Morphological Disambiguation of Hebrew: A Case Study in Classifier Combination

Danny Shacham
Department of Computer Science
University of Haifa
Haifa, Israel
dannysh@gmail.com

Shuly Wintner
Department of Computer Science
University of Haifa
Haifa, Israel
shuly@cs.haifa.ac.il

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 which combines them under linguistically motivated constraints. We investigate a number of techniques for combining the predictions of the classifiers. Our best result, 91.44% accuracy, reflects a 25% reduction in error rate compared with the previous state of the art.

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. This work has a twofold contribution: first, our system achieves over 91% accuracy on the full disambiguation task, reducing the error rate of the previous state of the art by 25%. More generally, we explore several ways for combining the predictions of simple classifiers under constraints; the insight gained from these experiments will 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 (Section 3) to predict some components of the analysis, and then combine them in several ways (Section 4) to predict a consistent result. We analyze the errors of the system in Section 5 and conclude with suggestions for future work.

1.1 Linguistic background

Hebrew morphology is rich and complex.[1] 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.

Verbs inflect for number, gender and person (first, second and third) and also for a combination of tense and aspect/mood, referred to simply as ‘tense’ below. Verbs can also take pronominal suffixes, which are interpreted as direct objects, and in some cases can also take nominative pronominal suffixes. A peculiarity of Hebrew verbs is that the participle form

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

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 439–447, Prague, June 2007.

can be used as 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. On top of that, the script dictates that many particles, including four of the most frequent prepositions, the definite article, the coordinating conjunction and some subordinating conjunctions, all attach to the words which immediately follow them. When the definite article *h* is prefixed by one of the prepositions *b*, *k* or *l*, it is assimilated with the preposition and the resulting form becomes ambiguous as to whether or not it is definite. For example, *bth* can be read either as *b+th* "in tea" or as *b+h+th* "in the tea". Thus, the form *$bth* can be read as an inflected stem (the verb "capture", third person singular feminine past), as *$+bth* "that+field", *$+b+th* "that+in+tea", *$+b+h+th* "that in the 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 adds significantly to the degree of ambiguity.

Our departure point in this work is HAMSAH (Yona and Wintner, 2007), a wide coverage, linguistically motivated morphological analyzer of Hebrew, which was recently re-implemented in Java and made available from the Knowledge Center for Processing Hebrew (`http://mila.cs.technion.ac.il/`). The output that HAMSAH produces for the form *$bth* is illustrated in Table 1. In general, it includes the part of speech (POS) as well as sub-category, where applicable, along with several POS-dependent features such as number, gender, tense, nominal state, definitness, etc.

1.2 The challenge of disambiguation

Identifying the correct morphological analysis of a given word in a given context is an important and non-trivial task. 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). Our main contribution in this work is a system that solves this problem with high accuracy.

Compared with POS tagging of English, 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. For example, analysis 4 of Table 1 (*$+b+h+th* "that+in+the+tea") corresponds to the tag sequence IN+IN+DT+NN.

Large tagset 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. HAMSAH produces 22 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 state. Possessive pronominal suffixes can have 15 different values, and prefix particle sequences can theoretically have hundreds of different forms. While not all the combinations of these values are possible, we estimate the number of possible analyses to be in the thousands.

Ambiguity Hebrew is highly ambiguous: HAMSAH outputs on average approximately 2.64 analyses per word token. Oftentimes 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 (see analyses 7 and 8 in Table 1). Morphological disambiguation of Hebrew is hence closer to the problem of word sense disambiguation than to standard POS tagging.

Anchors, which are often function words, are almost always morphologically ambiguous in Hebrew. These include most of the high-frequency forms. 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.

#	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+h*	
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 1: The analyses of the form *$bth*

Word order in Hebrew is freer than in English.

1.3 Related work

The idea of using short context for morphological disambiguation dates back to Choueka and Lusignan (1985). Levinger et al. (1995) were the first to apply it to Hebrew, but their work was hampered by the lack of annotated corpora for training and evaluation. 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).

Bar-Haim et al. (2005) use Hidden Markov Models (HMMs) to implement a segmenter and a tagger for Hebrew. The main innovation of this work is that it models word-segments (morphemes: prefixes, stem and suffixes), rather than full words. The accuracy of this system is 90.51% for POS tagging (a tagset of 21 POS tags is used) and 96.74% for segmentation (which is defined as identifying all prefixes, including a possibly assimilated definite article). As noted above, POS tagging does not amount to full morphological disambiguation.

Recently, Adler and Elhadad (2006) presented an unsupervised, HMM-based model for Hebrew morphological disambiguation, using a morphological analyzer as the only resource. A morpheme-based model learns both segmentation and tagging in parallel from a large (6M words) un-annotated corpus. Reported results are 92.32% for POS tagging and 88.5% for full morphological disambiguation. We refer to this result as the state of the art and use the same data for evaluation.

A supervised approach to morphological disambiguation of *Arabic* is given by Habash and Rambow (2005), who use two corpora of 120K words each to train several classifiers. Each 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). Note, however, the high baseline of each classifier (96.6%-99.9%, depending on the classifier) and the full disambiguation task (87.3%-92.1%, depending on the corpus). We use a very similar approach below, but we experiment with more sophisticated methods for combining simple classifiers to induce a coherent prediction.

2 Methodology

For training and evaluation, we use a corpus of approximately 90,000 word tokens, consisting of newspaper texts, which was automatically analyzed using HAMSAH and then manually annotated (Elhadad et al., 2005). Annotation consists simply of selecting the correct analysis produced by the analyzer, or an indication that no such analysis exists. When the analyzer does not produce the correct analysis, it is added manually. This is the exact setup of the experiments reported by Adler and Elhadad (2006).

Table 2 lists some statistics of the corpus, and a histogram of analyses is given in Table 3. Table 4 lists the distribution of POS in the corpus.

Tokens	89347
Types	23947
Tokens with no correct analysis	8218
Tokens with no analysis	130
Degree of ambiguity	2.64

Table 2: Statistics of training corpus

# analyses	# tokens	# analyses	# tokens
1	38468	7	1977
2	15480	8	1309
3	11194	9	785
4	9934	10	622
5	5341	11	238
6	3472	>12	397

Table 3: Histogram of analyses

POS	# tokens	% tokens
Noun	25836	28.92
Punctuation	13793	15.44
Proper Noun	7238	8.10
Verb	7192	8.05
Preposition	7164	8.02
Adjective	5855	6.55
Participle	3213	3.60
Pronoun	2688	3.01
Adverb	2226	2.49
Conjunction	2021	2.26
Numeral	1972	2.21
Quantifier	951	1.06
Negation	848	0.95
Interrogative	80	0.09
Prefix	29	0.03
Interjection	12	0.01
Foreign	6	0.01
Modal	5	0.01

Table 4: POS frequencies

In all the experiments described in this paper we use SNoW (Roth, 1998) as the learning environment, with *winnow* as the update rule (using *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 the feature space. SNoW has already been used successfully as the learning vehicle in a large collection of natural language related tasks and compared favorably with other classifiers (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 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.

Following Daya et al. (2004) and Habash and Rambow (2005), 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 HAMSAH (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 HAMSAH, but it does induce a ranking on the analyses (see Table 6 below for the level of ambiguity which remains after each simple classifier is applied). Then, we combine the outcomes of the simple classifiers to produce a consistent ranking which induces a linear order on the analyses.

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 and the total number of words in the test corpus that have a correct analysis. The *remaining level of ambiguity* is defined as the average number of analyses per word whose score is equal to the score of the top ranked analysis. This is greater than 1 only for the simple classifiers, where more than one analysis can have the same tag. In all the experiments we perform 10-fold cross-validation runs and 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.

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, 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.

3 Basic Classifiers

The simple classifiers are all built in the same way. They are trained on feature vectors that are generated from the output of the morphological analyzer, and tested on a clean output of the same analyzer. We defined several classifiers for the attributes of the morphological analyses. Since some attributes do not apply to all the analyses, we add a value of 'N/A' for the inapplicable attributes. An annotated corpus was needed in all those classifiers for training. We list the basic classifiers below.

POS 22 values (only 18 in our corpus), see Table 4.

Gender 'Masculine', 'Feminine', 'Masculine and feminine', 'N/A'.

Number 'Singular', 'Plural', 'Dual', 'N/A'.

Person 'First', 'Second', 'Third', 'N/A'.

Tense 'Past', 'Present', 'Participle', 'Future', 'Imperative', 'Infinitive', 'Bare Infinitive', 'N/A'.

Definite Article 'Def', 'indef', 'N/A'. Identifies also implicit (assimilated) definiteness.

Status 'Absolute', 'Construct' and 'N/A'.

Segmentation Predicts the number of letters which are prefix particles. Possible values are [0-6], 6 being the length of longest possible prefix sequence. Does not identify implicit definiteness.

Has properties A binary classifier which distinguishes between atomic POS categories (e.g., conjunction or negation) and categories whose words have attributes (such as nouns or verbs).

Each word in the training corpus induces features that are generated for itself and its immediate neighbors, using the output of the morphological analyzer. For each word in the window, we generate the following features: POS, number, gender, person, tense, state, definiteness, prefixes (where each possible prefix is a binary feature), suffix (binary: is there word suffixed?), number/gender/person of suffix, surface form, lemma, conjunction of the surface form and the POS, conjunction of the POS and the POS of prefixes and suffixes, and some disjunctions of POS. The total number of features for each example is huge (millions), but feature vectors are very sparse.

The simple classifiers can be configured in several ways. First, the size of the window around the target word had to be determined, and we experimented with several sizes, up to ± 3 words. 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. We experimented with two methods for tackling this problem: produce the *union* of all possible values for each feature; or select a single analysis, the baseline one, for each word, and generate only the features induced by this analysis. While this problem is manifested only during testing, it impacts also the training procedure, and so we experimented with feature generation at training using the correct analysis, the union of the analyses or the baseline analysis. The results of the experiments for the POS classifier are shown in Table 5. The best configuration uses a window of two words before and one word after the target word. For both testing and training we generate features using the baseline analysis.

With this setup, the accuracy of all the classifiers is shown in Table 6. We report results on two tasks: the entire test corpus; and words in the test corpus which do not occur in the training corpus, a much harder task. We list the *accuracy*, remaining level of *ambiguity* and reduction in error rate *ERR*, com-

Training	Testing	1 - 2	2 - 1	2 - 2	1 - 3	3 - 1	2 - 3	3 - 2	3 - 3
correct	baseline	91.37	91.53	91.69	91.55	91.69	91.83	91.75	**92.01**
correct	all	79.15	79.55	80.53	80.07	80.13	80.75	81.00	**82.07**
all	baseline	93.41	93.38	93.22	93.42	93.53	93.59	93.51	**93.61**
all	all	93.37	93.42	93.28	93.2	**93.61**	93.05	93.48	93.15
baseline	baseline	94.93	**94.97**	94.8	94.86	94.84	94.72	94.67	94.61
baseline	all	84.48	84.78	84.82	**85.65**	84.97	85.13	85.03	85.45

Table 5: Architectural configurations of the POS classifier: columns reflect the window size, rows refer to training and testing feature generation

	All words				Unseen words		
	baseline	classifier			baseline	classifier	
	accuracy	accuracy	ambiguity	ERR	accuracy	accuracy	ERR
POS	93.01	94.97	1.46	28.04	84.67	88.65	25.96
Gender	96.34	96.74	1.86	10.93	92.15	94.38	28.41
Number	96.79	97.92	1.91	35.20	92.35	95.91	46.54
Person	98.14	98.62	2.25	25.81	94.04	96.50	41.28
Tense	98.40	98.69	2.21	18.12	94.80	96.37	30.19
Definite Article	93.90	95.76	1.83	30.49	85.38	91.77	43.71
Status	92.73	95.06	1.57	32.05	84.46	89.85	34.68
Segmentation	99.12	97.80	2.25	—	97.67	97.66	—
Has properties	97.63	98.11	2.26	20.25	95.91	95.97	1.47

Table 6: Accuracy of the simple classifiers: ERR is reduction in error rate, compared with the baseline

pared with the baseline.

4 Combination of Classifiers

Given a set of simple classifiers, we now investigate various ways for combining their predictions. These predictions may be contradicting (for example, the POS classifier can predict 'noun' while the tense classifier predicts 'past'), and we use the constraints imposed by the morphological analyzer to enforce a consistent analysis.

First, we define a naïve combination along the lines of Habash and Rambow (2005). The scores assigned by the simple classifiers (except segmentation, for which we use the baseline) to each analysis are accumulated, and the score of the complete analysis is their sum (experiments with different weights to the various classifiers proved futile). Even after the combination, the remaining level of ambiguity is 1.05; in ambiguous cases back off to the baseline analysis, and then choose at random one of the top-ranking analyses. The result of the combination is shown in Table 7.

	baseline	classifier	ERR
All words	86.11	**90.26**	29.88
Unseen words	67.53	78.52	33.85

Table 7: Results of the naïve combination

Next, we define a hierarchical combination in which we try to incorporate more linguistic knowledge pertaining to the dependencies between the classifiers. As a pre-processing step we classify the target word to one of two groups, using the *has properties* classifier. Then, we predict the main POS of the target word, and take this prediction to be true; we then apply only the subset of the other classifiers that are relevant to the main POS.

The results of the hierarchical combination are shown in Table 8. As can be seen, the hierarchical combination performs worse than the naïve one. We conjecture that this is because the hierarchical combination does not fully disambiguate, and a random top-ranking analysis is chosen more often than in the case of the naïve combination.

	naïve	hierarchical	ERR
All words	90.26	89.61	—
Unseen words	78.52	78.08	—

Table 8: Results of the hierarchial combination

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* (deterministic) constraints also exist which rule out certain sequences of otherwise possible analyses. We now explore the utility of implementing such constraints to filter out linguistically impossible sequences.

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. 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.The (slightly simplified) 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 9. The best results are obtained when the constraints are applied to the hierarchical combination.

5 Error analysis

We conducted extensive error analysis of both the simple classifiers and the combination module. The analysis was performed over one fold of the annotated corpus (8933 tokens). Table 10 depicts, for some classifiers, a subset of the confusion matrix: it lists 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	#	total
has props	yes	no	110	167
	no	yes	57	
segmentation	1	0	160	176
state	const	abs	154	412
definiteness	def	indef	98	300

Table 10: Simple classifiers, confusion matrix

Several patterns can be observed in Table 10. The 'has properties' classifier is biased towards predicting 'yes' instead of 'no'. The 'segmentation' classifier, which predicts the length of the prefix, also displays a clear bias. In almost 90% of its errors it predicts no prefix instead of a prefix of length one. 'Status' and 'definiteness' are among the weakest classifiers, biased towards the default.

Other classifiers make more sporadic types of errors. Of particular interest is the POS classifier. Here, 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, almost 90% of mis-predicted verbs are predicted to be either nouns

	naïve	naïve + consts	ERR	hier.	hier. + cons	ERR
All words	90.26	90.90	6.57	89.61	**91.44**	17.61
Unseen words	78.52	79.56	4.84	78.08	81.74	16.70

Table 9: Accuracy results of various combination architectures. *ERR* is reduction in error rate due to the hard constraints. The best results are obtained using the hierarchical combination with hard constraints.

or adjectives, probably resulting from present-tense verbs in the training corpus which, in Hebrew, have similar distribution to nouns and adjectives.

The analysis of errors in the combination is more interesting. On the entire corpus, the disambiguator makes 7927 errors. Of those, 1476 (19%) are errors in which the correct analysis differs from the chosen one *only* in the value of the 'state' feature. Furthermore, in 1341 of the errors (17%) the system picks the correct analysis up to the value of 'definiteness'; of those, 1275 (16% of the errors) are words in which the definite article is assimilated in a preposition. In sum, many of the errors seem to be in the real tough cases.

6 Conclusions

Morphological disambiguation of Hebrew is a difficult task which involves, in theory, thousands of possible tags. We reconfirm the results of Daya et al. (2004) and Habash and Rambow (2005), which show that decoupling complex morphological tasks into several simple tasks improves the accuracy of classification. Our best result, 91.44% accuracy, reflects a reduction of 25% in error rate compared to the previous state of the art (Adler and Elhadad, 2006), and almost 40% compared to the baseline. We also show that imposing few context-dependent constraints on possible sequences of analyses improves the accuracy of the disambiguation. The disambiguation module will be made available through the Knowledge Center for Processing Hebrew (`http://mila.cs.technion.ac.il/`).

We believe that these results can be further improved in various ways. The basic classifiers can benefit from more detailed feature engineering and careful tuning of the parameters of the learning environment. There are various ways in which interrelated classifiers can be combined; we only explored three here. Using other techniques, such as inference-based training, in which the feature generation for training is done step by step, using information inferred in the previous step, is likely to yield better accuracy. We also believe that further linguistic exploration, based on deeper error analysis, will result in more hard constraints which can reduce the error rate of the combination module. Finally, we are puzzled by the differences between Hebrew and Arabic (for which the baseline and the current state of the art are significantly higher) on this task. We intend to investigate the linguistic sources for this puzzle in the future.

Acknowledgements

We are extremely grateful to Dan Roth for his continuing support and advise; to Meni Adler for providing the annotated corpus; to Dalia Bojan and Alon Itai for the implementation of the morphological analyzer; to Yariv Louck for the implementation of the deterministic constraints; to Nurit Melnik for help with error analysis; and to Yuval Nardi his help with statistical analysis. Thanks are due to Ido Dagan, Alon Lavie and Michael Elhadad for useful comments and advise. This research was supported by THE ISRAEL SCIENCE FOUNDATION (grant No. 137/06); by the Israel Internet Association; by the Knowledge Center for Processing Hebrew; and by the Caesarea Rothschild Institute for Interdisciplinary Application of Computer Science at the University of Haifa.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 665–672, Sydney, Australia, July. Association for Computational Linguistics.

Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. 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*, pages 39–46, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Yaacov Choueka and Serge Lusignan. 1985. Disambiguation by short context. *Computers and the Humanities*, 19:147–157.

Ezra Daya, Dan Roth, and Shuly Wintner. 2004. Learning Hebrew roots: Machine learning with linguistic constraints. In *Proceedings of EMNLP'04*, pages 357–364, Barcelona, Spain, July.

Michael Elhadad, Yael Netzer, David Gabay, and Meni Adler. 2005. Hebrew morphological tagging guidelines. Technical report, Department of Computer Science, Ben Gurion University.

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

Nizar Habash 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.

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

Moshe Levinger, 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.

Vasin Punyakanok 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.

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

Erel Segal. 1997. Morphological analyzer for unvocalized Hebrew words. Unpublished work.

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

Shlomo Yona and Shuly Wintner. 2007. A finite-state morphological grammar of Hebrew. *Natural Language Engineering*. To appear.