79.90

Table 12. *Results: accuracy of full disambiguation with backoff*

Task	Baseline	Naïve	Linear	Hierarchical
Development set	82.12	84.87	**85.41**	83.58
Ten-fold cross-validation	79.02	**84.15**	83.94	82.82
Test set	81.91	**82.63**	82.50	81.78

for this step is the observation that such forms may have different inflectional paradigms, so that other inflected forms of the same lexical ID can be more or less frequent in the unannotated corpus. Note that even this does not guarantee full disambiguation: some (different!) base forms have completely identical inflectional paradigms, and hence the unannotated corpus will have exactly as many occurrences as one of the analyses of the other.

3.5 Results

Tables 11 and 12 depict the accuracy of our disambiguation module with and without the unsupervised backoff module described in Section 3.4. As described above we report ten-fold cross-validation results for the training set (59,363 tokens), as well as the results of training on the entire training set and testing on the development (12,189) and test (1,537 tokens) sets. In all experiments we use the third training scenario (Section 3.2). We also provide the baselines defined in Section 2.4.

The results show that the naïve combination outperforms other combinations on the ten-fold cross-validation task and on the test set. In this scenario, the overall accuracy of full morphological disambiguation is 84.15 percent under ten-fold cross-validation evaluation, reflecting a reduction of approximately one quarter of the errors compared with the baseline. The accuracy on the (out-of-domain) test set is only slightly lower, i.e., 82.63 percent. The contribution of the unsupervised backoff module is an additional 1.5 to 1.9 percent points.

A statistical significance analysis, performed on the results of the ten-fold cross-validation experiments, shows that all combinations are statistically better than the baseline ($p < 0.001$). Both the naïve ($p < 0.001$) and the linear ($p < 0.01$) combinations are statistically better than the hierarchical combination; and finally, the difference between the naïve and the linear combinations is not statistically significant.

To better appreciate the contribution of our method, we conducted yet another evaluation, in which we ignored some subtle distinctions that previous works (in particular, Adler and Elhadad 2006) do not consider as significant. In other words, we view as ‘correct’ analyses that only differ from the human-annotated analysis in the sub-category of the main POS (for pronouns, proper names, participles *etc.*) or in the lexical ID. In such a scenario, the accuracy of the disambiguation module is 86.14 percent on the test set, 88.58 percent on the development set and 88.00 percent under ten-fold cross-validation runs. These figures compare favorably with the best reported results on this task.

Finally, to emphasize the contribution of classifier combination, we also evaluated the accuracy of the disambiguation module in predicting only the major POS category of each word. This is obtained simply by projecting the correct analysis onto the POS coordinate. Whereas the simple POS classifier was correct in 91.54 percent of the cases on the test set (93.09 percent on the development set), the disambiguator yields the correct POS in 92.19 percent of the words in the test set (93.89 percent of the development set, 93.17 percent under ten-fold cross-validation evaluation), a small but significant improvement, reflecting around 10 percent reduction in error rate. Other simple classifiers, predicting other features of the analysis, improve similarly in this way.

4 Error analysis

In this section we analyze the most frequent errors resulting from the disambiguation of the test set by the best performing system. Out of the 1,537 tokens, the system incorrectly disambiguated 239 tokens and another twenty-eight tokens remain ambiguous. We distinguish between incorrect and partially disambiguated tokens and provide a list of the most common errors in both cases.

4.1 Incorrectly disambiguated tokens

Construct. In many cases our system fails to correctly identify whether the word is in the absolute or the construct state. This problem is responsible for 14.6 percent of all errors. Overall, the system fails in 1.8 percent of the cases with this particular ambiguity.

Hidden definite article. One of the peculiarities of the Hebrew orthographic system is that the definite article ‘*h*’ is omitted, then it directly follows one of the prepositions ‘*b*’, ‘*l*’, ‘*k*’. In many such cases the system mistakenly tags the words as indefinite. This issue is responsible for 12.9 percent of all errors. The system fails in 3.5 percent of relevant cases.

Participles. Participles are often mistagged as nouns, adjectives or verbs. In few cases, words that were tagged as participles are in fact verbs in the past tense, nouns or adjectives. Participle-related errors are responsible for 10 percent of all errors, and 3.1 percent of the cases involving participle-related ambiguity are tagged wrongly.

Proper name. The Hebrew orthography does not capitalize proper names. In addition, many Hebrew proper names are derived from common nouns. Consequently, distinguishing proper names from nouns is challenging. The system makes quite a few mistakes (8.8 percent of all errors), mistagging nouns as proper names and the other way around, resulting in 5.2 percent of all cases with such ambiguity.

Noun versus verb. In some cases nouns can be mistakenly identified as verbs and the other way around. This is especially true for nominalizations of verbs. This issue is responsible for 5.9 percent of all errors. Less than 1 percent of the cases with this ambiguity are mistagged by the system.

Noun versus adjective. Another similar issue is the orthographic similarity between nouns and adjectives. For example, the word $*kv* can mean either 'silence' (noun) or 'silent/pacific' (adjective). This ambiguity is responsible for 5.4 percent of all errors. Only 0.5 percent of the relevant tokens are mistagged.

Wrong lexical item. The system resolves word sense ambiguity using an unsupervised classifier based on the frequency of lexical items in an analyzed, but not annotated, corpus. The wrong choices of this classifier are responsible for 5 percent of all errors and 20 percent of the relevant cases.

Participle subtype. Participles have three possible subtypes: verb, noun or adjective. In some cases (4.2 percent of all errors) the system fails to make the correct choice. This happens in 1.3 percent of the relevant cases.

Proper name subtype. Proper names are subcategorized to persons, locations and organizations. For example, the proper name *lwndwn* 'London' can refer to a person or to the city. The errors in proper name subtype comprise 2.9 percent of all errors (less than 1 percent of the cases with proper name subtype ambiguity).

Pronoun versus copula. Personal pronouns and copulas have some common forms in Hebrew, which have to be disambiguated. This issue is responsible for 2.9 percent of all errors or 7.3 percent of the cases with such ambiguity.

Multi-word expressions (MWE). In some cases the system fails to identify that a word belongs to a MWE and tags it as if it was a stand-alone word. This issue is responsible for 2.9 percent of all errors; 1.4 percent of MWEs are mistagged by the system.

4.2 Partially disambiguated tokens

Quantifier subtype. Our disambiguation module does not distinguish between partitive and determinizing quantifiers. This issue is responsible for 35.7 percent of all partially disambiguated tokens. The easiest way to overcome this problem is to construct an additional classifier for determining the subtype of a quantifier.

Lexical item. In some cases even the unsupervised classifier cannot make a distinction between different senses of a word. For example, $*ibh* can mean either 'return' or 'old age'. Both are nouns, and both occur in exactly the same inflected forms, and hence cannot be distinguished by the unsupervised classifier. This issue explains 21.4 percent of all partially disambiguated tokens. Note that this is really an issue of word sense, rather than morphological, disambiguation.

Proper name subtype. Misclassification of the category of proper names is responsible for 17.9 percent of partial disambiguation. Again, a dedicated classifier could alleviate some of the problem here.

Suffix. In some cases the system fails to tell whether or not a word (typically, ending with *i*) has a pronominal suffix. For example, the word *axri* can mean either ‘after’ or ‘after me’ depending on whether it has a pronominal suffix or not. This issue is responsible for 14.3 percent of all partially disambiguated tokens.

5 Conclusions

Morphological disambiguation of Hebrew is a hard task. It involves, in theory, thousands of possible tags, although in practice only a few hundreds are witnessed in a standard corpus. As shown by Habash and Rambow (2005), decoupling this task into several simple tasks improves the accuracy of morphological disambiguation for Arabic. We have shown here that using the same techniques for Hebrew improves accuracy over the baseline. Our best result, 84.15 percent accuracy, was obtained using simple supervised classifiers, for each individual morphological feature, backing off to an unsupervised classifier to reduce the remaining ambiguity.

We believe that these results can be further improved in various ways. The use of supervised machine-learning methods requires large annotated corpora for training. In this work we used a very small corpus, of a little less than 60,000 word tokens. Increasing the size of the training corpus can improve the accuracy of the basic classifiers, and hence of the full disambiguation process. As an indication that more training material is likely to improve the accuracy, we depict in Figure 2 a plot of the accuracy of the full disambiguation task as a function of the size of the training corpus. As the learning curve clearly shows, the improvement is nearly linear, and we thus have good reasons to believe that convergence of the accuracy is still faraway.

We also expect that fine-tuning of the classifiers, and in parallel, fine-tuning of SNoW, can also improve the accuracy of the basic classifiers. Another potential direction for future research would be to investigate more sophisticated methods for combining the outcomes of various classifiers. While our results demonstrate that the hierarchical combination does not improve the accuracy, one could possibly rely on classifiers with higher confidence first, and use their predictions to reduce the ambiguity and only then use less confident classifiers.

Perhaps the most promising research direction, however, is in combining morphological disambiguation with parsing. In languages with rich morphology, such as Hebrew, these two tasks are closely related and highly intertwined; a joint solution is likely to improve the accuracy of both tasks. Several preliminary works have recently explored this direction (Tsarfaty 2006; Cohen and Smith 2007; Goldberg and Tsarfaty 2008; Lee, Naradowsky and Smith 2011); further research is still needed in order to obtain the best possible results.

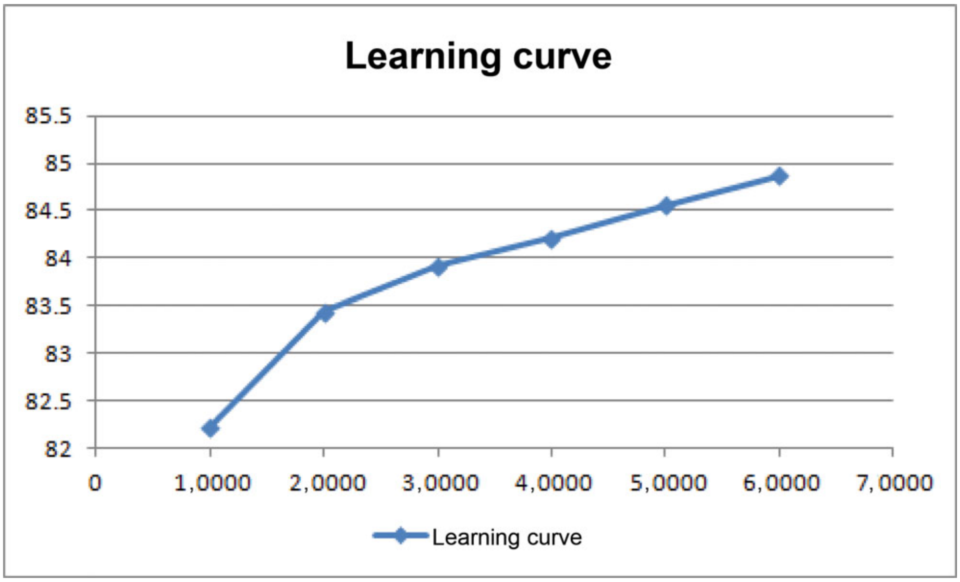

Fig. 2. (Colour online) Learning curve.

Acknowledgments

We are extremely grateful to Dan Roth for his support and advice; to Inbar Tubi, Yuval Pinter and Gily Chen for corpus annotation; and to Dalia Bojan, Gassan Tabajah, Yossi Jacob and Alon Itai for the implementation and support of the corpora and morphological analyzer. This research was supported by The Israel Science Foundation (Grants No. 137/06).

References

Adler, M., and Elhadad, M. July 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. 665–72. Stroudsburg, PA: Association for Computational Linguistics. `http://www.aclweb.org/anthology/P/P06/P06-1084`.

Adler, M. September 2007. Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach. PhD thesis, Ben-Gurion University.

Adler, M., Goldberg, Y., Gabay, D., and Elhadad, M. June 2008a. Unsupervised lexicon-based resolution of unknown words for full morphological analysis. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, pp. 728–736. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1083`.

Adler, M., Netzer, Y., Goldberg, Y., Gabay, D., and Elhadad, M. May 2008b. Tagging a Hebrew corpus: the case of participles. In *Proceedings of the*

Sixth International Conference on Language Resources and Evaluation (LREC'08), Marrakech, Morocco. European Language Resources Association (ELRA). ISBN 2-9517408-4-0. `http://www.lrec-conf.org/proceedings/lrec2008/`.

Bar-Haim, R., Sima'an, K., and Winter, Y. June 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, MI, pp. 39–46. Stroudsburg, PA: Association for Computational Linguistics. `http://www.aclweb.org/anthology/W/W05/W05-0706`.

Bar-Haim, R., Sima'an, K., and Winter, Y. 2008. Part-of-speech tagging of Modern Hebrew text. *Natural Language Engineering* **14**(2): 223–51.

Bentur, E., Angel, A., and Segev, D. December 1992. Computerized analysis of Hebrew words. *Hebrew Linguistics* **36**: 33–8 (in Hebrew).

Brill, E. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics* **21** (4): 543–66.

Carmel, D., and Maarek, Y. July 1999. Morphological disambiguation for Hebrew search systems. In *Proceedings of the 4th International Workshop, NGITS-99*, Lecture Notes in Computer Science, no. 1649, pp. 312–25. New York: Springer.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. March 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy, pp. 133–140. Association for Computational Linguistics. doi: 10.3115/974499.974523. URL `http://www.aclweb.org/anthology/A92-1018`.

Cohen, S. B., and Smith, N. A. June 2007. Joint morphological and syntactic disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic, pp. 208–17. Stroudsburg, PA: Association for Computational Linguistics. `http://www.aclweb.org/anthology/D/D07/D07-1022`.

Daya, E., Roth, D., and Wintner, S. July 2004. Learning Hebrew roots: machine learning with linguistic constraints. In *Proceedings of EMNLP'04*, Barcelona, Spain, pp. 357–64.

Daya, E., Roth, D., and Wintner, S. September 2008. Identifying semitic roots: machine learning with linguistics constraints. *Computational Linguistics* **34**(3): 429–48.

Florian, R. 2002. Named entity recognition as a house of cards: classifier stacking. In *Proceedings of CoNLL-2002*, Taiwan, pp. 175–8.

Goldberg, Y., and Tsarfaty, R. June 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, Columbus, OH, pp. 371–9. Stroudsburg, PA: Association for Computational Linguistics. `http://www.aclweb.org/anthology/P/P08/P08-1043`.

Habash, N., and Rambow, O. June 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)*, Ann Arbor, MI, pp. 573–80. Stroudsburg, PA: Association for Computational Linguistics. http://www.aclweb.org/anthology/P/P05/P05-1071.

Hajič, J. 2000. Morphological tagging: data vs. dictionaries. In *Proceedings of ANLP-NAACL Conference*, Seattle, WA, pp. 94–101.

Hajič, J., and Hladká, B. 1998. Tagging inflective languages: prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Stroudsburg, PA, pp. 483–90. Stroudsburg, PA: Association for Computational Linguistics.http://dx.doi.org/10.3115/980845.980927.

Itai, A., and Wintner, S. March 2008. Language resources for Hebrew. *Language Resources and Evaluation* **42**(1): 75–98.

Lafferty, J., McCallum, A., and Pereira, F. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, Williamstown, MA, pp. 282–9.

Lee, J., Naradowsky, J., and Smith, D. A. June 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, OR, pp. 885–94. Stroudsburg, PA: Association for Computational Linguistics. http://www.aclweb.org/anthology/P11-1089.

Lembersky, G. March 2003. *Named Entity Recognition in Hebrew.* Master's thesis, Department of Computer Science, Ben Gurion University, Beer Sheva, Israel (in Hebrew).

Levinger, M., Ornan, U., and Itai, A. September 1995. Learning morpho-lexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics* **21**(3): 383–404.

Manning, C. D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA.: The MIT Press.

Marshall, I. 1983. Choice of grammatical word-class without global syntactic analysis: tagging words in the LOB corpus. *Computers and the Humanities* **17**: 139–50.

McCallum, A., Freitag, D., and Pereira, F. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of Eighteenth International Conference on Machine Learning (ICML-00)*, Stanford, CA.

Powell, M. J. D. January 1964. An efficient method for finding the minimum of a function of several variable without calculating derivatives. *The Computer Journal* **7**(2): 155–62.

Punyakanok, V., and Roth, D. 2001. The use of classifiers in sequential inference. In *Proceedings of the 2000 Conference on Advances in Neural Information Processing Systems 13 (NIPS-13)*, Vancouver, British Columbia, Canada, pp. 995–1001. Cambridge, MA: The MIT Press.

Roth, D. 1998. Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of AAAI-98 and IAAI-98*, Madison, WI, pp. 806–13.

Roth, D., and Zelenko, D. 1998. Part of speech tagging using a network of linear separators. In *The 17th International Conference on Computational Linguistics (COLING-ACL 98)*, Montreal, Canada, pp. 1136–42.

Segal, E. 1997. Morphological analyzer for unvocalized Hebrew words. Unpublished work. `http://www.cs.technion.ac.il/~erelsgl/hmntx.zip` Accessed 15 July, 2012.

Segal, E. October 1999. *Hebrew Morphological Analyzer for Hebrew Undotted Texts*. Master's thesis, Technion, Israel Institute of Technology, Haifa, Israel (in Hebrew).

Shacham, D., and Wintner, S. June 2007. Morphological disambiguation of Hebrew: a case study in classifier combination. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning (EMNLP-CoNLL 2007)*, Prague, Czech Republic. Stroudsburg, PA: Association for Computational Linguistics.

Sima'an, K., Itai, A., Winter, Y., Altman, A., and Nativ, N. 2001. Building a tree-bank of Modern Hebrew text. *Traitement Automatique des Langues* **42**(2): 347–380.

Tsarfaty, R. July 2006. Integrated morphological and syntactic disambiguation for modern Hebrew. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*, Sydney, Australia, pp. 49–54. Stroudsburg, PA: Association for Computational Linguistics. `http://www.aclweb.org/anthology/P/P06/P06-3009`.

Wintner, S. 2008. Strengths and weaknesses of finite-state technology: a case study in morphological grammar development. *Natural Language Engineering* **14**(4): 457–69. ISSN 1351-3249. .

Yona, S., and Wintner, S. April 2008. A finite-state morphological grammar of Hebrew. *Natural Language Engineering* **14**(2): 173–90.