Joint Hebrew Segmentation and Parsing using a PCFG-LA Lattice Parser

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 experiment with extending a lattice parsing methodology for parsing Hebrew (Goldberg and Tsarfaty, 2008; Golderg et al., 2009) to make use of a stronger syntactic model: the PCFG-LA Berkeley Parser. We show that the methodology is very effective: using a small training set of about 5500 trees, we construct a parser which parses and segments unsegmented Hebrew text with an F-score of almost 80%, an error reduction of over 20% over the best previous result for this task. This result indicates that lattice parsing with the Berkeley parser is an effective methodology for parsing over uncertain inputs.

1 Introduction

Most work on parsing assumes that the lexical items in the yield of a parse tree are fully observed, and correspond to space delimited tokens, perhaps after a deterministic preprocessing step of tokenization. While this is mostly the case for English, the situation is different in languages such as Chinese, in which word boundaries are not marked, and the Semitic languages of Hebrew and Arabic, in which various particles corresponding to function words are agglutinated as affixes to content bearing words, sharing the same space-delimited token. For example, the Hebrew token *bcl*[1] can be interpreted as the single noun meaning "onion", or as a sequence of a preposition and a noun *b-cl* meaning "in (the) shadow". In such languages, the sequence of lexical items corresponding to an input string is ambiguous, and cannot be determined using a deterministic procedure. In this work, we focus on constituency parsing of Modern Hebrew (henceforth Hebrew) from raw unsegmented text.

A common method of approaching the discrepancy between input strings and space delimited tokens is using a pipeline process, in which the input string is pre-segmented prior to handing it to a parser. The shortcoming of this method, as noted by (Tsarfaty, 2006), is that many segmentation decisions cannot be resolved based on local context alone. Rather, they may depend on long distance relations and interact closely with the syntactic structure of the sentence. Thus, segmentation decisions should be integrated into the parsing process and not performed as an independent preprocessing step. Goldberg and Tsarfaty (2008) demonstrated the effectiveness of *lattice parsing* for jointly performing segmentation and parsing of Hebrew text. They experimented with various manual refinements of unlexicalized, treebank-derived grammars, and showed that better grammars contribute to better segmentation accuracies. Goldberg *et al.* (2009) showed that segmentation and parsing accuracies can be further improved by extending the lexical coverage of a lattice-parser using an external resource. Recently, Green and Manning (2010) demonstrated the effectiveness of lattice-parsing for parsing Arabic.

Here, we report the results of experiments coupling lattice parsing together with the currently best grammar learning method: the Berkeley PCFG-LA parser (Petrov et al., 2006).

[1]We adopt here the transliteration scheme of (Sima'an et al., 2001)

2 Aspects of Modern Hebrew

Some aspects that make Hebrew challenging from a language-processing perspective are:

Affixation Common function words are prefixed to the following word. These include: *m*("from") *f*("who"/"that") *h*("the") *w*("and") *k*("like") *l*("to") and *b*("in"). Several such elements may attach together, producing forms such as *wfmhfmf* (*w-f-m-h-fmf* "and-that-from-the-sun"). Notice that the last part of the token, the noun *fmf* ("sun"), when appearing in isolation, can be also interpreted as the sequence *f-mf* ("who moved"). The linear order of such segmental elements within a token is fixed (disallowing the reading *w-f-m-h-f-mf* in the previous example). However, the syntactic relations of these elements with respect to the rest of the sentence is rather free. The relativizer *f*("that") for example may attach to an arbitrarily long relative clause that goes beyond token boundaries. To further complicate matters, the definite article *h*("the") is not realized in writing when following the particles *b*("in"),*k*("like") and *l*("to"). Thus, the form *bbit* can be interpreted as either *b-bit* ("in house") or *b-h-bit* ("in the house"). In addition, pronominal elements may attach to nouns, verbs, adverbs, prepositions and others as suffixes (e.g. *lqxn*(*lqx-hn*, "took-them"), *elihm*(*eli-hm*,"on them")). These affixations result in highly ambiguous token segmentations.

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. This results in long and flat VP and S structures and a fair amount of sparsity.

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 and rich affixation) 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.

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

3 PCFG-LA Grammar Estimation

Klein and Manning (2003) demonstrated that linguistically informed splitting of non-terminal symbols in treebank-derived grammars can result in accurate grammars. Their work triggered investigations in automatic grammar refinement and state-splitting (Matsuzaki et al., 2005; Prescher, 2005), which was then perfected by (Petrov et al., 2006; Petrov, 2009). The model of (Petrov et al., 2006) and its publicly available implementation, the Berkeley parser[2], works by starting with a bare-bones treebank derived grammar and automatically refining it in split-merge-smooth cycles. The learning works by iteratively (1) splitting each non-terminal category in two, (2) merging back non-effective splits and (3) smoothing the split non-terminals toward their shared ancestor. Each of the steps is followed by an EM-based parameter re-estimation. This process allows learning tree annotations which capture many latent syntactic interactions. At inference time, the latent annotations are (approximately) marginalized out, resulting in the (approximate) most probable unannotated tree according to the refined grammar. This parsing methodology is very robust, producing state of the art accuracies for English, as well as many other languages including German (Petrov and Klein, 2008), French (Candito et al., 2009) and Chinese (Huang and Harper, 2009) among others.

The grammar learning process is applied to binarized parse trees, with 1st-order vertical and 0th-order horizontal markovization. This means that in

[2]http://code.google.com/p/berkeleyparser/

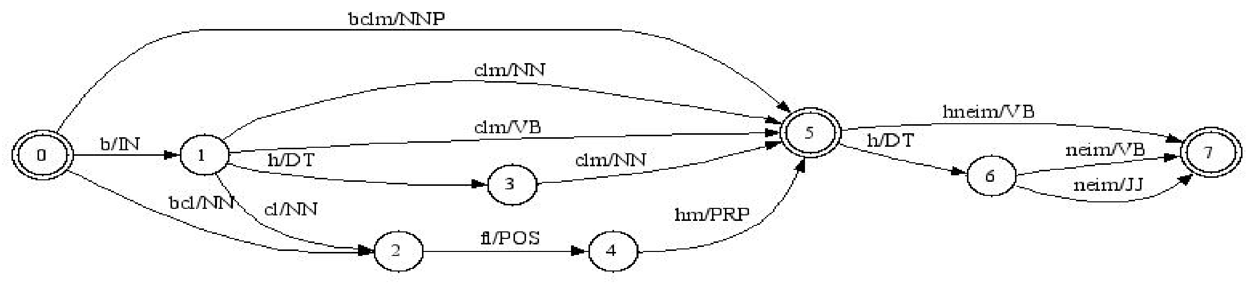

Figure 1: **Lattice representation of the sentence *bclm hneim***. Double-circles denote token boundaries. Lattice arcs correspond to different segments of the token, each lattice path encodes a possible reading of the sentence. Notice how the token *bclm* have analyses which include segments which are not directly present in the unsegmented form, such as the definite article *h* (1-3) and the pronominal suffix which is expanded to the sequence *fl hm* ("of them", 2-4, 4-5).

the initial grammar, each of the non-terminal symbols is effectively conditioned on its parent alone, and is independent of its sisters. This is a very strong independence assumption. However, it allows the resulting refined grammar to encode its own set of dependencies between a node and its sisters, as well as ordering preferences in long, flat rules. Our initial experiments on Hebrew confirm that moving to higher order horizontal markovization degrades parsing performance, while producing much larger grammars.

4 Lattice Representation and Parsing

Following (Goldberg and Tsarfaty, 2008) we deal with the ambiguous affixation patterns in Hebrew by encoding the input sentence as a *segmentation lattice*. Each token is encoded as a lattice representing its possible analyses, and the token-lattices are then concatenated to form the sentence-lattice. Figure 1 presents the lattice for the two token sentence "*bclm hneim*". Each lattice arc correspond to a lexical item.

Lattice Parsing The CKY parsing algorithm can be extended to accept a lattice as its input (Chappelier et al., 1999). This works by indexing lexical items by their start and end states in the lattice instead of by their sentence position, and changing the initialization procedure of CKY to allow terminal and preterminal sybols of spans of sizes > 1. It is then relatively straightforward to modify the parsing mechanism to support this change: not giving special treatments for spans of size 1, and distinguishing lexical items from non-terminals by a specified marking instead of by their position in the chart. We modified the PCFG-LA Berkeley parser to accept lattice input at inference time (training is performed as usual on fully observed treebank trees).

Lattice Construction We construct the token lattices using MILA, a lexicon-based morphological analyzer which provides a set of possible analyses for each token (Itai and Wintner, 2008). While being a high-coverage lexicon, its coverage is not perfect. For the future, we consider using unknown handling techniques such as those proposed in (Adler et al., 2008). Still, the use of the lexicon for lattice construction rather than relying on forms seen in the treebank is essential to achieve parsing accuracy.

Lexical Probabilities Estimation Lexical $p(t \rightarrow w)$ probabilities are defined over individual segments rather than for complete tokens. It is the role of the syntactic model to assign probabilities to contexts which are larger than a single segment. We use the default lexical probability estimation of the Berkeley parser.[3]

Goldberg *et al.* (2009) suggest to estimate lexical probabilities for rare and unseen segments using emission probabilities of an HMM tagger trained using EM on large corpora. Our preliminary experiments with this method with the Berkeley parser

[3]Probabilities for robust segments (lexical items observed 100 times or more in training) are based on the MLE estimates resulting from the EM procedure. Other segments are assigned smoothed probabilities which combine the $p(w|t)$ MLE estimate with unigram tag probabilities. Segments which were not seen in training are assigned a probability based on a single distribution of tags for rare words. Crucially, we restrict each segment to appear only with tags which are licensed by a morphological analyzer, as encoded in the lattice.

showed mixed results. Parsing performance on the test set dropped slightly.When analyzing the parsing results on out-of-treebank text, we observed cases where this estimation method indeed fixed mistakes, and others where it hurt. We are still uncertain if the slight drop in performance over the test set is due to overfitting of the treebank vocabulary, or the inadequacy of the method in general.

5 Experiments and Results

Data In all the experiments we use Ver.2 of the Hebrew treebank (Guthmann et al., 2009), which was converted to use the tagset of the MILA morphological analyzer (Golderg et al., 2009). We use the same splits as in previous work, with a training set of 5240 sentences (484-5724) and a test set of 483 sentences (1-483). During development, we evaluated on a random subset of 100 sentences from the training set. Unless otherwise noted, we used the basic non-terminal categories, without any extended information available in them.

Gold Segmentation and Tagging To assess the adequacy of the Berkeley parser for Hebrew, we performed baseline experiments in which either gold segmentation and tagging or just gold segmentation were available to the parser. The numbers are very high: an F-measure of about 88.8% for the gold segmentation and tagging, and about 82.8% for gold segmentation only. This shows the adequacy of the PCFG-LA methodology for parsing the Hebrew treebank, but also goes to show the highly ambiguous nature of the tagging. Our baseline lattice parsing experiment (without the lexicon) results in an F-score of around 76%.[4]

Segmentation $\rightarrow$ Parsing pipeline As another baseline, we experimented with a pipeline system in which the input text is automatically segmented and tagged using a state-of-the-art HMM pos-tagger (Goldberg et al., 2008). We then ignore the produced tagging, and pass the resulting segmented text as input to the PCFG-LA parsing model as a deterministic input (here the lattice representation is used while tagging, but the parser sees a deterministic, segmented input).[5] In the pipeline setting, we either allow the parser to assign all possible POS-tags, or restrict it to POS-tags licensed by the lexicon.

Lattice Parsing Experiments Our initial lattice parsing experiments with the Berkeley parser were disappointing. The lattice seemed too permissive, allowing the parser to chose weird analyses. Error analysis suggested the parser failed to distinguish among the various kinds of VPs: finite, non-finite and modals. Once we annotate the treebank verbs into finite, non-finite and modals[6], results improve a lot. Further improvement was gained by specifically marking the subject-NPs.[7] The parser was not able to correctly learn these splits on its own, but once they were manually provided it did a very good job utilizing this information.[8] Marking object NPs did not help on their own, and slightly degraded the performance when both subjects and objects were marked. It appears that the learning procedure managed to learn the structure of objects without our help. In all the experiments, the use of the morphological analyzer in producing the lattice was crucial for parsing accuracy.

Results Our final configuration (marking verbal forms and subject-NPs, using the analyzer to construct the lattice and training the parser for 5 iterations) produces remarkable parsing accuracy when parsing from unsegmented text: an F-score of **79.9**% (prec: **82.3** rec: **77.6**) and seg+tagging F of **93.8**%. The pipeline systems with the same grammar achieve substantially lower F-scores of 75.2% (without the lexicon) and 77.3 (with the lexicon). For comparison, the previous best results for parsing Hebrew are 84.1%F assuming gold segmentation and tagging (Tsarfaty and Sima'an, 2010)[9], and 73.7%F starting from unsegmented text (Golderg et

[4]For all the joint segmentation and parsing experiments, we use a generalization of parseval that takes segmentation into account. See (Tsarfaty, 2006) for the exact details.

[5]The segmentation+tagging accuracy of the HMM tagger on the Treebank data is 91.3%F.

[6]This information is available in both the treebank and the morphological analyzer, but we removed it at first. Note that the verb-type distinction is specified only on the pre-terminal level, and not on the phrase-level.

[7]Such markings were removed prior to evaluation.

[8]Candito *et al.* (2009) also report improvements in accuracy when providing the PCFG-LA parser with few manually-devised linguistically-motivated state-splits.

[9]The 84.1 figure is for sentences of length $\leq$ 40, and thus not strictly comparable with all the other numbers in this paper, which are based on the entire test-set.

System	Oracle	OOV Handling	Prec	Rec	F_1
Tsarfaty and Sima'an 2010	Gold Seg+Tag	–	-	-	84.1
Goldberg *et al.* 2009	None	Lexicon	73.4	74.0	73.8
Seg $\rightarrow$ PCFG-LA Pipeline	None	Treebank	75.6	74.8	75.2
Seg $\rightarrow$ PCFG-LA Pipeline	None	Lexicon	79.5	75.2	77.3
PCFG-LA + Lattice (Joint)	None	Lexicon	**82.3**	**77.6**	**79.9**

Table 1: Parsing scores of the various systems

al., 2009). The numbers are summarized in Table 1. While the pipeline system already improves over the previous best results, the lattice-based joint-model improves results even further. Overall, the PCFG-LA+Lattice parser improve results by 6 F-points absolute, an error reduction of about 20%. Tagging accuracies are also remarkable, and constitute state-of-the-art tagging for Hebrew.

The strengths of the system can be attributed to three factors: (1) performing segmentation, tagging and parsing jointly using lattice parsing, (2) relying on an external resource (lexicon / morphological analyzer) instead of on the Treebank to provide lexical coverage and (3) using a strong syntactic model.

Running time The lattice representation effectively results in longer inputs to the parser. It is informative to quantify the effect of the lattice representation on the parsing time, which is cubic in sentence length. The pipeline parser parsed the 483 pre-segmented input sentences in 151 seconds (3.2 sentences/second) not including segmentation time, while the lattice parser took 175 seconds (2.7 sents/second) including lattice construction. Parsing with the lattice representation is slower than in the pipeline setup, but not prohibitively so.

Analysis and Limitations When analyzing the learned grammar, we see that it learned to distinguish short from long constituents, models conjunction parallelism fairly well, and picked up a lot of information regarding the structure of quantities, dates, named and other kinds of NPs. It also learned to reasonably model definiteness, and that S elements have at most one Subject. However, the state-split model exhibits no notion of syntactic agreement on gender and number. This is troubling, as we encountered a fair amount of parsing mistakes which would have been solved if the parser were to use agreement information.

6 Conclusions and Future Work

We demonstrated that the combination of lattice parsing with the PCFG-LA Berkeley parser is highly effective. Lattice parsing allows much needed flexibility in providing input to a parser when the yield of the tree is not known in advance, and the grammar refinement and estimation techniques of the Berkeley parser provide a strong disambiguation component. In this work, we applied the Berkeley+Lattice parser to the challenging task of joint segmentation and parsing of Hebrew text. The result is the first constituency parser which can parse naturally occurring unsegmented Hebrew text with an acceptable accuracy (an F_1 score of 80%).

Many other uses of lattice parsing are possible. These include joint segmentation and parsing of Chinese, empty element prediction (see (Cai et al., 2011) for a successful application), and a principled handling of multiword-expressions, idioms and named-entities. The code of the lattice extension to the Berkeley parser is publicly available.[10]

Despite its strong performance, we observed that the Berkeley parser did not learn morphological agreement patterns. Agreement information could be very useful for disambiguating various constructions in Hebrew and other morphologically rich languages. We plan to address this point in future work.

Acknowledgments

We thank Slav Petrov for making available and answering questions about the code of his parser, Federico Sangati for pointing out some important details regarding the evaluation, and the three anonymous reviewers for their helpful comments. The work is supported by the Lynn and William Frankel Center for Computer Sciences, Ben-Gurion University.

[10]http://www.cs.bgu.ac.il/∼yoavg/software/blatt/

References

Meni Adler, Yoav Goldberg, David Gabay, and Michael Elhadad. 2008. Unsupervised lexicon-based resolution of unknown words for full morphological analysis. In *Proc. of ACL.*

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proc. of ACL (short-paper).*

Marie Candito, Benoit Crabbé, and Djamé Seddah. 2009. On statistical parsing of French with supervised and semi-supervised strategies. In *EACL 2009 Workshop Grammatical inference for Computational Linguistics*, Athens, Greece.

J. Chappelier, M. Rajman, R. Aragues, and A. Rozenknop. 1999. Lattice Parsing for Speech Recognition. In *In Sixth Conference sur le Traitement Automatique du Langage Naturel (TANL99)*, pages 95–104.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proc. of 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.*

Spence Green and Christopher Manning. 2010. Better Arabic parsing: Baselines, evaluations, and analysis. In *Proc. of COLING.*

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.*

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *Proc. of the EMNLP*, pages 832–841. Association for Computational Linguistics.

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

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*, Sapporo, Japan, July. Association for Computational Linguistics.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc of ACL.*

Slav Petrov and Dan Klein. 2008. Parsing German with latent variable grammars. In *Proceedings of the ACL Workshop on Parsing German.*

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of ACL*, Sydney, Australia.

Slav Petrov. 2009. *Coarse-to-Fine Natural Language Processing*. Ph.D. thesis, University of California at Bekeley, Berkeley, CA, USA.

Detlef Prescher. 2005. Inducing head-driven PCFGs with latent heads: Refining a tree-bank grammar for parsing. In *Proc. of ECML.*

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a Tree-Bank of Modern Hebrew text. *Traitement Automatique des Langues*, 42(2).

Reut Tsarfaty and Khalil Sima'an. 2010. Modeling morphosyntactic agreement in constituency-based parsing of Modern Hebrew. In *Proceedings of the NAACL/HLT Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, Los Angeles, CA.

Reut Tsarfaty. 2006. Integrated Morphological and Syntactic Disambiguation for Modern Hebrew. In *Proc. of ACL-SRW.*