Hebrew Dependency Parsing: Initial Results

Yoav Goldberg and **Michael Elhadad**
Ben Gurion University of the Negev
Department of Computer Science
POB 653 Be'er Sheva, 84105, Israel
{yoavg,elhadad}@cs.bgu.ac.il

Abstract

We describe a newly available Hebrew Dependency Treebank, which is extracted from the Hebrew (constituency) Treebank. We establish some baseline unlabeled dependency parsing performance on Hebrew, based on two state-of-the-art parsers, MST-parser and MaltParser. The evaluation is performed both in an artificial setting, in which the data is assumed to be properly morphologically segmented and POS-tagged, and in a real-world setting, in which the parsing is performed on automatically segmented and POS-tagged text. We present an evaluation measure that takes into account the possibility of incompatible token segmentation between the gold standard and the parsed data. Results indicate that (a) MST-parser performs better on Hebrew data than MaltParser, and (b) both parsers do not make good use of morphological information when parsing Hebrew.

1 Introduction

Hebrew is a Semitic language with rich morphological structure and free constituent order.

Previous computational work addressed unsupervised Hebrew POS tagging and unknown word resolution (Adler, 2007), Hebrew NP-chunking (Goldberg et al., 2006), and Hebrew constituency parsing (Tsarfaty, 2006; Golderg et al., 2009). Here, we focus on Hebrew dependency parsing.

Dependency-parsing got a lot of research attention lately, in part due to two CoNLL shared tasks focusing on multilingual dependency parsing (Buchholz and Erwin, 2006; Nivre et al., 2007). These tasks include relatively many parsing results for Arabic, a Semitic language similar to Hebrew. However, parsing accuracies for Arabic usually lag behind non-semitic languages. Moreover, while there are many published results, we could not find any error analysis or even discussion of the results of Arabic dependency parsing models, or the specific properties of Arabic making it easy or hard to parse in comparison to other languages.

Our aim is to evaluate current state-of-the-art dependency parsers and approaches on Hebrew dependency parsing, to understand some of the difficulties in parsing a Semitic language, and to establish a strong baseline for future work.

We present the first published results on Dependency Parsing of Hebrew.

Some aspects that make Hebrew challenging from a parsing perspective are:

Affixation Common prepositions, conjunctions and articles are prefixed to the following word, and pronominal elements often appear as suffixes. The segmentation of prefixes and suffixes is often ambiguous and must be determined in a specific context only. In term of dependency parsing, this means that the dependency relations occur not between space-delimited tokens, but instead between sub-token elements which we'll refer to as *segments*. Furthermore, any mistakes in the underlying token segmentations are sure to be reflected in the parsing accuracy.

Relatively free constituent order The ordering of constituents inside a phrase is relatively free. This is most notably apparent in the verbal phrases and sentential levels. In particular, while most sentences follow an SVO order, OVS and VSO configurations are also possible. Verbal arguments can appear before or after the verb, and in many ordering. For example, the message "went from Israel to Thailand" can be expressed as "went to Thailand from Israel", "to Thailand went from Israel", "from Israel went to Thailand", "from Israel to Thailand went" and "to Thailand from Israel went". This results in long and flat VP and S structures and a fair amount of sparsity, which suggests

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 129–133,
Paris, October 2009.

that a dependency representations might be more suitable to Hebrew than a constituency one.

Rich templatic morphology Hebrew has a very productive morphological structure, which is based on a root+template system. The productive morphology results in many distinct word forms and a high out-of-vocabulary rate, which makes it hard to reliably estimate lexical parameters from annotated corpora. The root+template system (combined with the unvocalized writing system) makes it hard to guess the morphological analyses of an unknown word based on its prefix and suffix, as usually done in other languages.

Unvocalized writing system Most vowels are not marked in everyday Hebrew text, which results in a very high level of lexical and morphological ambiguity. Some tokens can admit as many as 15 distinct readings, and the average number of possible morphological analyses per token in Hebrew text is 2.7, compared to 1.4 in English (Adler, 2007). This means that on average, every token is ambiguous with respect to its POS and morphological features.

Agreement Hebrew grammar forces morphological agreement between Adjectives and Nouns (which should agree in Gender and Number and definiteness), and between Subjects and Verbs (which should agree in Gender and Number).

2 Hebrew Dependency Treebank

Our experiments are based on the Hebrew Dependency Treebank (henceforth DepTB), which we derived from Version 2 of the Hebrew Constituency Treebank (Guthmann et al., 2009) (henceforth TBv2). We briefly discuss the conversion process and the resulting Treebank:

Parent-child dependencies TBv2 marks several kinds of dependencies, indicating the mother-daughter percolation of features such as number, gender, definiteness and accusativity. See (Guthmann et al., 2009) for the details. We follow TBv2's HEAD, MAJOR and MULTIPLE dependency marking in our-head finding rules. When these markings are not available we use head finding rules in the spirit of Collins. The head-finding rules were developed by Reut Tsarfaty and used in (Tsarfaty and Sima'an, 2008). We slightly extended them to handle previously unhandled cases. Some conventions in TBv2 annotations resulted in bad dependency structures. We identified these constructions and transformed the tree structure,

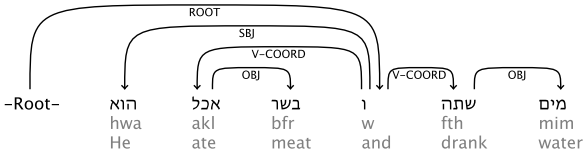

Figure 1: Coordinated Verbs

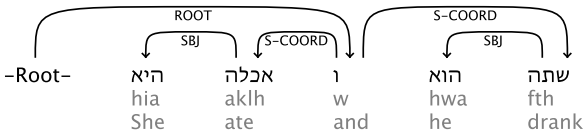

Figure 2: Coordinated Sentence

either manually or automatically, prior to the dependency extraction process.

The conversion process revealed some errors and inconsistencies in TBv2, which we fixed.

We take relativizers as the head S and SBAR, and prepositions as the heads of PPs. In the case the parent of a word X is an empty element, we take the parent of the empty element as the parent of X instead. While this may result in non-projective structures, in practice all but 34 of the resulting trees are projective.

We take conjunctions to be the head of a coordinated structure, resulting in dependency structures such as the one in Figures 1 and 2. Notice how in Figure 1 the parent of the subject "הוא/He" is the coordinator "ו/and", and not one of the verbs. While this makes things harder for the parser, we find this representation to be much cleaner and more expressive than the usual approach in which the first coordinated element is taken as the head of the coordinated structure.[1]

Dependency labels TBv2 marks 3 kinds of functional relations: Subject, Object and Complementizer. We use these in our conversion process, and label dependencies as being SBJ, OBJ or CMP, as indicated in TBv2. We also trivially mark the ROOT dependency, and introduce the relations INF_PREP, AT_INF POS_INF RB_INF between a base word and its suffix for the cases of suffix-inflected prepositions, accusative suffixes, possessive suffixes and inflected-adverbs, respectively. Still, most dependency relations remain unlabeled. We are currently seeking a method of reliably labeling the remaining edges with a rich set

[1] A possible alternative would be to allow multiple parents, as done in (de Marneffe et al., 2006), but current parsing algorithms require the output to be tree structured.

of relations. However, in the current work we focus on the unlabeled dependency structure.

POS tags The Hebrew Treebank follows a syntactic tagging scheme, while other Hebrew resources prefer a more morphological/dictionary-based scheme. For a discussion of these two tagging schemes in the context of parsing, see (Goldberg et al., 2009). In DepTB, we kept the two tagsets, and each token has two POS tags associated with it. However, as current dependency parsers rely on an external POS tagger, we performed all of our experiments only with the morphological tagset, which is what our tagger produces.

3 The Parsing Models

To establish some baseline results for Hebrew dependency parsing, we experiment with two parsing models, the graph-based MST-parser (McDonald, 2006) and the transition-based MaltParser (Nivre et al., 2006). These two parsers represent the current mainstream approaches for dependency parsing, and each was shown to provide state-of-the-art results on many languages (CoNLL Shared Task 2006, 2007).

Briefly, a graph-based parsing model works by assigning a score to every possible attachment between a pair (or a triple, for a second-order model) of words, and then inferring a global tree structure that maximizes the sum of these local scores. Transition-based models work by building the dependency graph in a sequence of steps, where each step is dependent on the next input word(s), the previous decisions, and the current state of the parser. For more details about these parsing models as well as a discussion on the relative benefits of each model, see (McDonald and Nivre, 2007).

Contrary to constituency-based parsers, dependency parsing models expect a morphologically segmented and POS tagged text as input.

4 Experiments

Data We follow the train-test-dev split established in (Tsarfaty and Sima'an, 2008). Specifically, we use Sections 2-12 (sentences 484-5724) of the Hebrew Dependency Treebank as our training set, and report results on parsing the development set, Section 1 (sentences 0-483). We do not evaluate on the test set in this work.

The data in the Treebank is segmented and POS-tagged. All of the models were trained on the gold-standard segmented and tagged data. When evaluating the parsing models, we perform two sets of evaluations. The first one is an oracle experiment, assuming gold segmentation and tagging is available. The second one is a real-world experiment, in which we segment and POS-tag the test-set sentences using the morphological disambiguator described in (Adler, 2007; Goldberg et al., 2008) prior to parsing.

Parsers and parsing models We use the freely available implementation of MaltParser[2] and MSTParser[3], with default settings for each of the parsers.

For MaltParser, we experiment both with the default feature representation (MALT) and the feature representation used for parsing Arabic in CoNLL 2006 and 2007 multilingual dependency parsing shared tasks (MALT-ARA).

For MST parser, we experimented with first-order (MST1) and second-order (MST2) models.

We varied the amount of lexical information available to the parser. Each of the parsers was trained on 3 datasets: LEXFULL, in which all the lexical items are available, LEX20, in which lexical items appearing less than 20 times in the training data were replaced by an OOV token, and LEX100 in which we kept only lexical items appearing more than 100 times in training.

We also wanted to control the effect of the rich morphological information available in Hebrew (gender and number marking, person, and so on). To this end, we trained and tested each model either with all the available morphological information (+MORPH) or without any morphological information (-MORPH).

Evaluation Measure We evaluate the resulting parses in terms of unlabeled accuracy – the percent of correctly identified (child,parent) pairs[4]. To be precise, we calculate:

$$\frac{number\_of\_correctly\_identified\_pairs}{number\_of\_pairs\_in\_gold\_parse}$$

For the oracle case in which the gold-standard token segmentation is available for the parser, this is the same as the traditional unlabeled-accuracy evaluation metric. However, in the real-word setting in which the token segmentation is done automatically, the yields of the gold-standard and the

[2] http://w3.msi.vxu.se/∼jha/maltparser/

[3] http://sourceforge.net/projects/mstparser/

[4] All the results are macro averaged. The micro-averaged numbers are about 2 percents higher for all cases.

	Features	MST1	MST2	MALT	MALT-ARA
-MORPH	Full Lex	83.60	84.31	80.77	80.32
	Lex 20	82.99	84.52	79.69	79.40
	Lex 100	82.56	83.12	78.66	78.56
+MORPH	Full Lex	83.60	84.39	80.77	80.73
	Lex 20	83.60	84.77	79.69	79.84
	Lex 100	83.23	83.80	78.66	78.56

Table 1: Unlabeled dependency accuracy with **oracle** token segmentation and POS-tagging.

	Features	MST1	MST2	MALT	MALT-ARA
-MORPH	Full Lex	75.64	76.38	73.03	72.94
	Lex 20	75.48	76.41	72.04	71.88
	Lex 100	74.97	75.49	70.93	70.73
+MORPH	Full Lex	73.90	74.62	73.03	73.43
	Lex 20	73.56	74.41	72.04	72.30
	Lex 100	72.90	73.78	70.93	70.97

Table 2: Unlabeled dependency accuracy with **automatic** token segmentation and POS-tagging.

automatic parse may differ, and one needs to decide how to handle the cases in which one or more elements in the identified (child,parent) pair are not present in the gold-standard parse. Our evaluation metric penalizes these cases by regarding any such case as a mistake.

5 Results and Analysis

Results are presented in Tables 1 and 2.

It seems that the graph-based parsers perform better than the transitions-based ones. We attribute this to 2 factors: first, our representation of coordinated structure is hard to capture with a greedy local search as performed by a transition-based parser, because we need to defer many attachment decisions until the final coordinator is revealed. The global inference of the graph-based parser is much more robust to these kinds of structure. Indeed, when evaluating the gold-morphology, fully-lexicalized models on a subset of the test-set (314 sentences) which does not have coordinated structures, the accuracy of MALT improves in 3.98% absolute (from 80.77 to 84.75), while MST improves only in 2.66% absolute (from 83.60 to 86.26). Coordination is hard for both parsing models, but more so to the transition based MALT.

Second, it might be hard for a transition-based parser to handle the free constituent order of Hebrew, as it has no means of generalizing from the training set to various possible constituent ordering. The graph-based parser's features and inference method do not take constituent order into account, making it more suitable for free constituent order language.

As expected, the Second-order graph based models perform better than the first-order ones. Surprisingly, the Arabic-optimized feature-set do not perform better than the English one for the transition-based parsers. Overall, morphological information seems to contribute very little (if at all) to any of the parsers in the gold-morphology (oracle) setting. MALTARA gets some benefit from the morphological information in the fully-lexicalized case, while the MST variants benefit from morphology in the lexically-pruned models.

Overall, full lexicalization is not needed. Indeed, less lexicalized LEX20 2nd-order graph-based models perform better than the fully lexicalized ones. This strengthens our intuition that robust lexical statistics are hard to acquire from small annotated corpora, even more so for a language with productive morphology such as Hebrew.

Moving from the oracle morphological disambiguation to an automatic one greatly hurts the performance of all the models. This is in line with results for Hebrew constituency parsing, where going from gold segmentation to a parser derived one caused a similar drop in accuracy (Golderg et al., 2009). This suggests that we should either strive to improve the tagging accuracy, or perform joint inference for parsing and morphological disambiguation. We believe the later would be a better way to go, but it is currently unsupported in state-of-the-art dependency parsing algorithms.

Interestingly, in the automatic morphological disambiguation setting MALTARA benefits a little from the addition of morpological features, while the MST models perform better *without* these features.

6 Conclusions

We presented the first results for unlabeled dependency parsing of Hebrew, with two state-of-the-art dependency parsing models of different families. We experimented both with gold morphological information, and with an automatically derived one. It seems that graph-based models have a slight edge in parsing Hebrew over current transition-based ones. Both model families are not currently making good use of morphological information.

References

Meni Adler. 2007. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach*. Ph.D. thesis, Ben-Gurion University of the Negev, Beer-Sheva, Israel.

Sabine Buchholz and Marsi Erwin. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2006. Noun phrase chunking in hebrew: Influence of lexical and morphological features. In *Proc. of COLING/ACL*.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. EM can find pretty good HMM POS-Taggers (when given a good start). In *Proc. of ACL*.

Yoav Golderg, Reut Tsarfaty, Meni Adler, and Michael Elhadad. 2009. Enhancing unlexicalized parsing performance using a wide coverage lexicon, fuzzy tag-set mapping, and EM-HMM-based lexical probabilities. In *Proc of EACL*.

Noemie Guthmann, Yuval Krymolowski, Adi Milea, and Yoad Winter. 2009. Automatic annotation of morpho-syntactic dependencies in a modern hebrew treebank. In *Proc of TLT*.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proc. of EMNLP*.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Joakim Nivre, Johan Hall, and Jens Nillson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proc. of LREC*.

Joakim Nivre, Johan Hall, Sandra Kubler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the EMNLP-CoNLL*.

Reut Tsarfaty and Khalil Sima'an. 2008. Relational-realizational parsing. In *Proc. of CoLING*, August.

Reut Tsarfaty. 2006. Integrated morphological and syntactic disambiguation for modern hebrew. In *Proceedings of ACL-SRW*.