Morphological Disambiguation of Hebrew

Dan Shacham

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE MASTER DEGREE

University of Haifa
Faculty of Social Science
Department of Computer Science

January, 2007

Morphological Disambiguation of Hebrew

By: Dan Shacham
Supervised By: Dr. Shuly Wintner

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE MASTER DEGREE

University of Haifa
Faculty of Social Science
Department of Computer Science

January, 2007

Approved by: _____ Date:_____
(supervisor)

Approved by: _____ Date:_____
(Chairman of M.A Committee)

Contents

Morphological Disambiguation of Hebrew

Dan Shacham

Abstract

Morphological analysis is a crucial stage in a variety of natural language processing applications. When languages with complex morphology are concerned, even shallow applications such as search engines, information retrieval or question answering, let alone heavier applications such as machine translation, require morphological analysis and disambiguation as a first step. The lack of a morphological disambiguation module for languages such as Hebrew or Arabic handicaps the performance of many other applications.

We present the HAifa morphological DisAmbiguation System (HADAS). This system uses the output of a morphological analyzer and a limited linguistic knowledge, for disambiguating Hebrew morphologically annotated text. HADAS consists of several (currently, 10) simple classifiers and a module which combines them. We build a classifier for each individual morphological feature. These classifiers are trained on feature vectors that are generated from the output of an annotated morphological analyzer, and ranks the possible analyses of each word. We investigate a number of techniques for combining and disambiguating the results produced by those classifiers. These techniques decrease the average level of ambiguity from 2.4 analyses per word to only 1.1 analyses per word. Our best result, 87.27% accuracy, was obtained using the simple supervised clasifiers, and a module for combining the results of those classifiers. This module consists of two phases; first we compose the confidence scores of all the analyses, as predicted by the classifiers. In the second phase we use a few context-dependent constraints to rule out some of the paths defined by the possible outcomes of the morphological analyzer on a sequence of words.

1 Introduction

1.1 Hebrew morphology

Hebrew and Arabic, like other Semitic languages, have rich and complex morphology. The major word formation machinery is root-and-pattern, where roots are sequences of three (typically) or more consonants, called *radicals*, and patterns are sequences of vowels and, sometimes, also consonants, with "slots" into which the root's consonants are being inserted. Words are created by *interdigitating* roots into patterns: the first radical is inserted into the first consonantal slot of the pattern, the second radical fills the second slot and the third fills the last slot. Inflectional morphology is highly productive and consists mostly of suffixes, but sometimes of prefixes or circumfixes. In general, inflectional morphology can be assumed to be concatenative, but derivational morphology is certainly non-concatenative.

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

The combination of a root with a pattern produces a *base* (or a *lexeme*), which can then be inflected in various forms. Nouns, adjectives and numerals inflect for number (singular, plural and, in rare cases, also dual) and gender (masculine or feminine). In addition, all these three types of nominals have two phonologically distinct forms, known as the *absolute* and *construct* states (the latter functions in compounds, see, e.g., Borer (1984)). Unfortunately, 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 pronominal suffixes which are interpreted as possessives. These inflect for number, gender and person: *spr+h→sprh* "her book", *spr+km→sprkm* "your book", etc. As expected, these processes involve certain morphological alternations, as in *mlkh+h→mlkth* "her queen", *mlkh+km→mlktkm* "your queen".

Verbs inflect for number, gender and person (first, second and third) and also for a combination of tense and aspect, which is traditionally analyzed as having the values past, present, future, imperative and infinite. Verbs can also take pronominal suffixes, which in this case are interpreted as direct objects, but such constructions are rare in contemporary Hebrew of the registers we are interested in. 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.

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

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

'	b	g	d	h	w	z	x	v	i	k	l	m	n	s	y	p	c	q	r	$	t
א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ	ל	מ	נ	ס	ע	פ	צ	ק	ר	ש	ת

consonant *[y]*. On top of that, the script dictates that many particles, including four of the most frequent prepositions (*b* "in", *k* "as", *l* "to" and *m* "from"), the definite article *h* "the", the coordinating conjunction *w* "and" and some subordinating conjunctions (such as *$* "that" and *k$* "when"), all attach to the words which immediately follow them. There is only one definite article in Hebrew, *ha-*, it does not inflect and it is attached (pre-nominally) to words. When the definite nominal is prefixed by one of the prepositions *b*, *k* or *l*, the definite article *h* is assimilated with the preposition and the resulting form becomes ambiguous as to whether or not it is definite. *bth* can be read either as *b+th* "in tea" or as *b+h+th* "in the tea". Thus, 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 exist two main standards for the Hebrew script: one in which vocalization diacritics, known as *niqqud* "dots", decorate the words, and another in which the dots are missing, and other characters represent some, but not all of the vowels. Most of the texts in Hebrew are of the latter kind; unfortunately, different authors use different conventions for the undotted script. Thus, the same word can be written in more than one way, sometimes even within the same document. This fact adds significantly to the degree of ambiguity.

1.2 Hebrew morphological analysis

Shapira and Choueka (1964) were probably the first to construct a computational system for processing Hebrew. This was the introduction to the Responsa project (Choueka, 1980), which included tools for linguistic processing, and in particular algorithms for morphological analysis. Algorithms were developed for automatic generation of "all the possible inflected and derived forms of all the bases in Hebrew", including those obtained by the combination of prepositions, conjunctions, articles etc. Based on the generation algorithm, a file was created which included all the possible Hebrew word forms, approximately 2,500,000 words. The analyzer implements a program which strips the possible affixes off the input word and checks whether the obtained result is indeed a valid form. Later, Choueka (1990) developed a system, called MLIM, which was used as the infrastructure for two major applications: a program for vocalization (adding the vowel diacritics to the unvocalized script of Hebrew) and an on-line dictionary which also includes a morphological analyzer. These applications clearly involve full context-dependent morphological analysis, but as they were integrated in commercial systems, they are not available for research purposes and their performance is difficult to evaluate. See Choueka (1993) for a short description of this system.

A different approach to Hebrew morphology was adopted by Ornan (1986), who developed a *Phonemic Script*, which is an unambiguous writing system for Hebrew, preserving the deep structure of the words. Based on this script, several applications were developed, including a program for vocalization (Ornan, 1985) and programs for morphological analysis and generation (Ornan and Kazatski, 1986). However, converting the standard Hebrew script into Phonemic Script remains a challenge, which was only recently solved in a commercial system (again, unavailable for evaluation).

Another commercial system, Avgad (Bentur et al., 1992), is based on a dictionary of 25,000 entries, which form the base for "hundreds of thousands" of Hebrew words (including inflected forms). The words in the lexicon were selected "out of the common words of Modern Hebrew". The system was used by Segal (1997) in order to construct a freely available morphological analyzer: the analyzer was built by automatically generating possible base forms, inflecting them in all possible ways and verifying the results against the existing analyzer.

Recently, a freely available, wide coverage morphological grammar of Hebrew was developed by Yona and Wintner (Forthcoming), using finite-state technology. The grammar constitutes the core of state of the art morphological analysis and generation systems for Hebrew, HAMSAH, which are regularly maintained

and extended. We use this morphological analyzer for the work reported here, and the examples in sequel are taken directly from its output.

The output HAMSAH on a few examples is illustrated in Figure 1. HAMSAH supports two output formats, a textual string (as in Figure 1) and an XML representation of the same information. For simplicity, we represent this output in tabular format; a few examples are given in Table 1 and Table 2. The rows of each table correspond to different analyses (of a given string), and the columns correspond to *attributes* of the analysis. As can be seen, the output is rather informative and includes the POS as well as sub-category, where applicable, along with several POS-dependent features such as number, gender, tense, nominal status, definitness, etc. Table 3 shows the relevant attributes for each major POS.

```
הרכבת   [+noun][+id]18182[+undotted]הרכבה[+transliterated]hrkbh[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+true
הרכבת   [+verb][+id]19729[+undotted]הרכיב[+transliterated]hrkib[+root]רכב[+binyan]+Hif'il
        [+person/gender/number]+2p/M/Sg[+script]+formal[+tense]+past
הרכבת   [+verb][+id]19729[+undotted]הרכיב[+transliterated]hrkib[+root]רכב[+binyan]+Hif'il
        [+person/gender/number]+2p/F/Sg[+script]+formal[+tense]+past
הרכבת   [+defArt]ה[+noun][+id]18975[+undotted]רכבת[+transliterated]rkbt[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+false

שבתה    [+noun][+id]17280[+undotted]שבת[+transliterated]ebt[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+false[+possessiveSuffix]+3p/F/Sg
שבתה    [+verb][+id]9430[+undotted]שבת[+transliterated]ebt[+root]שבת[+binyan]+Pa'al
        [+person/gender/number]+3p/F/Sg[+script]+formal[+tense]+past
שבתה    [+verb][+id]1541[+undotted]שבה[+transliterated]ebh[+root]שבה[+binyan]+Pa'al
        [+person/gender/number]+3p/F/Sg[+script]+formal[+tense]+past
שבתה    [+subord]ש[+preposition]ב[+noun][+id]19804[+undotted]תה[+transliterated]th
        [+gender]+masculine[+number]+singular[+script]+formal[+construct]+true
שבתה    [+subord]ש[+preposition]ב[+noun][+id]19804[+undotted]תה[+transliterated]th
        [+gender]+masculine[+number]+singular[+script]+formal[+construct]+false
שבתה    [+subord]ש[+preposition]ב[+defArt][+noun][+id]19804[+undotted]תה[+transliterated]th
        [+gender]+masculine[+number]+singular[+script]+formal[+construct]+false
שבתה    [+subord]ש[+noun][+id]19130[+undotted]בתה[+transliterated]bth[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+false
שבתה    [+subord]ש[+noun][+id]1379[+undotted]בת[+transliterated]bt[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+false[+possessiveSuffix]+3p/F/Sg

אתמול   [+adverb][+id]12448[+undotted]אתמול[+transliterated]atmwl
```

Figure 1: The analyses of הרכ בת שבתה אתמ ול

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 part-of-speech (POS) tagging, but is more challenging for languages with rich morphology (see section 2).

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, $A_r \in$

#	Lexical ID	lexeme	POS	Num	Gen	Per	Ten	Stat	Def	Pref	Suf
1	7763	*$mn*	Adj	sing	fem	N/A	N/A	abs	no		
2	7430	*$mn*	noun	sing	masc	N/A	N/A	abs	no		h
3	12311	*mn*	noun	sing	masc	N/A	N/A	abs	no	*$*	h
4	15791	*mnh*	noun	sing	fem	N/A	N/A	abs	no	*$*	
5	5492	*mnh*	verb	sing	masc	3	past	N/A	N/A	*$*	
6	10341	*$m*	verb	plural	fem	2	imperative	N/A	N/A		
7	14568	*$mn*	verb	sing	fem	3	past	N/A	N/A		
8	19796	*$imn*	verb	plural	fem	3	imperative	N/A	N/A		

Table 1: The analyses of the word *$mnh*(שמ נ ה)

#	Lexical ID	lexeme	POS	Num	Gen	Per	Ten	Stat	Def	Pref	Suf
1	17280	*$bt*	noun	sing	fem	N/A	N/A	abs	no		h
2	1379	*bt*	noun	sing	fem	N/A	N/A	abs	no	*$*	h
3	19130	*bth*	noun	sing	fem	N/A	N/A	abs	no	*$*	
4	19804	*th*	noun	sing	masc	N/A	N/A	abs	yes	*$b*	
5	19804	*th*	noun	sing	masc	N/A	N/A	abs	no	*$b*	
6	19804	*th*	noun	sing	masc	N/A	N/A	cons	no	*$b*	
7	1541	*$bh*	verb	sing	fem	3	past	N/A	N/A		
8	9430	*$bt*	verb	sing	fem	3	past	N/A	N/A		

Table 2: The analyses of the word *$bth*(שבתה)

$\{A_1, \ldots A_k\}$, 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:

$$P(A_i|w) = \frac{\textit{number of times } A_i \textit{ is the correct analysis of } w}{\textit{number of occurrences of } w}$$

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, Ornan, and Itai (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 this work cannot in itself be considered morphological disambiguation, due to the complete neglect 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

POS	possible attributes
Adjective	Gender, Number, Status, suffix
Adverb	Gender, Number, Person
AuxVerb	Tense, Gender, Number, Person, suffix
Conjuction	
Interjection	
Interrogative	
Negation	
Noun	Gender, Number, Status, suffix
Particle	Gender, Number, Person
Preposition	Gender, Number, Person, suffix
Pronoun	Gender, Number, Person
Proper name	Gender, Number
Punctuation	
Quantifier	suffix
Verb	Tense, Gender, Number, Person, suffix

Table 3: Part of Speech categories and their possible attributes

morphological information, but essentially not the lexeme. Here, too, contextual information is not taken into account, and the system only prunes relatively unlikely candidates independently 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% (very unlikely analyses are filtered out always, so 100% cannot be achieved) and the ratio of words with a single analysis is 62%. At a threshold of 0.5, accuracy is 86% (74% for ambiguous words) and 100% 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% accuracy. More reliable tests, however, reveal accuracy of 85.5% only (Lemberski, 2003, page 85). Furthermore, the performance of the program is unacceptable (the reported running time on "two papers" is thirty minutes).

The works described above were all hampered by the lack of annotated corpora for Hebrew. Recently, however, such corpora have become available through a treebank project (Sima'an et al., 2001) and the Knowledge Center for Processing Hebrew (Wintner and Yona, 2003), facilitating the application of advanced machine learning techniques to the problem of morphological disambiguation.

A segmenter and a tagger for Hebrew based on Hidden Markov Models (HMMs) is described in Bar-Haim, Simaan, and Winter (2005). This disambiguation model for segmentation and Part-Of-Speech is based on word-segments (morpheme level), rather than on word-level. 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 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 morphemes list back to a full analysis is shown as well. 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 smaller. However, this method lacks some word-level knowledge that is required for segmentation. In this work 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 21 POS tags is used) and for segmentation, and is 90.51% for the former and 96.74% 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 '$mnh' (Table 1). Analyses 5 and 7 differ in gender and lexical ID. In the word '$bth' (Table 2), analyses 5 and 6 differ in status. These two examples are ambiguous in a way that is not solved by the segmentor, since it cannot distinguish between different lexical IDs or the different values of the status attribute. In the work described here we suggest a model that uses more data than can be used in HMM, and can consider all attributes relevant to the disambiguation process.

Another morphological disambiguator is described by Adler and Elhadad (2006). This work presents an unsupervised stochastic model, where the only resource used is a morphological analyzer. 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 (6M words) with fully unsupervised learning are 92.32% for POS tagging and 88.5% for full morphological disambiguation.

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. Habash and Rambow (2005) developed a supervised morphological disambiguator, based on a training corpora of two sets of 120K 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% (depending on the test corpus). One thing to notice, though, is the high accuracy they report for the baseline of each classifier (96.6%-99.9%, depending on the classifier). The baseline of the whole disambiguating process is 87.3%-92.1% (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.

1.4 Machine learning techniques in natural language processing

There is a dramatic increase in research on machine learning methods in natural language processing (NLP) in the last years, using different learning methods to automatically extract linguistic knowledge from natural language corpora which replace or augment those constructed by the linguist. Traditional machine learning uses corpora as data for training and testing, applying mainly statistical and rule-based methods and sometimes a hybrid between the two. Many problems in natural language processing require learning of a mapping from one discrete structure to another. Examples are parsing (learning a function from strings to trees); named entity extraction (learning a function from strings to an underlying segmentation identifying people, places, locations and so on); and machine translation (learning a function from sentences in a source language to sentences in a target language). A very common approach to these tasks is to use some

kind of generative model, for example a hidden Markov model, or history-based models. These approaches are probabilistic, in that they attempt to model a joint or conditional probability distribution over possible structures. There are several challenges in applying these methods to natural language, since NLP problems involve rich structures, and feature spaces are often very high dimensional and very sparse (Collins, 2003).

In many machine learning applications it is necessary to make decisions that depend on the outcomes of several different classifiers in a way that provides a coherent inference that satisfies some constraints: the sequential nature of the data or other domain specific constraints. Consider, for example, the problem of chucking natural language sentences where the goal is to identify several kinds of phrases (e.g., noun phrases and verb phrases) in sentences. A task of this sort involves multiple predictions that interact in some way. One way to address the problem is to use two classifiers for each type of phrases, one of which recognizes the beginning of the phrase, and the other its end. Clearly, there are constraints over the predictions; for instance, phrases cannot overlap and there may also be probabilistic constraints over the order of phrases and over their lengths. The goal is to maximize some global measure of accuracy, not necessarily to maximize the performance of each individual classifier involved in the decision.

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 inter-dependent, 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 (Punyakanok and Roth, 2001; McCallum, Freitag, and Pereira, 2000) and a large number of dynamic programming schemes used in the context of sequential predictions (e.g., Named Entities (Tjong Kim Sang and De Meulder, 2003)) 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.

In this work we use a third class of solutions. We decouple the multi-class classification task into several simple processes and then combine the results using some linguistic knowledge. This approach was shown to be successful for identifying the roots of Semitic words (Daya, Roth, and Wintner, Forthcoming) and 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.

1.5 Research objectives

Identifying the correct morphological analysis of a given word in a given context is an important task. Many applications such as search engines, information retrieval or question answering, as well as heavier applications such as machine translation, require the information that can be extracted from such morphological

analyses. As we show in section 2, morphological disambiguation of Hebrew is a complex, non-trivial task.

The main objective of this work is to devise a novel machine learning paradigm for solving complex morphological problems, and in particular morphological disambiguation of Hebrew (and Arabic). The method we propose is general enough to cope with a variety of other complex morphological problems, in a variety of languages. The resulting paradigm is an innovative technique for decoupling a complex problem into simpler components and combining the components in a sophisticated way guaranteeing higher accuracy.

For humans, the task of choosing the correct analysis in a specific context is simple, even with a short context. Our main goal is to simulate this task by choosing the correct analysis, out of a predefined list of analyses, or in other words to rank the analyses produced by a morphological analyzer. Unlike POS tagging, the task does not involve assigning an analysis to words which the analyzer does not recognize. However, selecting an analysis immediately induces a POS tagging for the target word (by projecting the analysis on the POS coordinate).

2 The challenge

Most POS tagging algorithms are either rule-based or stochastic. Handcrafting a set of rules for Hebrew morphological disambiguation may be a viable solution, but it requires a considerable investment of effort. Furthermore, such a set of rules is task- and language-specific: it cannot be used for Arabic morphological disambiguation, for example. Stochastic taggers use a manually annotated corpus to learn patterns which can then be applied to previously unseen data. The most common and successful techniques for POS tagging in English use Hidden Markov Models (HMM) (Marshall, 1983; Church, 1988), Transformation-Based Learning (Brill, 1995) or more sophisticated learning algorithms (Roth and Zelenko, 1998). With a baseline of $90-91\%$ accuracy, it is not very surprising that the best POS taggers for English achieve an accuracy of $96-97\%$ (Manning and Schütze, 1999, section 10). Crucially, the size of the tagset for English is very small: the Penn Treebank tagset has 45 tags (including punctuation), and other proposed tagsets (e.g., the Brown Corpus one) are all smaller than 100 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 2). It can be segmented in the following ways, each inducing a completely different sequence of POS tags:

$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

Large tagset In contrast to the limited tagset of standard POS tagging approaches in English, the number of different 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 15 different parts of speech, some with subcategories; 6 values for the number feature (including disjunctions of values), 4 for gender, 5 for person, 7 for tense and 3 for nominal status. Possessive pronominal suffixes can have 15 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. While not all the combinations of these values are possible, we estimate the number of possible analyses to be in the thousands.

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% of the tokens were ambiguous (Levinger, Ornan, and Itai, 1995, p. 385). HAMSAH outputs on average approximately 2.4 analyses per word token. 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 2), 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" vs. "her Saturday"). Such phenomena make morphological disambiguation of Hebrew closer to the problem of word sense disambiguation (WSD) than to standard POS tagging.

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

3 Methodology

Our departure point is the output of a morphological analyzer, HAMSAH. For training and evaluation, a corpus of 2,000 sentences (approximately 38,000 word tokens) taken from the daily newspaper HaAretz was automatically analyzed and manually annotated using a dedicated GUI for the human annotator. Annotation consists simply of selecting the correct analysis produced by the analyzer, or an indication that no such analysis exists. Some statistics on this corpus are given in Tables 4 , 5 and 6

In all the experiments described in this paper we use SNoW (Roth, 1998) as the learning environment, with *winnow* as the update rule (running SNoW with perceptron yielded very similar results). 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 already been 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

Number of tokens	37127
Number of types	12570
Distinct lexical entries	4898
Number of tokens with no correct analysis	3165
Number of tokens with no analysis	1121
Degree of ambiguity	2.42
Average number of different inflected forms for lexical ID	2.72

Table 4: Statistics of training corpus

Number of Analyses	1	2	3	4	5	6	7	8	9	10
Number of tokens	15393	6832	5248	3533	2516	1125	503	233	273	126
Number of Analyses	11	12	13	14	15	16				
Number of tokens	86	58	47	21	2	10				

Table 5: Histogram of analyses in the training corpus

of the 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.

3.1 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. 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), 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 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 POS, and a collection of

Part-Of-Speech	noun	verb	preposition	proper name	adjective
Number of tokens	10251	5028	3209	2802	2270
Part-Of-Speech	quantifier	pronoun	adverb	conjunction	negation
Number of tokens	1065	1097	1032	901	461
Part-Of-Speech	interrogative	interjection	punctuation	particle	auxiliary verb
Number of tokens	29	20	5797	0	0

Table 6: Distribution of POS in the training corpus

features such as number, gender, person, tense, status etc. As an example of this strategy, consider again the case of *$bth* (See Table 2), 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. However, this approach has two shortcomings. First, the number of tags remains very high: with seven possible values for the location of the prefix sequence (0-6), approximately 50 tags for the lexeme and two values for the suffix, the tag set has 700 members, probably too high for good accuracy in the face of limited annotated data. Second, this encoding does not suffice to uniquely disambiguate the corpus in the face of analyses which differ only in the lexeme, as we have seen above. We therefore opt for the more complex, yet more promising technique of combining classifiers.

3.2 Architecture

We present the HAifa morphological DisAmbiguation System (HADAS), which consists of several (currently, 10) simple classifiers and a module which combines them. We build a classifier for each of the attributes that can contribute to the disambiguation of the analyses produced by HAMSAH. For example, the word shown in Table 1 can be either a noun, a verb or an adjective, hence a classifier for main POS is required. The same word has two verb readings, one in the past and one in the imperative, hence a *tense* classifier is needed.

Each classifier predicts a small set of possible values (e.g., 15 values for the POS classifier, 8 values for tense), and hence can be highly accurate. In particular, the basic classifiers do not suffer from problems of data sparseness. 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 POS: noun and verb, the POS classifier only ranks those two for this word and gives a score of 0 to the other POS values. In this way, the classifiers induce a ranking on the analyses themselves. For example, ranking the three POS values of the analyses of Table 1 directly induces a ranking of those analyses, where analyses whose main POS is noun (analyses 2, 3 and 4) are ranked the same as analysis #2.

In some cases the analyzer produces no analyses at all for a given word. In those cases we assign to the word a single proper name analysis.

3.3 Evaluation

For evaluation we consider only the words that have at least one correct analysis in the annotated corpus. *Accuracy* is defined as the ratio between the number of words classified correctly by HADAS and the total number of words in the test corpus that have a correct analysis, as annotated by the lexicographers. *Remaining level of ambiguity* is defined as the average number of remaining analyses per word. Remaining analyses are the analyses whose score is equal to the score of the top ranked analysis. Note that remaining level of ambiguity is greater than 1 only for the simple classifiers, where more than one analysis can have the same tag. For the combination of classifiers the remaining level of ambiguity is always 1. For reliability

in all experiments that we describe we perform 10-fold cross-validation runs. We report the average of the 10 runs, both on the entire corpus and on a subset of the corpus in which we only test on words which do *not* occur in the training corpus.

When evaluating the various classifiers and the combination, we compare their performance to a baseline prediction computed as follows. The baseline tag of the token w_i is the most prominent tag of all the occurrences of w_i in the corpus. The baseline for the combination is the most prominent analysis of all the occurrences of w_i in the corpus. If w_i does not occur in the corpus, we back off and select the most prominent tag in the corpus independently of the word w_i. For the combination baseline, we select the analysis of the most prominent lexical ID. This lexical ID is chosen from the list of all possible lexical IDs of w_i. If there is more than one possible value, one top-ranking value is chosen at random.

Formally, let N be the size (in tokens) of the training corpus. For a given token w_i, $1 \leq i \leq N$, let W_i be the number of occurrences of w_i in the training corpus. The morphological analyzer produces k analyses for (each occurrence of) w_i; let A_j^i be the number of occurrences of w_i for which the *j*-th analysis ($1 \leq j \leq k$) is annotated as the correct in context. Let T_j^i be the tag induced by the *j*-th analysis of w_i (this tag is the entire analysis when full morphological is concerned, but only one of its coordinates, such as POS or tense, for the simple classifiers). Let B_j^i be the number of times T_j^i is the correct tag in the training corpus. Let S_j^i be the lexical ID of analysis *j* in w_i. Let C_j^i be the number of occurrences of S_j^i for which the *j*-th analysis ($1 \leq j \leq k$) is annotated as the correct in context.

For each simple classification task, the baseline tag of some token w_i is

$$\hat{t_i} = \begin{cases} T^i_{\hat{j}}, \text{where} \hat{j} = argmax_j A^i_j & \text{if } w_i \text{ occurs in the training corpus} \\ T^i_{\hat{j}}, \text{where} \hat{j} = argmax_j B^i_j & \text{otherwise} \end{cases}$$

For the combined classification task, the baseline tag of some token w_i is

$$\hat{t_i} = \begin{cases} T^i_{\hat{j}}, \text{where} \hat{j} = argmax_j A^i_j & \text{if } w_i \text{ occurs in the training corpus} \\ T^i_{\hat{j}}, \text{where} \hat{j} = argmax_j C^i_j & \text{otherwise} \end{cases}$$

4 Basic Classifiers

As mentioned above, we decouple the task of morphological disambiguation by defining several basic classifiers. We use two types of basic classifiers: *supervised* and *unsupervised*.

4.1 Unsupervised classifiers

This type of classifiers do not rely on the existence of an annotated corpus and are based on statistical data gathered from a large Hebrew corpus. We use this type of classifiers to score a whole analysis. This score is combined with the scores obtained from the supervised classifiers (see section 5.1).

We implemented two unsupervised classifiers. The first is a re-implementation of the HMD algorithm (Carmel and Maarek, 1999), which evaluates the likelihood of each analysis *pattern*, using statistical data which reflect the relative frequency of the morphological patterns in a typical Hebrew sentence. The second classifier approximates the *morpho-lexical probability* of each word, re-implementing Levinger, Ornan, and Itai (1995). See Section 1.3 for more details. The results of replicating those classifiers on our data are depicted in Table 7. Each classifier ranks the analyses of target words, but different analyses may have the same rank, yielding a remaining level of ambiguity. We report the *accuracy*, defined as the ratio between the

number of words whose top-ranking analysis is correct and the total number of words, and the remaining level of ambiguity. The training corpus used for those classifiers is of 400K word tokens.

	accuracy	ambiguity
Levinger	68.22	1.2
Carmel&Maarek	85.55	1.3

Table 7: Results of the unsupervised classifiers

4.2 Supervised classifiers

The supervised classifiers are all built in the same way. They are trained on feature vectors that are generated from the output of the annotated morphological analyzer, and tested on a clean output of the same analyzer. We defined several classifiers for the attributes of the morphological analyses. Note that some attributes do not apply for all the analyses (see Table 3). For example, the attribute 'tense' is inapplicable for an analysis whose main-POS is noun. We therefore add a value of 'irrelevant' for the inapplicable attributes. An annotated corpus was needed in all those classifiers for training. We describe each of the classifiers below.

Part-of-Speech The Part-Of-Speech (POS) classifier predicts the main POS of the word. The possible values of this attribute are: 'noun', 'verb', 'adjective', 'adverb', 'auxiliary verb', 'conjuction', 'interjuction', 'interrogative', 'negation', 'particle', 'preposition', 'pronoun', 'proper name', 'punctuation' and 'quantifier'.

Segmentation The segmentation classifier predicts the index of the beginning of the base. This classifier ranks 7 options [0-6], where 6 is the length of longest possible prefix sequence.

Gender The gender classifier ranks 4 values: 'Masculine', 'Feminine', 'Masculine and feminine' and 'irrelevant'.

Number The number classifier ranks 4 values: 'Singular', 'Plural', 'Dual' and 'irrelevant'.

Person The person classifier ranks 4 values: 'First', 'Second', 'Third' and 'irrelevant'.

Has suffix This is a binary classifier which predicts 'Yes' when the word has a pronominal suffix and 'No' otherwise.

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

Tense The tense classifier ranks 8 values: 'Past', 'Present', 'Participial', 'Future', 'Imperative', 'Infinitive', 'Bare Infinitive' and 'irrelevant'.

Definite Article The definiteness classifier ranks 3 values: 'Definite', 'indefinite' and 'irrelevant'. It identifies both implicit and explicit forms of definiteness.

Status The status classifier ranks 3 values: 'Absolute', 'Construct' and 'irrelevant'.

4.3 Training the supervised classifiers

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

- The main POS (15 options)
- Number (4 options)
- Person (4 options)
- Tense (7 options)
- Status (3 options)
- Definite article (3 options)
- Relevance 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 POS (see Table 3). These features are binary features whose values are 'relevant' and 'irrelevant'. For example, tense is not relevant for an analysis whose POS is noun. The features are 'is the number relevant' (this is the disjunction of all the values of 'Number' 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 (12 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 subsection 1.1, nominals take pronominal suffixes which inflect for number, gender and person. This inflection is combined into a single value (there can be 64 options; 4 Gender $\times$ 4 Number $\times$ 4 Person).
- Does the word have a suffix? (2 options)
- Complex POS (a feature that combines the POS of the prefix, base and suffix) (480 options). This feature represents the full POS of a surface form, and is created from a cartesian product of the 15 values of POS, 4 values of the prefix' POS ('conjuction', '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 (21000 values, the size of our lexicon). To reduce data sparseness, we consider not only the surface form of the word, but also its lemma.
- A conjuction of the surface form and the main-POS.

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 3 words before and after the current word. Another issue is feature generation. It is straight-forward during training, but during evaluation and testing the feature extractor is presented only with the set of analyses produced by HAMSAH for each word, and has no access to the *correct* analysis. For example, when presented with an

instance such as the analyses depicted in Table 1, the feature extractor cannot determine a unique value for the POS feature. We experimented with two methods for tackling this problem. In method A we produce the *union* of all possible values for each feature. In the running example we generate features for POS=noun, POS=verb and POS=adj. In method B we select a single analysis for each word and generate only the features induced by the selected analysis. The analysis that we select is the most prominent one, determined by the same criteria that are used for baseline computation (see section 3.3). While the problem discussed here is manifested only during testing, it impacts also the training procedure. We experimented with three variants of feature generation for training; using only the correct analysis, using all the possible analyses of a given word and using the most prominent for consistency with the training methods. We experimented with all the possible combinations of the above with the feature generation for testing (using all the possible analyses of a given word and using the most prominent).

All the above experiments were performed on the POS classifier (other classifiers produced similar results). As seen in Table 8, the best configuration is using a window of two words before and one word after the current word. Both testing and training are done using the most prominent feature for creating the feature vector.

Training	Testing	1-2	2 - 1	2 - 2	1 - 3	3 - 1	2 - 3	3 - 2	3 - 3
correct	prominent	92.11	92.09	92.27	92.25	92.14	**92.30**	92.27	92.27
correct	all	85.77	85.53	86.28	86.55	86.73	86.90	86.72	**87.15**
all	prominent	**93.35**	93.15	93.32	93.30	93.28	92.61	92.84	91.90
all	all	93.68	**93.82**	93.57	93.38	93.62	93.20	93.14	92.68
prominent	prominent	94.00	**94.05**	93.96	93.82	93.95	93.66	93.66	93.30
prominent	all	88.82	89.07	89.45	89.18	89.69	**89.71**	89.69	89.69

Table 8: Results for the POS classifier, with all the possible configurations

4.4 Intermediate results

Using the training setup described above, the results of all the classifiers are shown in Table 9. We report results on two tasks: one for the entire test corpus and one only for words in the test corpus which were not seen in the training corpus. Resolving the ambiguity of the unseen words shows how well the classifiers really learn, and it is much harder task since none of the feature vectors was seen in the training phase. We show in this table the level of ambiguity remaining after each classifiers predictions (recall from Table 4 that the original level of ambiguity of our test corpus is 2.42 analyses per word.) As opposed to English, where the POS tagger solves the ambiguity, in Hebrew, there are still 1.5 analyses left after using the POS tagger. We also report the reduction in error rate, compared with the baseline described in Section 3.3.

5 Combination of Classifiers

In the previous section we described the basic classifiers that are used in the disambiguation system. We now discuss several ways for combining the predictions of those classifiers to yield a single analysis result.

5.1 Naïve combination

The naïve 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

	All words				Unseen words		
	baseline	classifier			baseline	classifier	
	accuracy	accuracy	ambiguity	E.r.r[a]	accuracy	accuracy	E.r.r
POS	92.89	94.05	1.53	16.3	85.47	89.02	29.8
Segmentation	98.19	95.81	2.09	-	95.83	95.74	-
Has Suffix	99.69	99.36	2.24	-	99.24	98.36	-
Gender	96.04	95.63	1.80	-	91.59	93.27	19.9
Definite Article	93.66	93.14	1.68	-	86.71	85.59	-
Number	96.44	96.85	1.84	11.5	91.87	94.23	29
Person	97.96	97.64	2.09	-	94.73	94.28	-
Status	91.53	92.85	1.54	15.6	82.68	86.65	22.9
Tense	96.35	96.19	1.99	-	91.69	92.02	3.9
Has properties	99.74	99.45	2.36	-	99.77	99.79	8.6

[a]E.r.r. = error rate reduction

Table 9: Results for simple classifiers

those morphological constrains. Each classifier is assigned a weight, which is determined empirically (we experimented with several combinations of weights, but the results were similar to the all equal weights). To combine the predictions of the simple classifiers, we accumulate the weighted scores of all the classifiers. Each classifier predicts a single morphological feature, which induces a rank of the entire list of analyses. For example, consider a word whose six analyses are described in Table 10. Of the six analyses, three have noun as the main POS, two have verb and one is an adjective. The POS classifier generates a score for each analysis as shown in the *score* column. This score induces a partially-ordered rank for the whole list of analyses.

	POS	score	rank
1	noun	0.6	1
2	verb	0.3	2
3	verb	0.3	2
4	noun	0.6	1
5	noun	0.6	1
6	adjective	0.1	3

Table 10: An example score for a word with 6 analyses

In the same way, we can build a table that shows the scores each classifier scored each analysis (example shown in Table 11).

	Classifier 1	Classifier 2	Classifier 3	Combination
Analysis 1	0.25	0.1	...	0.2
Analysis 2	0.3	0.1		0.15
Analysis 3	0.3	0.6		0.6
Analysis 4	0.15	0.1	...	0.05

Table 11: An example of the combination of the classifiers scores

The naïve combination accumulates the weighted scores of all the classifiers, yielding a single score per analysis. This score induces a total rank on the list of analyses. As described in Section 3.3 this analyses scoring does not always imply a single top ranked analysis. In such cases one top-ranking value is chosen at random. The remaining level of ambiguity of the combination is about 1.1 analyses per word. The result of the combination is shown in Table 12.

Weights	All words			Unseen words		
	baseline	classifier	E.r.r	baseline	classifier	E.r.r
equal	85.55	86.87	8.8	85.40	86.80	9.5

Table 12: Results of the Naïve combination

5.2 Hierarchial combination

In the hierarchial combination we try to incorporate more linguistic knowledge pertaining to the dependencies between the classifiers. We use Table 3 to determine which attributes are relevant to which POS. For example, verbs have values for 'tense', 'gender', 'number' and 'person' and can only have 'irrelevant' as the possible value for 'status' and 'definite article'; conjunctions have no attributes with values other than 'irrelevant'. In this method we first predict the main POS of the target word, and take this prediction to be true; we then apply only a subset of the other classifiers, determined by the main POS.

As a pre-processing step we classify the target word to one of two groups: words belonging to POS categories which are atomic, vs. words belonging to POS categories which have additional features. This is done by a dedicated classifier, called *has properties* below, which is trained like all other supervised classifiers. We also compute the predictions of the segmentation and suffix classifiers first. The results of the hierarchical combination are shown in Table 13. As can be seen, the hierarchical combination does not improve on the naïve one. See Algorithm 1.

Algorithm 1 *[Hierarchial combination] For each target word, run the classifiers in the specified order and accumulate the scores generated by the selected classifiers as follows:*

1. *Combine the scores generated by the 'segmentation', 'has suffix' and 'has properties 'classifiers.*
2. *If the prediction of the 'has properties' classifier is 'has no properties', then run the POS classifier, add its score and halt.*
3. *Run the POS classifier and predict the POS.*
4. *Add the scores generated by the gender and person classifiers.*
5. *If the main POS is either 'noun' or 'adjective', then add the scores generated by the definiteness and status classifiers.*
6. *If the main POS is either 'verb' or 'aux verb', then add the scores generated by the person and tense classifiers.*
7. *If the main POS is either 'adverb', 'particle', 'preposition' or 'pronoun', then add the scores generated by the person classifier.*
8. *If the main POS is 'quantifier', then add the score generated by the definiteness classifier.*

All words		Unseen words	
naïve combination	hierarchial combination	naïve combination	hierarchial combination
86.87	86.35	86.80	86.12

Table 13: Results of the hierarchial combination

5.3 Sentence constraints

The combination of independent classifiers under the constraints imposed by the possible morphological analyses is intended to capture context-dependent constraints on possible sequences of analyses. Such constraints are stochastic in nature, but linguistic theory tells us that several *hard* (i.e., deterministic) constraints also exist which rule out certain sequences of otherwise possible analyses. In this section we explore the utility of implementing such constrains to filter out linguistically impossible sequences (we are grateful to Yehoyariv Louck for his help with the research and implementation of this module).

The methodology employed in this module is the following. Using several linguistic sources, we defined a set of constraints, each of which is a linguistically impossible sequence of analyses (all sequences are of length 2, although in principle longer ones could have been defined). We then checked the annotated corpus for violations of these constraints; we used the corpus to either verify the correctness of a constraint or further refine it (or abandon it altogether, in some cases). We then re-iterated the process with the new set of constraints.

The result was a small set of six constraints which are not violated in our annotated corpus; surely more research will yield many more such constraints. We used the constraints to rule out some of the paths defined by the possible outcomes of the morphological analyzer on a sequence of words. Each of the constraints below contributes a non-zero reduction in the error rate of the disambiguation module. In the following, a *nominal phrase* is a sequence of words beginning with either a noun, a proper name, a quantifier, a pronoun, an adjective or a verb in present tense; the first word can be prefixed by 'h' or 'w' only. The (slightly simplified version of the) constraints are:

1. A verb in any tense but present cannot be followed by the genitive preposition '$l' (of).

2. A preposition with no attached pronomial suffix must be followed by a nominal phrase. This rule is relaxed for some prepositions which can be followed by the prefix '$'.

3. The preposition 'at' must be followed by a definite nominal phrase.

4. Construct-state words must be followed by a nominal phrase.

5. A sequence of two verbs is only allowed if: one of them is the verb 'hih' (be); one of them has a prefix; the second is infinitival; or the first is imperative and the second is in future tense.

6. A non-numeral quantifier must be followed by either a nominal phrase or a punctuation.

Imposing the linguistically motivated constraints on the classifier combination improved the results to some extent, as depicted in Table 14. We believe that more research can result in more deterministic rules, improving the disambiguation even further.

All words			Unseen words		
naïve combination	sentence constrains	E.r.r	naïve combination	sentence constrains	E.r.r
86.87	87.27	3	86.80	87.09	2.2

Table 14: Results of the sentence constrains

5.4 Error analysis

HADAS provides a way for estimating the confidence that a specific analysis of a word is indeed correct in a given context. Many factors may influence this confidence. The score of each simple classifier may affect the overall ranking. The combination of the classifiers may also introduce errors. In this section we analyze the errors of both the simple classifiers and the combination task. We refer to the prediction of a classifier as a 'tag', whereas the prediction of the combination is referred to as an 'analysis'.

5.4.1 Errors of simple classifiers

We observed the errors of the classifiers, trying to find patterns in the types of errors. The analysis was done over one fold of the annotated corpus (about 3700 words for testing and 33K words for training). In Table 15 we show some statistics pertaining to the errors made by some of the classifiers (only subsets of the confusion matrices are shown). The table depicts, for each classifier, the *correct* tag, the *chosen*, or predicted, tag, the number of occurrences of the specific error and the total number of errors made by the classifier.

classifier	correct	chosen	occurrences	overall mistakes
has suffix	no	yes	15	20
segmentation	1	0	130	146
has properties	no	yes	30	30
status	construct N/A	absolute absolute	112 80	302
person	third	N/A	37	78
tense	Beinoni	N/A	45	120

Table 15: Classifier's error analysis

Several patterns can be observed in Table 15. The 'has suffix' classifier is correct for more than 99% of the words. When this classifier is wrong, there is a clear bias where 'yes' is chosen instead of 'no' in 75% of the errors. The 'has properties' classifier is even more biased. In 100% of its errors 'yes' is chosen instead of 'no'. The 'segmentation' classifier, which predicts the length of the prefix, has also a clear bias. In almost 90% of its mistakes it predicts the word has no prefix, where it has a prefix of one letter. The 'status' classifier has a less clear type of errors. Although in 63% of the cases this classifier is wrong it predicts 'absolute', the correct tag can be either 'construct' or 'N/A'. The 'person' classifier predicts 'N/A' instead of 'third' in about 50% of the errors.

These types of errors show a clear bias in many of the classifiers. Such biases can be addressed in several ways. Fine-tuning of the features selected for the classifier, along with fine-tuning of the training method can yield an improvement for these classifiers.

Other classifiers have more sporadic types of errors. Table 16 shows (part of) the confusion matrix of the POS classifier. The rows list the correct tags and the columns—the predicted tags. The last column

indicates the total number of times this tag was predicted wrongly. The total number of errors made by the POS classifier is 225.

	noun	adjective	preposition	proper name	verb	total
adjective	20					22
noun		25	10	15	14	71
verb	47	25				81
total	91	54	13	23	19	

Table 16: POS error analysis

Several patterns can be observed for the POS classifier as well. For example, when adjectives are mis-predicted, they are predicted as nouns. This can be explained by the morphological similarity of the two categories, and in particular by the similar syntactic contexts in which they occur. Similarly, $72/81$ of mis-predicted verbs are predicted to be either nouns or adjectives, probably resulting from present-tense verbs in the training corpus which, in Hebrew, have similar distribution to nouns and adjectives. Wrong predictions of nouns are more difficult to analyze. The POS classifier is biased towards predicting adjectives: in 24% of the errors 'adjective was predicted, although the frequency of adjectives in the corpus is only 6%.

Table 17 shows the definiteness classifier's confusion matrix. The total number of errors made by this classifier is 243. As shown in this table, in 58% of its errors, the classifier mis-predicts 'definite' as the correct tag.

	definite	non	N/A	total
definite		92	49	141
non	16		7	23
N/A	67	12		79
	83	104	56	

Table 17: Definiteness error analysis

5.4.2 Errors in the combination

We also observed the types of errors of the combination, trying to find common patterns here as well. As opposed to the basic classifiers, which predict a tag (that may be identical for several possible analyses), the combination predicts only one analysis. Cases where a basic classifier was wrong may be corrected by the other classifiers and not be counted as an error. The opposite situation may occur as well: the combination may predict a wrong analysis, but some of its morphological features may be predicted correctly by one of the classifiers. Table 18 shows the common types of errors of the combination. For each component of the analysis it lists one or two common errors, along with the number of occurrences of the error and the total number of errors for this component. In the one fold tested for error analysis, the combination resulted in 489 errors out of 3700 words tested.

One interesting observation is that while the 'has suffix' classifier is strongly biased toward predicting 'yes' instead of 'no' (Table 15), after the combination the value of this component is overwhelmingly 'no' rather than 'yes': the combination over-corrected this component. Another observation is that in about 56% of the errors the combination yielded an analysis whose status component is wrong, and specifically, the

classifier	correct	chosen	occurrences	overall mistakes
POS	verb	noun	50	225
has suffix	yes	no	31	36
definite	definite N/A	non definite	85 72	233
person	third	N/A	34	73
status	construct N/A	absolute absolute	110 86	278
tense	Beinoni	N/A	45	120

Table 18: Combination's error analysis

wrong status was 'absolute in 70% of those errors. This kind of errors are harder to address, and are left for future work.

6 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, 87.27% accuracy, was obtained using simple supervised clasifiers, for each individual morphological feature. Using the unsupervised classifiers did not improve this result. We have shown that a very few context-dependent constraints on possible sequences of analyses improves the accuracy.

We believe that these results can be further improved in various ways. The use of supervised machine learning methods requires a large annotated corpus for training. In this work we used a very small corpus, of about 30K word tokens. Increasing the size of the training corpus can produce additional accuracy to the basic classifiers, and thus to the disambiguation process. 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.

In another track, there are various ways in which inter-related classifiers can be combined. Here we only used three ways of combining the classifiers. Using other techniques, such as inferenced based training, in which the feature creation for the training is done step by step, using the inferenced information learned in the previous step is likely to yield better accuracy. Improving the combination algorithm already tested here may also improve the results; other weights for the basic classifiers and refining the sentence constraints should improve the accuracy.

References

Adler, Meni and Michael Elhadad. 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*, pages 665–672, Sydney, Australia, July. Association for Computational Linguistics.

Bar-Haim, Roy, Khalil Simaan, and Yoad Winter. 2005. Choosing an optimal architecture for segmentation and pos-tagging of modern hebrew. In *Proceedings of the ACL-2005 Workshop on Computational Approaches to Semitic Languages*.

Bentur, Esther, Aviella Angel, and Danit Segev. 1992. Computerized analysis of Hebrew words. *Hebrew Linguistics*, 36:33–38, December. In Hebrew.

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

Borer, Hagit. 1984. *Parametric Syntax – Case Studies in Semitic and Romance Languages*, volume 13 of *Studies in Generative Grammar*. Foris Publications, Dordrecht, Holland.

Brill, Eric. 1995. Transformation-based error-driven learing and natural language processing: a case study in part-of-speech tagging. *Computational linguistics*, 21(4):543–566.

Carmel, David and Yoelle Maarek. 1999. Morphological disambiguation for Hebrew search systems. In *Proceedings of the 4th international workshop, NGITS-99*, number 1649 in Lecture notes in computer science, pages 312–325. Springer, July.

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

Choueka, Yaacov. 1990. MLIM - a system for full, exact, on-line grammatical analysis of Modern Hebrew. In Yehuda Eizenberg, editor, *Proceedings of the Annual Conference on Computers in Education*, page 63, Tel Aviv, April. In Hebrew.

Choueka, Yaacov. 1993. Response to "Computerized analysis of Hebrew words". *Hebrew Linguistics*, 37:87, December. In Hebrew.

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

Collins, Michael. 2003. Tutorial: Machine learning methods in natural language processing. In *COLT*, page 655.

Daya, Ezra, Dan Roth, and Shuly Wintner. Forthcoming. Learning to identify Semitic roots. In Antal van den Bosch, Guenter Neumann, and Soudi Abdelhadi, editors, *Arabic Computational Morphology: Knowledge- based and Empirical Methods*, Text, Speech, and Language Technology. Springer.

Florian, Radu. 2002. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pages 175–178. Taiwan.

Habash, Nizar and Owen Rambow. 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)*, pages 573–580, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML-01*, pages 282–289.

Lemberski, Gennadiy. 2003. Named entity recognition in Hebrew. Master's thesis, Department of Computer Science, Ben Gurion University, Beer Sheva, Israel, March. In Hebrew.

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

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. The MIT Press, Cambridge, Mass.

Marshall, I. 1983. Choice of grammatical word-class without global syntactic analysis: tagging words in the LOB corpus. *Computers and the Humanities*, 17:139–150.

McCallum, A., D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML-00*, Stanford, CA.

Ornan, Uzzi. 1985. Vocalization by a computer: a linguistic lesson. In Ben-Zion Luria, editor, *Avraham Even-Shoshan Book*. Kiryat-Sefer, Jerusalem, pages 67–76. In Hebrew.

Ornan, Uzzi. 1986. Phonemic script: A central vehicle for processing natural language – the case of Hebrew. Technical Report 88.181, IBM Research Center, Haifa, Israel.

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

Punyakanok, Vasin and Dan Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems 13*, pages 995–1001. MIT Press.

Roth, Dan. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of AAAI-98 and IAAI-98*, pages 806–813, Madison, Wisconsin.

Roth, Dan and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *COLING-ACL 98, The 17th International Conference on Computational Linguistics*, pages 1136–1142.

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

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

Shapira, Meir and Yaacov Choueka. 1964. Mechanographic analysis of Hebrew morphology: possibilities and achievements. *Leshonenu*, 28(4):354–372. In Hebrew.

Sima'an, Khalil, Alon Itai, Yoad Winter, Alon Altman, and N. Nativ. 2001. Building a tree-bank of Modern Hebrew text. *Traitment Automatique des Langues*, 42(2).

Tjong Kim Sang, E. and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL-2003*, pages 142–147.

Wintner, Shuly and Shlomo Yona. 2003. Resources for processing Hebrew. In *Proceedings of the MT-Summit IX workshop on Machine Translation for Semitic Languages*, New Orleans, September.

Yona, Shlomo and Shuly Wintner. Forthcoming. A finite-state morphological grammar of Hebrew. *Natural Language Engineering*.