Enhancing Unlexicalized Parsing Performance using a Wide Coverage Lexicon, Fuzzy Tag-set Mapping, and EM-HMM-based Lexical Probabilities

Yoav Goldberg[1*] **Reut Tsarfaty**[2†] **Meni Adler**[1‡] **Michael Elhadad**[1]

[1]Department of Computer Science, Ben Gurion University of the Negev
{yoavg|adlerm|elhadad}@cs.bgu.ac.il

[2]Institute for Logic, Language and Computation, University of Amsterdam
R.Tsarfaty@uva.nl

Abstract

We present a framework for interfacing a PCFG parser with lexical information from an external resource following a different tagging scheme than the treebank. This is achieved by defining a stochastic mapping layer between the two resources. Lexical probabilities for rare events are estimated in a semi-supervised manner from a lexicon and large unannotated corpora. We show that this solution greatly enhances the performance of an unlexicalized Hebrew PCFG parser, resulting in state-of-the-art Hebrew parsing results both when a segmentation oracle is assumed, and in a real-word parsing scenario of parsing unsegmented tokens.

1 Introduction

The intuition behind unlexicalized parsers is that the lexicon is mostly separated from the syntax: specific lexical items are mostly irrelevant for accurate parsing, and can be mediated through the use of POS tags and morphological hints. This same intuition also resonates in highly lexicalized formalism such as CCG: while the lexicon categories are very fine grained and syntactic in nature, once the lexical category for a lexical item is determined, the specific lexical form is not taken into any further consideration.

Despite this apparent separation between the lexical and the syntactic levels, both are usually estimated solely from a single treebank. Thus, while PCFGs can be accurate, they suffer from vocabulary coverage problems: treebanks are small and lexicons induced from them are limited.

The reason for this treebank-centric view in PCFG learning is 3-fold: the English treebank is fairly large and English morphology is fairly simple, so that in English, the treebank does provide mostly adequate lexical coverage[1]; Lexicons enumerate analyses, but don't provide probabilities for them; and, most importantly, the treebank and the external lexicon are likely to follow different annotation schemas, reflecting different linguistic perspectives.

On a different vein of research, current POS tagging technology deals with much larger quantities of training data than treebanks can provide, and lexicon-based unsupervised approaches to POS tagging are practically unlimited in the amount of training data they can use. POS taggers rely on richer knowledge than lexical estimates derived from the treebank, have evolved sophisticated strategies to handle OOV and can provide distributions $p(t|w, context)$ instead of "best tag" only.

Can these two worlds be combined? We propose that parsing performance can be greatly improved by using a wide coverage lexicon to suggest analyses for unknown tokens, and estimating the respective lexical probabilities using a semi-supervised technique, based on the training procedure of a lexicon-based HMM POS tagger. For many resources, this approach can be taken only on the proviso that the annotation schemes of the two resources can be aligned.

We take Modern Hebrew parsing as our case study. Hebrew is a Semitic language with rich

[*]Supported by the Lynn and William Frankel Center for Computer Sciences, Ben Gurion University

[†]Funded by the Dutch Science Foundation (NWO), grant number 017.001.271.

[‡]Post-doctoral fellow, Deutsche Telekom labs at Ben Gurion University

[1]This is not the case with other languages, and also not true for English when adaptation scenarios are considered.

morphological structure. This rich structure yields a large number of distinct word forms, resulting in a high OOV rate (Adler et al., 2008a). This poses a serious problem for estimating lexical probabilities from small annotated corpora, such as the Hebrew treebank (Sima'an et al., 2001).

Hebrew has a wide coverage lexicon / morphological-analyzer (henceforth, *KC Analyzer*) available[2], but its tagset is different than the one used by the Hebrew Treebank. These are not mere technical differences, but derive from different perspectives on the data. The Hebrew TB tagset is syntactic in nature, while the KC tagset is lexicographic. This difference in perspective yields different performance for parsers induced from tagged data, and a simple mapping between the two schemes is impossible to define (Sec. 2).

A naive approach for combining the use of the two resources would be to manually re-tag the Treebank with the KC tagset, but we show this approach harms our parser's performance. Instead, we propose a novel, *layered* approach (Sec. 2.1), in which syntactic (TB) tags are viewed as contextual refinements of the lexicon (KC) tags, and conversely, KC tags are viewed as lexical clustering of the syntactic ones. This layered representation allows us to easily integrate the syntactic and the lexicon-based tagsets, without explicitly requiring the Treebank to be re-tagged.

Hebrew parsing is further complicated by the fact that common prepositions, conjunctions and articles are prefixed to the following word and pronominal elements often appear as suffixes. The segmentation of prefixes and suffixes can be ambiguous and must be determined in a specific context only. Thus, *the leaves of the syntactic parse trees do not correspond to space-delimited tokens*, and the yield of the tree is not known in advance.

We show that enhancing the parser with external lexical information is greatly beneficial, both in an artificial scenario where the token segmentation is assumed to be known (Sec. 4), and in a more realistic one in which parsing and segmentation are handled jointly by the parser (Goldberg and Tsarfaty, 2008) (Sec. 5). External lexical information enhances *unlexicalized* parsing performance by as much as 6.67 F-points, an error reduction of 20% over a Treebank-only parser. Our results are not only the best published results for parsing Hebrew, but also on par with state-of-the-art *lexicalized* Arabic parsing results assuming gold-standard fine-grained Part-of-Speech (Maamouri et al., 2008).[3]

[2]http://mila.cs.technion.ac.il/hebrew/resources/lexicons/

2 A Tale of Two Resources

Modern Hebrew has 2 major linguistic resources: the Hebrew Treebank (*TB*), and a wide coverage Lexicon-based morphological analyzer developed and maintained by the Knowledge Center for Processing Hebrew (*KC Analyzer*).

The Hebrew Treebank consists of sentences manually annotated with constituent-based syntactic information. The most recent version (V2) (Guthmann et al., 2009) has 6,219 sentences, and covers 28,349 unique tokens and 17,731 unique segments[4].

The KC Analyzer assigns morphological analyses (prefixes, suffixes, POS, gender, person, etc.) to Hebrew tokens. It is based on a lexicon of roughly 25,000 word lemmas and their inflection patterns. From these, 562,439 unique word forms are derived. These are then prefixed (subject to constraints) by 73 prepositional prefixes.

It is interesting to note that even with these numbers, the Lexicon's coverage is far from complete. Roughly 1,500 unique tokens from the Hebrew Treebank cannot be assigned any analysis by the KC Lexicon, and Adler et al.(2008a) report that roughly 4.5% of the tokens in a 42M tokens corpus of news text are unknown to the Lexicon. For roughly 400 unique cases in the Treebank, the Lexicon provides some analyses, but not a correct one. This goes to emphasize the productive nature of Hebrew morphology, and stress that robust lexical probability estimates cannot be derived from an annotated resource as small as the Treebank.

Lexical vs. Syntactic POS Tags The analyses produced by the KC Analyzer are not compatible with the Hebrew TB.

The KC tagset (Adler et al., 2008b; Netzer et al., 2007; Adler, 2007) takes a lexical approach to POS tagging ("a word can assume only POS tags that would be assigned to it in a dictionary"), while the TB takes a syntactic one ("if the word in this particular positions functions as an Adverb, tag it as an Adverb, even though it is listed in the dictionary only as a Noun"). We present 2 cases that emphasize the difference: **Adjectives:** the Treebank

[3]Our method is orthogonal to lexicalization and can be used in addition to it if one so wishes.

[4]In these counts, all numbers are conflated to one canonical form

treats any word in an adjectivial position as an Adjective. This includes also demonstrative pronouns ילד זה (**this** boy). However, from the KC point of view, the fact that a pronoun can be used to modify a noun does not mean it should appear in a dictionary as an adjective. **The MOD tag:** similarly, the TB has a special POS-tag for words that perform syntactic modification. These are mostly adverbs, but almost any Adjective can, in some circumstances, belong to that class as well. This category is highly syntactic, and does not conform to the lexicon based approach.

In addition, many adverbs and prepositions in Hebrew are lexicalized instances of a preposition followed by a noun (e.g., ברכות, “in+softness”, *softly*). These can admit both the lexicalized and the compositional analyses. Indeed, many words admit the lexicalized analyses in one of the resource but not in the other (e.g., לטובת “for+benefit” is Prep in the TB but only Prep+Noun in the KC, while for מצד “from+side” it is the other way around).

2.1 A Unified Resource

While the syntactic POS tags annotation of the TB is very useful for assigning the correct tree structure when the correct POS tag is known, there are clear benefits to an annotation scheme that can be easily backed by a dictionary.

We created a unified resource, in which every word occurrence in the Hebrew treebank is assigned a KC-based analysis. This was done in a semi-automatic manner – for most cases the mapping could be defined deterministically. The rest (less than a thousand instances) were manually assigned. Some Treebank tokens had no analyses in the KC lexicon, and some others did not have a correct analysis. These were marked as “UNKNOWN” and “MISSING” respectively.[5]

The result is a Treebank which is morphologically annotated according to two different schemas. On average, each of the 257 TB tags is mapped to 2.46 of the 273 KC tags.[6] While this resource can serve as a basis for many linguistically motivated inquiries, the rest of this paper is devoted to using it for constructing a better parser.

Tagsets Comparison In (Adler et al., 2008b), we hypothesized that due to its syntax-based nature, the Treebank morphological tagset is more suitable than the KC one for syntax related tasks. Is this really the case? To verify it, we simulate a scenario in which the complete gold morphological information is available. We train 2 PCFG grammars, one on each tagged version of the Treebank, and test them on the subset of the development set in which every token is completely covered by the KC Analyzer (351 sentences).[7] The input to the parser is the yields and disambiguated pre-terminals of the trees to be parsed. The parsing results are presented in Table 1. Note that this scenario does not reflect actual parsing performance, as the gold information is never available in practice, and surface forms are highly ambiguous.

Tagging Scheme	Precision	Recall
TB / syntactic	82.94	83.59
KC / dictionary	81.39	81.20

Table 1: evalb results for parsing with Oracle morphological information, for the two tagsets

With gold morphological information, the TB tagging scheme is more informative for the parser.

The syntax-oriented annotation scheme of the TB is more informative for parsing than the lexicographic KC scheme. Hence, we would like our parser to use this TB tagset whenever possible, and the KC tagset only for rare or unseen words.

A Layered Representation It seems that learning a treebank PCFG assuming such a different tagset would require a treebank tagged with the alternative annotation scheme. Rather than assuming the existence of such an alternative resource, we present here a novel approach in which we view the different tagsets as corresponding to different aspects of the morphosyntactic representation of pre-terminals in the parse trees. Each of these layers captures subtleties and regularities in the data, none of which we would want to (and sometimes, cannot) reduce to the other. We, therefore, propose to retain both tagsets and learn a *fuzzy mapping* between them.

In practice, we propose an integrated representation of the tree in which the bottommost layer represents the yield of the tree, the surface forms

[5]Another solution would be to add these missing cases to the KC Lexicon. In our view this act is harmful: we don’t want our Lexicon to artificially overfit our annotated corpora.

[6]A “tag” in this context means the complete morphological information available for a morpheme in the Treebank: its part of speech, inflectional features and possessive suffixes, but not prefixes or nominative and accusative suffixes, which are taken to be separate morphemes.

[7]For details of the train/dev splits as well as the grammar, see Section 4.2.

are tagged with dictionary-based KC POS tags, and syntactic TB POS tags are in turn mapped onto the KC ones (see Figure 1).

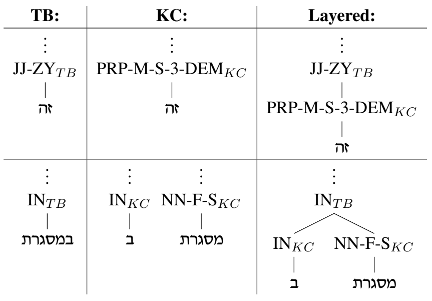

Figure 1: Syntactic (TB), Lexical (KC) and Layered representations

This representation helps to retain the information both for the syntactic and the morphological POS tagsets, and can be seen as capturing the interaction between the morphological and syntactic aspects, allowing for a seamless integration of the two levels of representation. We refer to this intermediate layer of representation as a *morphosyntactic-transfer layer* and we formally depict it as $p(t_{KC}|t_{TB})$.

This layered representation naturally gives rise to a generative model in which a phrase level constituent first generates a syntactic POS tag (t_{TB}), and this in turn generates the lexical POS tag(s) (t_{KC}). The KC tag then ultimately generates the terminal symbols (w). We assume that a morphological analyzer assigns all possible analyses to a given terminal symbol. Our terminal symbols are, therefore, pairs: $\langle w, t\rangle$, and our lexical rules are of the form $t \rightarrow \langle w, t\rangle$. This gives rise to the following equivalence:

$$p(\langle w, t_{KC}\rangle|t_{TB}) = p(t_{KC}|t_{TB})p(\langle w, t_{KC}\rangle|t_{KC})$$

In Sections (4, 5) we use this layered generative process to enable a smooth integration of a PCFG treebank-learned grammar, an external wide-coverage lexicon, and lexical probabilities learned in a semi-supervised manner.

3 Semi-supervised Lexical Probability Estimations

A PCFG parser requires lexical probabilities of the form $p(w|t)$ (Charniak et al., 1996). Such information is not readily available in the lexicon. However, it can be estimated from the lexicon and large *unannotated* corpora, by using the well-known Baum-Welch (EM) algorithm to learn a trigram HMM tagging model of the form $p(t_1, \ldots, t_n, w_1, \ldots, w_n) = argmax \prod p(t_i|t_{i-1}, t_{i-2})p(w_i|t_i)$, and taking the emission probabilities $p(w|t)$ of that model.

In Hebrew, things are more complicated, as each emission w is not a space delimited token, but rather a smaller unit (a morphological segment, henceforth a *segment*). Adler and Elhadad (2006) present a lattice-based modification of the Baum-Welch algorithm to handle this segmentation ambiguity.

Traditionally, such unsupervised EM-trained HMM taggers are thought to be inaccurate, but (Goldberg et al., 2008) showed that by feeding the EM process with sufficiently good initial probabilities, accurate taggers ($> 91\%$ accuracy) can be learned for both English and Hebrew, based on a (possibly incomplete) lexicon and large amount of raw text. They also present a method for automatically obtaining these initial probabilities.

As stated in Section 2, the KC Analyzer (Hebrew Lexicon) coverage is incomplete. Adler et al.(2008a) use the lexicon to learn a Maximum Entropy model for predicting possible analyses for unknown tokens based on their orthography, thus extending the lexicon to cover (even if noisily) any unknown token. In what follows, we use *KC Analyzer* to refer to this extended version.

Finally, these 3 works are combined to create a state-of-the-art POS-tagger and morphological disambiguator for Hebrew (Adler, 2007): initial lexical probabilities are computed based on the MaxEnt-extended KC Lexicon, and are then fed to the modified Baum-Welch algorithm, which is used to fit a morpheme-based tagging model over a very large corpora. Note that the emission probabilities $P(W|T)$ of that model cover all the morphemes seen in the unannotated training corpus, even those not covered by the KC Analyzer.[8]

We hypothesize that such emission probabilities are good estimators for the morpheme-based $P(T \rightarrow W)$ lexical probabilities needed by a PCFG parser. To test this hypothesis, we use it to estimate $p(t_{KC} \rightarrow w)$ in some of our models.

4 Parsing with a Segmentation Oracle

We now turn to describing our first set of experiments, in which we assume the correct segmen-

[8]$P(W|T)$ is defined also for words not seen during training, based on the initial probabilities calculation procedure. For details, see (Adler, 2007).

tation for each input sentence is known. This is a strong assumption, as the segmentation stage is ambiguous, and segmentation information provides very useful morphological hints that greatly constrain the search space of the parser. However, the setting is simpler to understand than the one in which the parser performs both segmentation and POS tagging, and the results show some interesting trends. Moreover, some recent studies on parsing Hebrew, as well as all studies on parsing Arabic, make this oracle assumption. As such, the results serve as an interesting comparison. Note that in real-world parsing situations, the parser is faced with a stream of ambiguous unsegmented tokens, making results in this setting not indicative of real-world parsing performance.

4.1 The Models

The main question we address is the incorporation of an external lexical resource into the parsing process. This is challenging as different resources follow different tagging schemes. One way around it is re-tagging the treebank according to the new tagging scheme. This will serve as a baseline in our experiment. The alternative method uses the Layered Representation described above (Sec. 2.1). We compare the performance of the two approaches, and also compare them against the performance of the original treebank without external information.

We follow the intuition that external lexical resources are needed only when the information contained in the treebank is too sparse. Therefore, we use treebank-derived estimates for reliable events, and resort to the external resources only in the cases of rare or OOV words, for which the treebank distribution is not reliable.

Grammar and Notation For all our experiments, we use the same grammar, and change only the way lexical probabilities are implemented. The grammar is an unlexicalized treebank-estimated PCFG with linguistically motivated state-splits.[9]

In what follows, a lexical event is a word segment which is assigned a single POS thereby functioning as a leaf in a syntactic parse tree. A *rare (lexical) event* is an event occurring less than K times in the training data, and a *reliable (lexical) event* is one occurring at least K times in the training data. We use *OOV* to denote lexical events appearing 0 times in the training data. $count(\cdot)$ is a counting function over the training data, $rare$ stands for any rare event, and w_{rare} is a specific rare event. $KCA(\cdot)$ is the KC Analyzer function, mapping a lexical event to a set of possible tags (analyses) according to the lexicon.

[9]Details of the grammar: all functional information is removed from the non-terminals, finite and non-finite verbs, as well as possessive and other PPs are distinguished, definiteness structure of constituents is marked, and parent annotation is employed. It is the same grammar as described in (Goldberg and Tsarfaty, 2008).

Lexical Models

All our models use relative frequency estimated probabilities for reliable lexical events: $p(t \rightarrow w|t) = \frac{count(w,t)}{count(t)}$. They differ only in their treatment of rare (including OOV) events.

In our **Baseline**, no external resource is used. We smooth for rare and OOV events using a per-tag probability distribution over rare segments, which we estimate using relative frequency over rare segments in the training data: $p(w_{rare}|t) = \frac{count(rare,t)}{count(t)}$. This is the way lexical probabilities in treebank grammars are usually estimated.

We experiment with two flavours of lexical models. In the first, **LexFilter**, the KC Analyzer is consulted for rare events. We estimate rare events using the same per-tag distribution as in the baseline, but use the KC Analyzer to filter out any incompatible cases, that is, we force to 0 the probability of any analysis not supported by the lexicon:

$$p(w_{rare}|t) = \begin{cases} \frac{count(rare,t)}{count(t)} & t \in KCA(w_{rare}) \\ 0 & t \notin KCA(w_{rare}) \end{cases}$$

Our second flavour of lexical models, **LexProbs**, the KC Analyzer is consulted to propose analyses for rare events, and the probability of an analysis is estimated via the HMM emission function described in Section 3, which we denote B: $p(w_{rare}|t) = B(w_{rare}, t)$

In both **LexFilter** and **LexProbs**, we resort to the relative frequency estimation in case the event is not covered in the KC Analyzer.

Tagset Representations

In this work, we are comparing 3 different representations: *TB*, which is the original Treebank, *KC* which is the Treebank converted to use the KC Analyzer tagset, and *Layered*, which is the layered representation described above.

The details of the lexical models vary according to the representation we choose to work with.
For the *TB* setting, our lexical rules are of the form

$t_{tb} \rightarrow w$. Only the **Baseline** models are relevant here, as the tagset is not compatible with that of the external lexicon.

For the *KC* setting, our lexical rules are of the form $t_{kc} \rightarrow w$, and their probabilities are estimated as described above. Note that this setting requires our trees to be tagged with the new (KC) tagset, and parsed sentences are also tagged with this tagset.

For the *Layered* setting, we use lexical rules of the form $t_{tb} \rightarrow w$. Reliable events are estimated as usual, via relative frequency over the original treebank. For rare events, we estimate $p(t_{tb} \rightarrow w|t_{tb}) = p(t_{tb} \rightarrow t_{kc}|t_{tb})p(t_{kc} \rightarrow w|t_{kc})$, where the transfer probabilities $p(t_{tb} \rightarrow t_{kc})$ are estimated via relative frequencies over the layered trees, and the emission probabilities are estimated either based on other rare events (**LexFilter**) or based on the semi-supervised method described in Section 3 (**LexProbs**).

The layered setting has several advantages: First, the resulting trees are all tagged with the original TB tagset. Second, the training procedure does not require a treebank tagged with the KC tagset: Instead of learning the transfer layer from the treebank we could alternatively base our counts on a different parallel resource, estimate it from unannotated data using EM, define it heuristically, or use any other estimation procedure.

4.2 Experiments

We perform all our experiments on Version 2 of the Hebrew Treebank, and follow the train/test/dev split introduced in (Tsarfaty and Sima'an, 2007): section 1 is used for development, sections 2-12 for training, and section 13 is the test set, which we do not use in this work. All the reported results are on the development set.[10] After removal of empty sentences, we have 5241 sentences for training, and 483 for testing. Due to some changes in the Treebank[11], our results are not directly comparable to earlier works. However, our baseline models are very similar to the models presented in, e.g. (Goldberg and Tsarfaty, 2008).

In order to compare the performance of the model on the various tagset representations (TB tags, KC tags, Layered), we remove from the test set 51 sentences in which at least one token is marked as not having any correct segmentation in the KC Analyzer. This introduces a slight bias in favor of the KC-tags setting, and makes the test somewhat easier for all the models. However, it allows for a relatively fair comparison between the various models.[12]

Results and Discussion

Results are presented in Table 2.[13]

Baseline				
	rare: < 2		rare: < 10	
	Prec	Rec	Prec	Rec
TB	72.80	71.70	67.66	64.92
KC	72.23	70.30	67.22	64.31
LexFilter				
	rare: < 2		rare: < 10	
	Prec	Rec	Prec	Rec
KC	77.18	76.31	77.34	76.20
Layered	76.69	76.40	76.66	75.74
LexProbs				
	rare: < 2		rare: < 10	
	Prec	Rec	Prec	Rec
KC	77.29	76.65	77.22	76.36
Layered	76.81	76.49	76.85	76.08

Table 2: evalb results for parsing with a segmentation Oracle.

As expected, all the results are much lower than those with gold fine-grained POS (Table 1).

When not using any external knowledge (**Baseline**), the TB tagset performs slightly better than the converted treebank (KC). Note, however, that the difference is less pronounced than in the gold morphology case. When varying the rare words threshold from 2 to 10, performance drops considerably. Without external knowledge, the parser is facing difficulties coping with unseen events.

The incorporation of an external lexical knowledge in the form of pruning illegal tag assignments for unseen words based on the KC lexicon (**LexFilter**) substantially improves the results ($\sim$ 72 to $\sim$ 77). The additional lexical knowledge clearly improves the parser. Moreover, varying the rare words threshold in this setting hardly affects the parser performance: the external lexicon suffices to guide the parser in the right direction. Keeping the rare words threshold high is desirable, as it reduces overfitting to the treebank vocabulary.

We expected the addition of the semi-supervised $p(t \rightarrow w)$ distribution (**LexProbs**) to improve the parser, but found it to have an insignificant effect. The correct segmentation seems

[10] This work is part of an ongoing work on a parser, and the test set is reserved for final evaluation of the entire system.

[11] Normalization of numbers and percents, correcting of some incorrect trees, etc.

[12] We are forced to remove these sentences because of the artificial setting in which the correct segmentation is given. In the no-oracle setting (Sec. 5), we do include these sentences.

[13] The layered trees have an extra layer of bracketing ($t_{TB} \rightarrow t_{KC}$). We remove this layer prior to evaluation.

to remove enough ambiguity as to let the parser base its decisions on the generic tag distribution for rare events.

In all the settings with a Segmentation Oracle, there is no significant difference between the KC and the Layered representation. We prefer the layered representation as it provides more flexibility, does not require trees tagged with the KC tagset, and produces parse trees with the original TB POS tags at the leaves.

5 Parsing without a Segmentation Oracle

When parsing real world data, correct token segmentation is not known in advance. For methodological reasons, this issue has either been set-aside (Tsarfaty and Sima'an, 2007), or dealt with in a pipeline model in which a morphological disambiguator is run prior to parsing to determine the correct segmentation. However, Tsarfaty (2006) argues that there is a strong interaction between syntax and morphological segmentation, and that the two tasks should be modeled jointly, and not in a pipeline model. Several studies followed this line, (Cohen and Smith, 2007) the most recent of which is Goldberg and Tsarfaty (2008), who presented a model based on unweighted lattice parsing for performing the joint task.

This model uses a morphological analyzer to construct a lattice over all possible morphological analyses of an input sentence. The arcs of the lattice are $\langle w, t\rangle$ pairs, and a lattice parser is used to build a parse over the lattice. The Viterbi parse over the lattice chooses a lattice path, which induces a segmentation over the input sentence. Thus, parsing and segmentation are performed jointly.

Lexical rules in the model are defined over the lattice arcs ($t \rightarrow \langle w, t\rangle | t$), and smoothed probabilities for them are estimated from the treebank via relative frequency over terminal/preterminal pairs. The lattice paths themselves are unweighted, reflecting the intuition that all morphological analyses are a-priori equally likely, and that their perspective strengths should come from the segments they contain and their interaction with the syntax.

Goldberg and Tsarfaty (2008) use a data-driven morphological analyzer derived from the treebank. Their better models incorporated some external lexical knowledge by use of an Hebrew spell checker to prune some illegal segmentations.

In what follows, we use the layered representation to adapt this joint model to use as its morphological analyzer the wide coverage KC Analyzer in enhancement of a data-driven one. Then, we further enhance the model with the semi-supervised lexical probabilities described in Sec 3.

5.1 Model

The model of Goldberg and Tsarfaty (2008) uses a morphological analyzer to constructs a lattice for each input token. Then, the sentence lattice is built by concatenating the individual token lattices. The morphological analyzer used in that work is data driven based on treebank observations, and employs some well crafted heuristics for OOV tokens (for details, see the original paper). Here, we use instead a morphological analyzer which uses the KC Lexicon for rare and OOV tokens.

We begin by adapting the rare vs. reliable events distinction from Section 4 to cover unsegmented tokens. We define a *reliable token* to be a token from the training corpus, which each of its possible segments according to the training corpus was seen in the training corpus at least K times.[14] All other tokens are considered to be rare.

Our morphological analyzer works as follows:
For reliable tokens, it returns the set of analyses seen for this token in the treebank (each analysis is a sequence of pairs of the form $\langle w, t_{TB}\rangle$).
For rare tokens, it returns the set of analyses returned by the KC analyzer (here, analyses are sequences of pairs of the form $\langle w, t_{KC}\rangle$).

The lattice arcs, then, can take two possible forms, either $\langle w, t_{TB}\rangle$ or $\langle w, t_{KC}\rangle$.
Lexical rules of the form $t_{TB} \rightarrow \langle w, t_{TB}\rangle$ are reliable, and their probabilities estimated via relative frequency over events seen in training.
Lexical rules of the form $t_{TB} \rightarrow \langle w, t_{KC}\rangle$ are estimated in accordance with the transfer layer introduced above: $p(t_{TB} \rightarrow \langle w, t_{KC}\rangle) = p(t_{KC}|t_{TB})p(\langle w, t_{KC}\rangle|t_{KC})$.

The remaining question is how to estimate $p(\langle w, t_{KC}\rangle|t_{KC})$. Here, we use either the **LexFilter** (estimated over all rare events) or **LexProbs** (estimated via the semisupervised emission probabilities)models, as defined in Section 4.1 above.

5.2 Experiments

As our **Baseline**, we take the best model of (Goldberg and Tsarfaty, 2008), run against the current

[14]Note that this is more inclusive than requiring that the token itself is seen in the training corpus at least K times, as some segments may be shared by several tokens.

version of the Treebank.[15] This model uses the same grammar as described in Section 4.1 above, and use some external information in the form of a spell-checker wordlist. We compare this Baseline with the **LexFilter** and **LexProbs** models over the Layered representation.

We use the same test/train splits as described in Section 4. Contrary to the Oracle segmentation setting, here we evaluate against all sentences, including those containing tokens for which the KC Analyzer does not contain any correct analyses.

Due to token segmentation ambiguity, the resulting parse yields may be different than the gold ones, and evalb can not be used. Instead, we use the evaluation measure of (Tsarfaty, 2006), also used in (Goldberg and Tsarfaty, 2008), which is an adaptation of parseval to use characters instead of space-delimited tokens as its basic units.

Results and Discussion

Results are presented in Table 3.

	rare: < 2		rare: < 10	
	Prec	Rec	Prec	Rec
Baseline	67.71	66.35	—	—
LexFilter	68.25	69.45	57.72	59.17
LexProbs	73.40	73.99	70.09	73.01

Table 3: Parsing results for the joint parsing+seg task, with varying external knowledge

The results are expectedly lower than with the segmentation Oracle, as the joint task is much harder, but the external lexical information greatly benefits the parser also in the joint setting. While significant, the improvement from the **Baseline** to **LexFilter** is quite small, which is due to the Baseline's own rather strong illegal analyses filtering heuristic. However, unlike the oracle segmentation case, here the semisupervised lexical probabilities (**LexProbs**) have a major effect on the parser performance ($\sim$ 69 to $\sim$ 73.5 F-score), an overall improvement of $\sim$ 6.6 F-points over the Baseline, which is the previous state-of-the art for this joint task. This supports our intuition that rare lexical events are better estimated using a large unannotated corpus, and not using a generic treebank distribution, or sparse treebank based counts, and that lexical probabilities have a crucial role in resolving segmentation ambiguities.

[15]While we use the same software as (Goldberg and Tsarfaty, 2008), the results reported here are significantly lower. This is due to differences in annotation scheme between V1 and V2 of the Hebrew TB

The parsers with the extended lexicon were unable to assign a parse to about 10 of the 483 test sentences. We count them as having 0-Fscore in the table results.[16] The Baseline parser could not assign a parse to more than twice that many sentences, suggesting its lexical pruning heuristic is quite harsh. In fact, the unparsed sentences amount to most of the difference between the **Baseline** and **LexFilter** parsers.

Here, changing the rare tokens threshold has a significant effect on parsing accuracy, which suggests that the segmentation for rare tokens is highly consistent within the corpus. When an unknown token is encountered, a clear bias should be taken toward segmentations that were previously seen in the same corpus. Given that that effect is remedied to some extent by introducing the semi-supervised lexical probabilities, we believe that segmentation accuracy for unseen tokens can be further improved, perhaps using resources such as (Gabay et al., 2008), and techniques for incorporating some document, as opposed to sentence level information, into the parsing process.

6 Conclusions

We present a framework for interfacing a parser with an external lexicon following a different annotation scheme. Unlike other studies (Yang Huang et al., 2005; Szolovits, 2003) in which such interfacing is achieved by a restricted heuristic mapping, we propose a novel, stochastic approach, based on a layered representation. We show that using an external lexicon for dealing with rare lexical events greatly benefits a PCFG parser for Hebrew, and that results can be further improved by the incorporation of lexical probabilities estimated in a semi-supervised manner using a wide-coverage lexicon and a large unannotated corpus. In the future, we plan to integrate this framework with a parsing model that is specifically crafted to cope with morphologically rich, free-word order languages, as proposed in (Tsarfaty and Sima'an, 2008).

Apart from Hebrew, our method is applicable in any setting in which there exist a small treebank and a wide-coverage lexical resource. For example parsing Arabic using the Arabic Treebank and the Buckwalter analyzer, or parsing English biomedical text using a biomedical treebank and the UMLS Specialist Lexicon.

[16]When discarding these sentences from the test set, result on the better LexProbs model leap to 74.95P/75.56R.

References

M. Adler and M. Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proc. of COLING/ACL2006.*

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

Meni Adler, Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2008b. Tagging a hebrew corpus: The case of participles. In *Proc. of LREC 2008.*

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

Eugene Charniak, Glenn Carroll, John Adcock, Anthony Cassandra, Yoshihiko Gotoh, Jeremy Katz, Michael Littman, and John McCann. 1996. Taggers for parsers. *Artif. Intell.*, 85(1-2):45–57.

Shay B. Cohen and Noah A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP-CoNLL-07*, pages 208–217.

David Gabay, Ziv Ben Eliahu, and Michael Elhadad. 2008. Using wikipedia links to construct word segmentation corpora. In *Proc. of the WIKIAI-08 Workshop, AAAI-2008 Conference.*

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

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 2008.*

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.*

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2008. Enhanced annotation and parsing of the arabic treebank. In *INFOS 2008, Cairo, Egypt, March 27-29, 2008.*

Yael Netzer, Meni Adler, David Gabay, and Michael Elhadad. 2007. Can you tag the modal? you should! In *ACL07 Workshop on Computational Approaches to Semitic Languages*, Prague, Czech.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a tree-bank of modern hebrew text. *Traitement Automatique des Langues*, 42(2).

P. Szolovits. 2003. Adding a medical lexicon to an english parser. In *Proc. AMIA 2003 Annual Symposium.*

Reut Tsarfaty and Khalil Sima'an. 2007. Three-dimensional parametrization for parsing morphologically rich languages. In *Proc. of IWPT 2007.*

Reut Tsarfaty and Khalil Sima'an. 2008. Relational-realizational parsing. In *Proc. of CoLING*, pages 889–896, Manchester, UK, August. Coling 2008.

Reut Tsarfaty. 2006. Integrated Morphological and Syntactic Disambiguation for Modern Hebrew. In *Proceedings of ACL-SRW-06.*

MS Yang Huang, MD Henry J. Lowe, PhD Dan Klein, and MS Russell J. Cucina, MD. 2005. Improved identification of noun phrases in clinical radiology reports using a high-performance statistical natural language parser augmented with the umls specialist lexicon. *J Am Med Inform Assoc*, 12(3), May.