

Building a Tree-Bank of Modern Hebrew Text *

K. Sima'an[†] A. Itai[‡] Y. Winter[‡] A. Altman[‡] N. Nativ[§]

Abstract

This paper describes the process of building the first tree-bank for Modern Hebrew text. A major concern in this process is the need for reducing the cost of manual annotation by the use of automatic means. To this end, we explore the joint utility of an automatic morphological analyzer, a probabilistic parser and a small, manually annotated tree-bank.

We describe our initial tree-bank: a newly developed Hebrew annotation scheme which integrates morphological and syntactic analyses, and 500 annotated sentences from a daily newspaper. We discuss the application of a language-independent probabilistic parser to this tree-bank, and report surprisingly encouraging empirical results that are obtained when a combination of the morphological analyzer and the probabilistic parser are applied for the analysis of new text. Based on these results, we describe a semi-automatic procedure for future enlargement of our tree-bank.

1 Introduction

In recent years, the availability of corpora and linguistically annotated language resources such as tree-banks sparked new momentum into the research towards computational solutions for the ambiguity problem in language processing. Tree-banks have been instrumental in acquiring reliable language processing modules, such as part-of-speech taggers, parsers and language models for speech-understanding. Unfortunately, this development has been restricted mostly to languages with large communities or to communities with suitable resources for corpus-collection and tree-bank annotation. The cost of corpus-collection and manual tree-bank annotation is by large the main prohibitive factor. This is the case for, among others, the Modern Hebrew language, for which large annotated (manually corrected) tree-banks are not available.

In this paper we describe an on-going project that develops a tree-bank of Modern Hebrew texts from a daily newspaper. We concentrate on the first step of acquiring an initial environment which allows us to reduce the manual annotation cost by engaging automatic language processing modules in the annotation loop. A serious problem in achieving this goal is the absence of Hebrew syntactic processing modules, i.e. Hebrew parsers. The alternative, general purpose probabilistic parsers, are also not directly applicable in this case, because these demand an initial training tree-bank. To overcome this “deadlock”, we explore the

*This research was supported by a grant no. 120-015 of the Israeli Ministry of Science.

[†]Induction of Linguistic Knowledge, Tilburg University & Computational Linguistics, University of Amsterdam, Spuistraat 134, 1012 VB Amsterdam, The Netherlands. khalil.simaan@hum.uva.nl

[‡]Departement of Computer Science, Technion – Israel Institute of Technology, Haifa, Israel. {itai,winter}@cs.technion.ac.il.

[§]Department of English, Hebrew University, Mount Scopus, Jerusalem, Israel

utility of a small Hebrew tree-bank for enabling further semi-automatic annotation both on the morphological and the syntactic levels. The results of this work are of value for corpus linguistics and corpus based NLP in establishing a strong connection between morphological analysis and parsing, in the development of a morpho-syntactic annotation scheme for Hebrew, and in the new methodology that emerges in the construction of tree-banks using small data sets of manually analyzed material.

Modern Hebrew, as many other languages, has a rich morphology. Hebrew words often consist of more than one morpheme and include elements representing agreement on different features with other words in the same sentence. Therefore, considerable ambiguity exists already at the word-level. This situation is further complicated in Modern Hebrew texts by the use of a writing system that omits various disambiguation clues such as vowels. Hence, morphological analysis is a principal element of a sentence processing system for Modern Hebrew. Therefore, we relied on an existing morphological-analyzer, that was developed using a small set of manually analyzed words, complemented with automatic learning means and heuristic rules [Segal, 2000].

The availability of this analyzer provided us with the first step toward the manual annotation of a small tree-bank of 500 morphologically and syntactically analyzed sentences. The annotation was done by manual correction of the output of the morphological analyzer, followed by a manual annotation at the syntactic level. The result is a small tree-bank featuring a single annotation scheme, where morphology and syntax are integrated into parse-trees. Although more costly to acquire, this relatively rich annotation scheme seems an adequate answer to the strongly related morphology and syntax of Hebrew.

Given a small tree-bank of 500 annotated sentences, the question we face is how to further expand the tree-bank without paying the full cost of manual annotation. The by far largest burden of further tree-bank expansion is the cost of syntactic annotation of sentences. It demands relatively skilled annotators, involves the bulk of the annotation work and requires special attention with respect to the consistency of annotation. Therefore, it is natural to address the need for reducing the cost of further annotation at the syntactic level. However, the question is of course whether this small tree-bank allows the training of a usefully accurate probabilistic parser.

We address the latter question by empirically testing an existing (language independent) probabilistic parser [Sima'an, 2000] on the analysis of new, unseen sentences from the same domain. We report encouraging empirical results of experiments on various settings for future semi-automatic annotation with the help of this parser together with the Hebrew morphological analyzer. The experiments exhibit the positive utility of the small but well-annotated tree-bank as an initial step towards the acquisition of further annotated material.

The structure of this paper is as follows. Section 2 discusses the complexity of morphological analysis of Modern Hebrew texts and the currently available automatic systems for morphological analysis. Section 3 describes the annotation scheme of our small tree-bank, including morphological and syntactic annotations, and summarizes the process of annotation. Section 4 reports on various empirical experiments with applying a probabilistic parsing model to the small tree-bank and the conclusions pertaining to the possibilities for further semi-automatic annotation of new material. Section 5 describes the future paths which can lead us to a larger tree-bank with reduced manual cost. Finally, section 6 summarizes our conclusions from this enterprise.

2 Hebrew morphological analysis - Background

2.1 The problem

Morphological ambiguity is very common in written Modern Hebrew. Most Hebrew texts are unvocalized. This means that the standard Hebrew notation of vowels is totally missing from the text. The rich agreement system of the language, the fact that many affixes (prepositions, determiners, pronominals and conjunctions) are prepended or appended to the word, and the lack of representation of gemination further contribute to the morphological ambiguity. The dimension of the morphological ambiguity in Hebrew is demonstrated in Table 2.1. The data was obtained by running context independent morphological analyzer (see below) on large texts, randomly chosen from the Hebrew press, consisting of 38,898 word-tokens. According to this table, the average number of possible analyses per word-token was 2.1, while 55% of the word-tokens were morphologically ambiguous.

no. of Analyses	1	2	3	4	5	6	7	8	9	10	11	12	13
no. of Word-Tokens	17551	9876	6401	2493	1309	760	337	134	10	18	1	3	5
%	45.1	25.4	16.5	7.1	3.37	1.27	0.87	0.34	0.02	0.05	0.002	0.007	0.01

Table 1: The dimension of morphological ambiguity in Hebrew

A morphological analysis of a word in Hebrew should extract the following information:

- Lexical entry.
- Part of speech.
- Prefixes (conjunctions, prepositions, and the definiteness marker).
- Gender and number.
- Status – a flag indicating whether a noun or adjective is in its construct or absolute form.
- Person (for verbs and pronouns).
- Tense (for verbs only).
- Gender, number and person of pronoun suffixes.

For example, the morphological analysis of the Hebrew string *wkfraiti*¹ (pronounced *ukšera'iti*=“and-when-I-saw”) is as follows:

- Lexical entry: *rah* (the verb ‘to see’).
- Part of speech: verb; gender: feminine or masculine; number: singular; person: first person; tense: past.
- Prefixes: *w+kf* (‘and’+‘when’).

¹In both this paper and in the tree-bank we use a Latin transliteration, in which each Hebrew letter is represented by a single Latin letter. See appendix A for the details of this transliteration.

The above string is unambiguous. But most Hebrew strings have more than one analysis. Consider, for example, the string *lmwrh*, which has (at least) the following possible analyses:

- Preposition *l*=“to” + feminine noun *mwrh*=“teacher”:
“to a female teacher”, pronounced *lemora*.
- Preposition *l*=“to” + masculine noun *mwrh*=“teacher”:
“to a male teacher”, pronounced *lemore*.
- Preposition *l*=“to” + definiteness marker *h* (unvocalized) + feminine noun *mwrh*=“teacher”:
“to the female teacher”, pronounced *lamora*.
- Preposition *l*=“to” + definiteness marker *h* (unvocalized) + masculine noun *mwrh*=“teacher”:
“to the male teacher”, pronounced *lamore*.
- Noun *lmwr*=“*lemur*” + female suffix *h* :
“a female lemur”, pronounced *lemura*.
- Noun *lmwr*=“*lemur*” + possessive feminine suffix *h* :
“her male lemur”, pronounced *lemura*.

2.2 Existing morphological analyzers for Hebrew

Context-independent morphological analyzers A *context-independent* morphological analyzer gets a string and returns the set of all possible morphological analyses of that string, regardless of context. Due to the regularity of Hebrew morphology, a context-independent morphological analyzer, which gives all the possible analyses per word, is feasible. In fact, both academic and commercial analyzers are available.

Among the commercial systems, most noteworthy is the *Rav Millim* analyzer developed by Choueka [Choueka,]. Within a Machine Translation project, IBM Haifa Scientific Center, developed a morphological analyzer [Ben-Tur et al., 1992] that was later used in some commercial products. The sources of these programs are not publicly available. We used a publicly available morphological analyzer written by Erel Segal [Segal, 2000].

Finding the correct analysis Choosing the *correct* analysis in context from the set of all possible analyses is a much more difficult problem, since potentially it requires all levels of linguistic and semantic knowledge. However, several researchers used bounded context to choose the most likely analysis. Choueka and Lusignan [Choueka and Lusignan, 1985] proposed to consider the immediate context of a word and to take advantage of the observation that quite often if a pair of adjacent words appears more than once, the same analysis will be the correct one in all their occurrences. This method leaves open the question of how to choose the correct analysis of the first occurrence of the word.

Orly Albeck [Albeck, 1992] attempted to mimic the way humans analyze text by manually constructing rules that would allow to find the right analysis with no backtracking.

Since there is no large annotated Hebrew corpus, it is impossible to directly estimate the distributions of ambiguous words. The large number of manifestations of each lexical entry introduces a severe sparseness problem, which further aggravates the problem. Levinger et al. [Levinger et al., 1995] gathered statistics on similar words to estimate these distributions. For each analysis of a morphologically ambiguous word they searched a large corpus to find words

(also morphologically ambiguous ones) that differ only in one feature (such as gender) from the original analysis. They assumed that the distributions of the original and the perturbed words were the same. Thus, occurrence statistics of the perturbed words can be used to estimate the distribution of the analyses of the original ambiguous word. In most cases one analysis was much more frequent than all the rest. Choosing this analysis in all cases gives a rough approximation to the correct choice. Levinger [Levinger, 1992] enhanced this system with a short context filter that looked for agreement, and several other local features.

Segal [Segal, 2000] tried to overcome the sparseness problem by assuming that the distribution of the features is independent of the distribution of the lexical entries. Segal therefore estimates the probability of each sequence of features from a large corpus, and the probabilities of lexical entries are estimated using Levinger’s method of “similar words”. The probability of an analysis is the product of the probability of the sequence of features and the probability of the lexical entry. Choosing the most probable analysis as the correct analysis yields a 14% error. To reduce the error rate, Segal employed two more heuristics:

1. Correction rules are automatically devised by a method resembling Brill’s transformation based method [Brill, 1995]. The program considers three types of rules: rules that apply to a pair of syntactic categories, such as “change an undetermined noun followed by a determined adjective to a determined noun followed by the adjective”; rules that apply to a syntactic category and a single word; and rules that apply to pairs of words. The program considers all possible rules and adopts those rules that improved the performance of the morphological analyzer on the training set.
2. Segal uses a rudimentary deterministic parser that uses some lexical dependency rules. This part of the system is completely heuristic and was not acquired from the corpus.

For each morphological analysis of a word the system assigns a score, which is initialized to the probabilities given by Segal’s version of the similar words method. The two heuristics are then employed to modify the score. The analysis with largest score is chosen as the correct analysis. Segal reports on tests according to which, with these heuristics, the analyzer finds the correct analysis of 96% of the words of test data. As mentioned below, the morphological analysis of a word by the analyzer often corresponds to more than one sequence of part of speech tags, which means that in terms of standard POS tagging, the actual precision of the morphological analyzer is somewhat reduced. In our work we did not attempt to evaluate Segal’s results, but his analyzer proved useful in providing an initial POS tagging for the words in the corpus.

3 Building a Hebrew tree-bank

Most current work on probabilistic models of syntactic analysis is devoted to English (see [Manning and Schütze, 1999] for some references), for which large tree-banks exist, e.g., the Penn Tree-Bank [Marcus et al., 1993]. In this kind of work, morphological analysis has been of minor concern: actual word-occurrences and their POS tags, instead of morphemes, are utilized in probabilistic language models. Impoverished forms of morphological analysis are often consulted in order to deal with unknown words (see [Collins, 1997, Charniak, 1999]). Clearly, statistics over word-occurrence is more prone to sparse-data problems than morpheme occurrence. However, for English, the existence of large syntactic tree-banks, extended POS

tag-sets and the less complex nature of English word-structure, have been instrumental in avoiding the need for the level of morphology in the tree-banks.

Corpora of morphologically rich languages are currently developed for Czech ([Hajic and Hladk, 1998, Bemova et al., 1999]), Turkish ([Tur et al., 1999]) and Japanese [Kurohashi and Nagao, 1998]. Unlike English, in these languages it is much harder to ignore the morphological level in the construction of the tree-bank, because the word level is often not fine-grained enough for the syntactic analysis. Hebrew is another example for such a morphologically rich language, which is substantially different from the other language mentioned above, but there are currently no tree-banks for this language. Building such a tree-bank is an important step in facilitating future work on models of Hebrew sentence processing, and as a test case for combining corpus based techniques for morphological analysis and syntactic analysis.

As discussed in the preceding section, morphological ambiguity is very common in Hebrew texts, and syntactically important morphemes like prepositions, conjunctions and pronouns, as well as many agreement features, appear as word affixes. Therefore, the syntactic annotation of the corpus must involve word-segmentation into morphemes, morpheme POS tagging and feature annotation. In our tree-bank, words are analyzed as strings of morphemes, where each morpheme is given a part of speech tag. Because of the lack of vocalization in Hebrew written texts, it often happens that the representation of a morpheme does not appear as a part of the word, and consequently the concatenation of morphemes in the morphological analysis of a word is not identical with the word itself. For instance, as we have seen, a definiteness marker after a preposition is often not vocalized in modern Hebrew texts. For instance, as we have seen, both Hebrew words *lamora* (“to-the-female-teacher”) and *lemora* (“to-a-female-teacher”) are spelled as *lmwrh*. However, the morphemes in the analysis of the first reading are *l-h-mwrh*, where the morpheme ‘*h*’, which represents definiteness, does not appear in the original string.

The tree-bank itself is based on 500 sentences from articles in *Ha’aretz* daily newspaper. This text was chosen because it represents a fairly standard example of written modern Hebrew. The complexity of the syntactic structures in this corpus and the average length of its sentences resemble those of the *Wall Street Journal* corpus, and therefore make it possible to compare the performance of similar parsing models on the Hebrew tree-bank to the much studied Penn tree-bank. In this section we describe the annotation scheme of the corpus, including the POS tag-set, the syntactic tag-set and bracketing methodology. Then we describe the construction process of the tree-bank and give some figures on its structure. The tree-bank itself is available at the following URL:

<http://www.cs.technion.ac.il/~winter/Corpus-Project/Hebrew-Treebank.adj>

3.1 POS tag-set

We have tried to keep as close as possible to the English tag-set used by the Penn tree-bank. However, Hebrew is much richer than English in morphological marking of agreement features. Consequently, for many POS tags there are additional agreement features for gender (g), number (n), person (p) or tense (t), with values as specified in table 2. In addition, tags for nouns, adjectives and numerals may have an ‘H’ marking, which indicates morphemes that are inherently definite.

The decision of whether to represent an affix using a separate morpheme or using a feature is not always straightforward, and involves considerations of consistency and ease of

g (gender):	Z=male, N=female, B=both
n (number):	Y=singular, R=plural, B=both
p (person):	1,2,3
t (tense):	V=past, H=present, T=future, C=imperative

Table 2: agreement features

annotation. For instance, the word *bo*, spelled *bw*, is a prepositional phrase with the meaning “in-him”. For reason of consistency in the tagging of prepositional phrases, where a full noun phrase is normally present, this word was analyzed as consisting of the two morphemes *b* (“in”) and *hu* (spelled *hwa*=“he”). However, other affixes, like the genitive morpheme *o* in *toxnito* (spelled *tknitw=plan-he*=“his plan”), are tagged as features of the morpheme. The reason for this choice are more complex genitive constructions like the following, which are given the specified part of speech tags.

- (1) tknitw/NN-NY-H-ZY3 fl/POS rz/NNP-ZY
 plan-he of Raz
 “Raz’s plan”

For linguistic arguments supporting this analysis, see [Engelhardt, 1999] (but compare with [Borer, 1984]).

Besides the addition of features, other modifications in the Penn tag-set were motivated by phenomena or categories that are special to Hebrew, or by cases where the distinctions of the Penn tag-set are insufficient for our purposes.

The POS tags we added to the tag-set of the Penn tree-bank are the following:

1. **AGR,AUX**: On the use of these two POS tags see the discussion below of the predicative construction and the syntactic tag PREDP.
2. **AT**: A special tag for the accusative marker *ʔet* (spelled *at*) which appears as a separate word in Hebrew. For example:

 finh/VN-ZY3V at/AT h/H mdiniwt/NN-NY
 changed ACC the policy
3. **CDT**: Numerals in determiner position, which are distinguished from appearances of cardinal numbers as in dates or figures, which are classified as CD.
4. **COM**: Complementizers, including *ki* and *še* (“that”), which in Hebrew are systematically distinguished from prepositions.
5. **JJT**: The construct state form of adjectives. For instance, the following phrase has the adjectival meaning “pitiless”, where the word *xsrt*, meaning “-less” or “missing”, is a construct state adjective, which, due to its appearance in a compound, is different in its form from the absolute entry of the adjective (*xsrh*).

xsrt/JJT-NY rxmin/NN-ZR
 missing pity

6. **H**: A special tag assigned to the definiteness marker *h*, which appears with nouns, adjectives and numerals.
7. **HAM**: A special tag for the complementizer/yes-no question word *ha?im* (“whether”).
8. **MD**: A tag for the class of “modal” verbs in Hebrew – verbs that subcategorize for an infinitival complement. Examples: *heci’a* (“proposed”), *hiskim* (“agreed”).
9. **MOD**: Assigned to modificational elements like *rq* (= *rak* = “only”) and *gm* (= *gam* = “also”) and to nominal prefixes like *anti* (“anti-”) and *kdam* (“pre-”), which in Hebrew texts appear as separate words.
10. **NNG/NNGT**: Gerund noun/Gerund noun in construct state form.
11. **NNT**: Noun in construct state form.
12. **QW**: WH words like *when*, *where* and *how*, which do not appear in a determiner position (unlike e.g. *which* as in *which students arrived?*).
13. **REL**: The relativizers *še*, *ašer* and *ha* (= “that”).
14. **VB-M**: A verb in its infinitive form.
15. **ZVL**: Irrelevant data, like initials of author name etc.

Note that the construct state tags NNT, NNGT and JJT, by contrast to the tags NN, NNG and JJ, do not have the definite and genitive features. This is due to the ungrammaticality of examples such as (2c), as opposed to (2a) and (2b).

- (2) a. mfxqi/NNT-ZR aimwn/NN-ZY
 matches training
 “training matches”
- b. mfxqih/NN-ZR-H-NR3
 matches-her
 “her matches”
- c. *mfxqih aimwn
 matches-her training

The part of speech tag-set that was used for annotating the corpus is given in table 3.

3.2 Syntactic tag-set and bracketing methodology

The syntactic tag-set includes syntactic tags which are marked for *agreement features* and *functional features*, the use of which is explained below. The syntactic tag-set is given in table 4, with the agreement features possible on each syntactic tag. The values of agreement features are identical to the values possible on lexical tags that were given in table 2 above. The functional features are given in table 5, with the tags on which they appear.

Among the syntactic tags, the ones that do not appear in the Penn Tree-bank tag-set are the following:

1.	AGR-gn	Agreement particle
2.	AT	Accusative marker
3.	AUX	Auxiliary verb
4.	CC	Coordinating conjunction
5.	CD-gn-(H)	Numeral (definite)
6.	CDT-gn-(H)	Numeral determiner (definite)
7.	COM	Complementizer
8.	DT	Determiner
9.	IN	Preposition
10.	JJ-gn-(H)	Adjective (definite)
11.	JJT-gn	Construct state adjective
12.	H	Definiteness marker
13.	HAM	Yes/No question word
14.	MD-gnpt	Modal
15.	MOD	Modifier
16.	NN-gn-(H H-gnp)	Noun (definite definite-genitive)
17.	NNG-gn-(H H-gnp)	Gerund noun (definite definite-genitive)
18.	NNGT-gn	Construct state gerund
19.	NNP-gn	Proper noun
20.	NNT-gn	Construct state noun
21.	POS	Possessive item
22.	PRP-gnp	Personal pronoun
23.	QW	Question/WH word
24.	RB	Adverb
25.	RBR	Adverb, comparative
26.	REL	Relativizer
27.	VB-gnpt	Verb, finite
28.	VB-M	Verb, infinite
29.	WDT-gn	Determiner question word
30.	ZVL	Garbage
31.	yy*	various symbols, see appendix A

Table 3: The Hebrew POS tag-set

1.	ADJP-gn-(H)	Adjective phrase
2.	ADVP	Adverb phrase
3.	FRAG	Fragment of a declarative sentence
4.	FRAGQ	Fragment of an interrogative sentence
5.	INTJ	Interjection
6.	NP-gn-(H)	Noun phrase
7.	PP	Preposition phrase
8.	PREDP	Predicate phrase
9.	PRN	Parenthetical
10.	S	Declarative sentence
11.	SBAR	Clause introduced by a COM, REL or IN word
12.	SQ	Interrogative sentence
13.	VP	Verb phrase
14.	VP-MD	Verb phrase with a modal verb
15.	VP-INF	Verb phrase with an infinitival verb
Empty categories:		
16.	*T*	NP “trace” in relative clauses
17.	*PRO*	an “understood” NP
18.	*NONE*	missing element

Table 4: The Hebrew syntactic tag-set

1. **FRAG**: A short fragment, which functions as an indicative sentential unit but has no obvious sentential structure. For instance: *xbl* (“too bad”), *ech mspar 3* (“advice number 3”).
2. **FRAGQ**: Similar to FRAG, but in the interrogative. For instance: *wmh bintiim* (“and-what in-the-meantime?”).
3. **PREDP, VP-INF, VP-MD**: See below.

One of the special characteristics of modern Hebrew is the relatively free order of the verb arguments. In sentences where an adverbial is preposed to the beginning of the sentence, the subject often follows the verb, which means that the identification of the verb’s complement(s) cannot be based on simple phrase structure notation. We therefore use a “flat” sentence structure, where the subject, the verb and both its adjuncts and arguments are all daughters of S. In general, we avoid nesting of adjunction structure also with other categories besides the verb phrase. In order to distinguish between the verb’s arguments and its adjuncts, we use special functional features. These features are summarized in table 5. The features OBJ and COM are for complements. Direct NP objects of finite verbs are marked by OBJ. The feature COM is used only for complements of finite verbs that are not direct NP objects, and for any complements of other categories (including infinitival verbs and gerunds). The feature ADV is for adverbial NPs (e.g. *four times*), and it is used in order to distinguish them from the verb’s arguments. On the CNJ feature see below. An example for the usefulness of these features is given in figure 1, where the subject appears between the verb and the object. The translation of this example is: “the newspaper put side by side the red letters and the symbol

feature	role	appears on tags
SBJ	subject	NP, SBAR, VP-INF
OBJ	object	NP
COM	complement	NP, PP, SBAR
ADV	adverbial	NP
CNJ	conjunction	all tags

Table 5: functional features

(S		
(PP (IN lcd)		to-the-side-of
(NP		
(NP-NR-H (H H) (NN-NR awtiwt))		the-letters
(JJ-NR-H (H H) (JJ-NR adwmwt))))		the-red
(VP (VB-ZY3V hcib))		put
(NP-SBJ (H h) (NN-ZY eitwn))		the-newspaper
(NP-OBJ (AT at)		ACC
(NP (NNT-ZY sml)		symbol
(NP (NNT-NY tnwet)		movement
(NP		
(NP-NY-H (H h) (NN-NY htngdwt))		the-resistance
(PP (IN l)		to
(NP (NNT-NY mlxmt)		war
(NP (NNP-NY wiijtnam))))))))))		Vietnam

Figure 1: use of functional features

of the movement resisting to the war in Vietnam”.

In Hebrew, verbless clauses are very common. To annotate NPs, PPs and ADJPs that function as predicates, we added a category PREDP that dominates them under this usage. There are two different kinds of copulas that can appear in such predicative constructions. One, for which we use the tag AUX, is inflected for tense (e.g. *haya=hih*=“was”). Another, for which we use the tag AGR, appears in the present (e.g. *hu=hwa*=“is”). The distribution of the two particles is quite different, which is the reason for their different classification. For instance, the past form copula allows for “subject-AUX inversion” as in the annotated sentence in figure 2, which is translated: “in the days of the cold war, Tukey was an enthusiastic anti-communist”. By contrast, the present form copula cannot be used in such constructions.

In addition to these functional features, the CNJ feature marks conjunctions. This feature, in addition to explicit head marking on syntactic tags, is needed in order to identify the *head constituent* of a given subtree X , which consists of subtrees Y_1, \dots, Y_n . This is done according to the following rules:

1. If X 's category is marked by $+i$, then Y_i is X 's head constituent.
2. Otherwise: if X is marked by the CNJ feature, then any subtree Y_i with a category

(S		
(PP (IN b)		in
(NP (NNT-ZR imi)		the-days-of
(NP		
(NP-NY-H (H h) (NN-NY mlxmh))		the-war
(ADJP-NY-H (H h) (JJ-NY qrh))))		the-cold
(S (AUX-ZY3V hih)		was
(NP-SBJ (NNP-BY jwqi))		Tukey
(PREDP		
(NP		
(MOD anji)		anti
(NN-ZY qwmwnisj)		communist
(ADJP (JJ-ZY nlhb))))))		enthusiastic

Figure 2: subject-AUX inversion

different than CC (coordinating conjunction) is a head constituent of X .

3. Otherwise: if X, Y_1, \dots, Y_n are all of the same category then Y_1, \dots, Y_n are all head constituents of X .
4. Otherwise: The head constituent of X is Y_i , with the smallest i such that Y_i is not of any category in $\{AT, DT, WDT, H, MOD, AGR, AUX\}$.

Rule 3 is due to the common appearance of appositional constructions in Hebrew. For instance, in the noun phrase *h-mamn rz* (“the-coach Raz”) both constituents are NPs, and both are head constituents.

The agreement features (see table 2) of the head constituent determine the agreement features of the whole constituent. The same is true for the definiteness marking feature H, except for one notorious case – the Hebrew construct state. In this case the head constituent is the construct state nominal (the leftmost sub-constituent), but the definiteness of the construction is determined by the nominal following it. For instance, the construct state NP in (3a) is definite whereas in (3b) it is indefinite, although the head (the first nominal in the construction) is indefinite in both cases.

- (3) a. (NP-NY-H (NNT-NY nbxrt) (NP-ZY-H (H h) (NN-ZY nwer)))
team the-youth
“the youth team”
- b. (NP-ZR (NNT-ZR ciwni) (NP (NN-NY drk))) marks road
“road marks”

The empty categories for *T* (trace) and *PRO* are quite standardly adopted from theoretical linguistics. Traces are used for “missing” NPs in relatives. PRO elements, which are quite frequent in Hebrew, indicate that a subject is missing, but the verb nevertheless has distinct agreement features. NONE elements are used for cases of missing elements which are not easily classified as one of these two empty categories. The following examples illustrate the use of these empty categories.

- (4) rpwbliqaim/NN-ZR h/REL *T*/-NONE- nwjim/MD-ZRAH lhskim/VB-M at/IN
 republicans who *T* tend to-agree with
 hm/PRP-ZR3
 them
 “Republicans who tend to agree with them”
- (5) *PRO*/-NONE- fmeti/VB-BY1v at/AT h/H tiawrih/NN-NY h/H zw/JJ-NY
 PRO heard-I ACC the theory the this
 “I heard this theory”
- (6) jwqi/NNP-BY qibl/VB-ZY3V rq/MOD 52/CDT-BR alp/CDT-BR *NONE*/-NONE-
 Tukey received only 52 thousand *NONE*
 “Tukey received only 52 thousand (dollars)”

3.3 The construction process of the tree-bank

In the annotation of the corpus we made use of two available software tools:

- The morphological analyzer of Segal [Segal, 2000], which was used to obtain preliminary segmentation, POS tags and agreement features of words in the corpus.
- The SEMTAGS graphical tool of Bonnema [Bonnema, 1997], which was used for aiding manual syntactic annotation.

The annotation of the sentences in the tree-bank was obtained as follows:

1. *General translation from morphological analysis to POS tag sequences:* PERL scripts were developed in order to transform the morphological scheme of [Segal, 2000] into a preliminary POS tag-set. This translation is not one-to-one, because the tag-set that is used for the morphological analysis is less fine-grained than the one we used for syntactic annotation. For instance, Hebrew prefixes that have different syntactic roles (preposition, relativizer, conjunction, etc.) were all identified in the morphological analysis of a word under a morphological feature of “conjunction”. In average, each morphological analysis of a word corresponds to 1.4 sequences of POS tags in the corpus, with standard deviation 1.
2. *Preliminary morphological analysis:* The morphological annotation of 250 sentences that were manually analyzed by Segal, and additional 250 sentences that were automatically analyzed by Segal’s morphological analyzer [Segal, 2000], was translated to the POS tag-set using the PERL scripts.
3. *Correction of morphology, and syntactic analysis:* These 500 sentences were manually given a syntactic annotation. This process was aided by the SEMTAGS graphical tool. In this process the annotators corrected wrong morphological analyses, chose among the different translations of correct morphological analyses into POS tag sequences, and added syntactic annotation according to the scheme described above.
4. *Correction of annotation scheme:* The lexical and syntactic annotation scheme was updated according to the experience gained with these 500 sentences, and the sentences in the tree-bank were corrected accordingly.

The annotation process was very slow – around 40 words per hour for full syntactic annotation. The annotation rate for the Penn tree-bank (375-475 words per hour) reported in [Marcus et al., 1993] was 10 times higher. We believe that there are two main reasons for this difference:

- The annotation of the Penn tree-bank relied on considerable experience from tagging of other large English corpora. By contrast, no annotated Hebrew corpus is presently available, and consequently some of the syntactic and lexical conventions had to be modified during the annotation process.
- The syntactic annotation of the tree-bank had to be performed without the aid of any parser, while the Penn tree-bank was annotated by correcting the output of an English parser (Donald Hindle’s Fidditch).

This indicates that a parser for Hebrew could significantly increase the annotation rate. In the section 4, we describe the adjustment of the Tree-Gram model for Hebrew and results of testing the parser generated by the Tree-Gram model on the annotated sentences.

3.4 Figures about the tree-bank

Some numbers summarizing numerical aspects of our tree-bank are listed in table 6. Most notable are the figures concerning the frequencies of occurrence of morphemes and words: while 67% of all morphemes occur only once, this is 75% for words. Figure 3 plots the frequency counts of the 50 most frequent morphemes and words. As expected, morphemes are more frequent than words and suffer less from sparse-data problems. Among the ten most frequent morphemes, one finds *h* (“the”), various prepositions e.g. *b* (“in”) and *l* (“to”) and punctuation marks (e.g. the comma).

Other figures pertaining to the ambiguity of morphemes on the POS tag level are shown in the last rows of table 6. While over all morphemes the average number of POS tags per morpheme is 1.3 (std 0.7), this figure rises to 2.4 (std 0.9) for morphemes occurring more than once. This could be seen as evidence for the fact that in our small tree-bank, the ambiguity level of once occurring morphemes (1.0 (0.0)) is too far from reality. Hence, once occurring morphemes could be provide a coverage problem for POS tagging and parsing. In the sequel, we employ the Hebrew morphological analyzer as a supplementary source of knowledge on the unknown and once-occurring morphemes to avoid problems of coverage.

Figure 4 plots the number of sentences as a function of their length (i.e. number of morphemes). The mean sentence length is 22.8 morphemes with a std of 13.7. These figures are comparable with the average sentence length (counting words instead of morphemes, though) in the WSJ tree-bank. About 90% of all sentences consist of 40 or less morphemes. Very few sentences consist of more than 70 morphemes (exactly 4 sentences, i.e. 0.8%).

Table 6 also shows figures pertaining to the syntactic annotation in the tree-bank. The total number of constituent nodes (beyond POS tags) in parse-trees expresses that the parse-trees contain, on average, just over one constituent-node per word. The number of (unique) Tree-Bank rules² that constitute the parse-trees is partitioned into lexical and non-lexical rules. The lexical rules are those consisting of a left-hand side labeled with a POS tag and a

²A one-level subtree in a tree-bank parse-tree, i.e. a node and the sequence of its child-nodes, maybe considered to represent a phrase-structure (Context-Free) rule: the label of the node is the left-hand side symbol and the sequence of child-node labels is the right-hand side of this rule.

Sentence, word and morpheme statistics	
Number of sentences	498
Number of unique sentences	492
Number of unique morphemes	3130
Number (%) of once-occurring morphemes	2103 (67.1%)
Total count of morpheme-occurrences	10866
Average occurrence per morpheme	3.5
Number of unique words	4027
Number (%) of once-occurring words	3033 (75.3%)
Total count of word-occurrences	8419
Average occurrence per word	2.0
POS tags and constituent labels (with semantic/complement tags attached)	
POS tag-set without (with) features	42 (204)
Bare constituent-labels set	22
set of constituent-labels without (with) features	99 (128)
Total number of constituents in all trees	11123
Morpheme ambiguity: mean and std of POS tags with features	
over all morphemes	1.3 (0.7)
over morphemes with occurrence-count > 1	2.4 (0.9)
Number of different rules	
<i>lexical</i> rules without (with) features	3431 (3515)
once-occurring <i>lexical</i> rules without (with) features	2363 (2439)
<i>non-lexical</i> rules without (with) features	1106 (1580)
once-occurring <i>non-lexical</i> rules without (with) features	728 (1012)

Table 6: The tree-bank in numbers

right-hand side labeled with a morpheme (i.e. terminal). About 69% of the lexical-rules and 65% of the non-lexical rules occur exactly once. The length of the right-hand side (rhs) of a non-lexical rule may vary from one to ten symbols; the number of different rules per rhs-length is as follows: 78 (unary), 296 (binary), 290 (3-nary), 204, 139, 49, 37, 6, 5, 1 (10-nary). With so many different rules containing 3-10 symbols (about 65%) on the right-hand side, there is reasonable chance that many more will appear in parse-trees of future, unseen sentences.

4 Evaluating the utility of a small tree-bank

The main reason for building a Hebrew tree-bank is to facilitate Hebrew language processing, in particular morphological analysis and syntactic parsing. The cost of annotation of the present tree-bank, currently estimated at about 400-500 man hours for the first 498 sentences, exhibits the complexity of the task that a human annotator must accomplish. The question that arises at this stage of the project is whether we can use the currently available tree-bank of 498 morphologically and syntactically analyzed sentences for obtaining a more efficient *semi-automatic* annotation process.

The small size of the tree-bank implies at least two problems for probabilistic parsers.

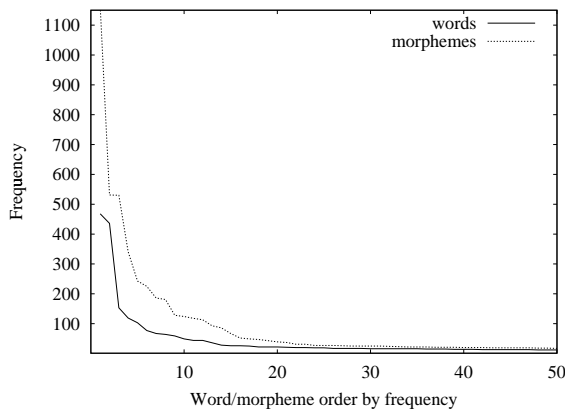


Figure 3: Frequency counts of the 50 most frequent words/morphemes

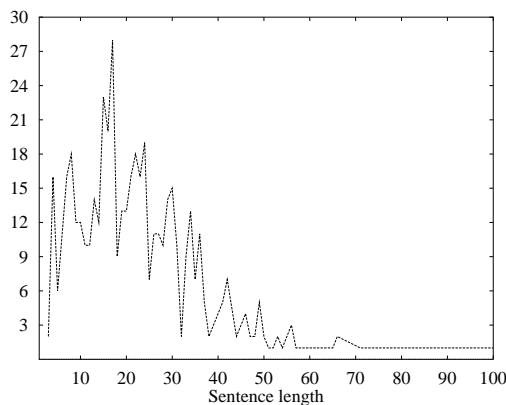


Figure 4: Counts of sentences as a function of their length

In the first-place, the notorious sparse-data problem: because of the small tree-bank, many phenomena will not have occurred in it and other phenomena that did occur will probably be over- or under-represented, i.e. their distributions will not be representative. Once occurring phenomena provide good examples of this sparseness problem as table 6 shows: one-occurring morphemes (67% of all morphemes) are dramatically less ambiguously represented at the POS tag level than other morphemes, and the many once-occurring (Tree-Bank) rules (66%) signify, most probably, highly non-saturated distributions over syntactic structures, but possibly annotation errors. An additional problem is the problem of parser-coverage: as mentioned in section 3.4, the length of the symbol-sequences on the right-hand side of grammar rules varies between one and ten. Around 65% of the non-lexical rules occurred only once. Hence, there is a strong possibility for variation in grammar rules in parse-trees of future, novel sentences. This implies that it might be hard to predict syntactic structure with high accuracy.

In what follows we explore the utility of the available tree-bank in combination with the Tree-gram model [Sima'an, 2000] and the morphological analyzer of Segal [Segal, 2000]. After a short introduction of the Tree-Gram model, we describe the experiments we ran on the tree-bank using this model. Analyzing the results of these experiments, we conclude that at this stage, semi-automatic segmentation of words (without full POS tagging) in conjunction to the Tree-gram Hebrew parser that is generated from the existing tree-bank, constitute a useful tool for semi-automatic syntactic annotation of a larger Hebrew corpus.

4.1 Tree-gram probabilistic parsing

In many contemporary models of language processing, a tree-bank provides an important source for statistics over linguistic phenomena, which can be employed for resolving ambiguities during processing. In probabilistic syntactic parsing in particular, a tree-bank is used for inducing probabilistic grammars e.g. [Scha, 1990, Bod, 1992, Magerman, 1995, Bod, 1995, Collins, 1997, Charniak, 1999, Sima'an, 2000].

A probabilistic language model consists of a probabilistic grammar and a model of how probabilities of parse-trees and sentences are derived. In a probabilistic grammar, a formal grammar is extended with a finite set of conditional probabilities, each associated with a

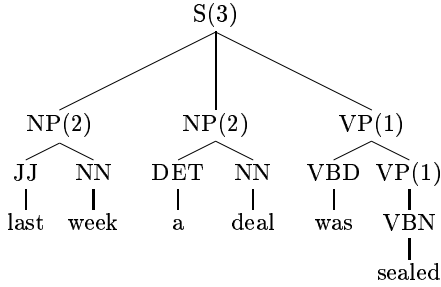


Figure 5: An example parse-tree. The number of the head-child of a node is specified between brackets (e.g. the third child (VP) of the node labeled S carries the head).

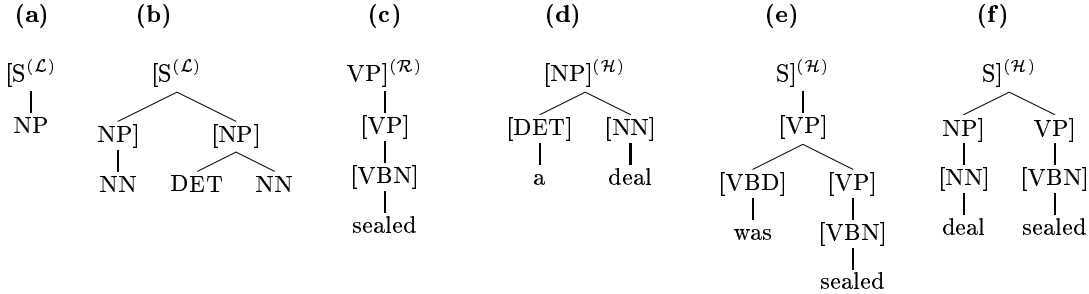


Figure 6: Some T-grams extracted from the tree in figure 5: the superscript on the root label specifies the *T-gram role*, e.g. the left-most T-gram is in the LEFT role. Non-leaf nodes are marked with “[” (left-STOP) and “]” (right-STOP) to specify whether they are complete from the left/right or both (the other non-complete nodes, i.e. from both sides, are not marked at all).

$\langle rule, history \rangle$ pair. The *rule* is a “rewrite-event” (e.g. a Context-Free Grammar (CFG) rule) and the *history* consists of contextual material. The probability expresses the “likelihood” of the rewrite-event *given the history*, i.e. it is conditioned on that history. Usually, these probabilities are acquired from the tree-bank by normalizing the relative frequency counts $f(e)$ of the rewrite-event e by the relative frequency count $f(h)$ of the history h ; smoothing methods are also used for estimating the probabilities of rewrite-events that did not occur together with some histories in the training tree-bank. When parsing an input sentence S , a probabilistic model aims at finding the parse-tree T which maximizes the joint probability of S with T , i.e., it aims at solving $argmax_{T \in G} P(T, S)$ for the probabilistic grammar G .

The question of what rewrite-events and what histories to extract from the tree-bank trees is a major question in probabilistic parsing. In this paper we adopt the Tree-gram model [Sima’an, 2000], which combines aspects from Data-Oriented Parsing (DOP) [Scha, 1990, Bod, 1995, Scha et al., 1999, Sima’an, 1999] with aspects from Bilexical-Dependency Markov-Grammars [Collins, 1997, Charniak, 1999]. In what follows we provide a brief overview of the Tree-gram model. A formal description and further details of the Tree-gram model can be found in [Sima’an, 2000].

The rewrite-events of the Tree-gram model are *connected subgraphs* of the tree-bank trees, called Tree-grams. A Tree-gram is a “partial” parse-tree: its leaf nodes are labeled either with terminals or with non-terminals, and its internal and root nodes are labeled with non-terminal symbols. A non-terminal node may be *complete*, in the sense that the node already dominates all “necessary” children, or it may be incomplete. A complete node is labeled with an extra

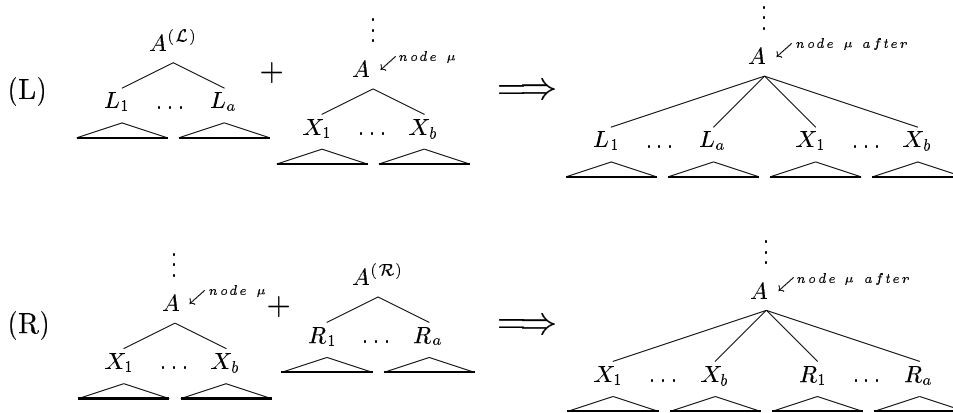


Figure 7: A T-gram is generated by attachment at μ in a partial parse-tree. The T-gram being generated is marked with \mathcal{L} and \mathcal{R} to denote its role. In (L) a LEFT Tree-gram and in (R) a RIGHT Tree-gram is generated. Note that node μ must be non-complete.

special symbol “STOP” both at its left/right sides, specifying that the node does not accept anymore children to the left/right of the children that it currently dominates. When STOP is absent from either or both sides of a node, the node is incomplete and the subtrees that it dominates may be extended with more subtrees that are embedded in some other Tree-grams as described next (hence, non-terminal leaf nodes are always incomplete).

The Tree-grams are partitioned into three subsets, called *roles*, according to the kind of children that the root of a Tree-gram dominates. First, the *child-node* of a given node is the child that carries its head-word. When a Tree-gram’s root dominates its head-child (and possibly other children), the Tree-gram is in the “Head” role; when it dominates only children which are originally found (in the tree-bank tree from which the Tree-gram was extracted) to the left (right) of the head-child (e.g. left-modifiers of the head-child), it is in the LEFT (RIGHT) role. In essence, these roles express information about the nature of the Tree-gram with respect to the context from which it was extracted. Some example Tree-grams are shown in figure 6.

The history h on which the probability $P(t|h)$ of a Tree-gram t is conditioned, consists of the label of the root-node of t and the role of t (i.e. HEAD, LEFT or RIGHT). Further conditioning material in the histories aims at capturing information about the nature of the child/sister nodes that it usually appears with (as encountered in the training tree-bank). For example, LEFT and RIGHT Tree-grams probabilities may be conditioned on so called “Markovian” information, e.g the label of the sister to the left/right in the original tree-bank tree; while HEAD tree-grams probabilities may be conditioned on a subcategorization-frame set of the head-word (see [Collins, 1997, Sima’an, 2000]).

Tree-gram rewrite processes, i.e. derivations, start from the start-symbol TOP, which is an incomplete non-terminal. At each rewrite-step, an incomplete node μ is selected and rewritten by a suitable Tree-gram. When μ is a leaf node labeled with a non-terminal A , it is rewritten by a HEAD Tree-gram with a root labeled A (much like rewriting takes place in CFGs, i.e. “vertical expansion”); when a non-leaf node μ is labeled with a non-terminal A and it is incomplete, it may be rewritten with LEFT and RIGHT Tree-grams that have roots

also labeled A . The latter rewriting allows *horizontal* expansion of the parse-tree at node μ (see figure 7). The rewrite process terminates when the resulting parse-tree consists of only complete nodes. The probability of a derivation is equal to the product of the probabilities of all Tree-grams that participate in the rewrite steps, while the probability of a parse-tree is the sum of the probabilities of all possible derivations that generate it (through the different Tree-gram combinations). In the present implementation, for an input sentence S , the parser aims at finding the derivation d which solves $\text{argmax}_d P(S, d)$, rather than the parse-tree T that solves $\text{argmax}_T P(S, T)$. We choose to do so, simply because the latter problem is known to be NP-Complete, while the first one is solvable in time cubic in sentence-length [Sima'an, 1996].

4.2 Tree-gram parsing of the Hebrew tree-bank

Most existing parsing models were applied to English for which large tree-banks exist, e.g. the Wall Street Journal (WSJ) tree-bank [Marcus et al., 1993] which consists of approximately 50,000 syntactically analyzed sentences. Morphology does not play the major role in these tree-banks. Evidence for this observation comes from various work in the literature: many models which were applied to the WSJ could afford to collect statistics over joint-occurrences of syntactic and lexical phenomena, where the words are not segmented into morphemes, e.g. [Collins, 1997, Charniak, 1999, Sima'an, 2000]. Furthermore, usually, a single probabilistic model is used for both part-of-speech tagging (POS tagging) and syntactic parsing.

At this stage, our tree-bank is too small (1% of the WSJ) for allowing the full integration of both morphological analysis and syntactic parsing into one model. In fact, it is unclear whether direct application of the existing parsing models to Hebrew, with its complex morphology, can be as successful as for English. Clearly, this question becomes relevant only when the size of the tree-bank allows better estimation of language models. Meanwhile, we aim at minimizing the cost of tree-bank annotation.

There are various different settings for utilizing the existing tree-bank through the Tree-gram model together with the available morphological analyzer. Given a Hebrew sentence, the analysis of this sentence consists of 1) segmentation of the words into morphemes, 2) the assignment of a POS tag and suitable features to each morpheme, and 3) the assignment of a syntactic structure. In what follows we explore the second and third aspects of sentence analysis and leave word-segmentation for future work. Word-segmentation into morphemes demands technical coordination of the morphological analyzer with the Tree-gram parser which has not been carried out.

In the sequel we explore the following settings for further semi-automatic annotation of the Hebrew corpus by means of the Tree-gram model:

POS tagging + Parsing using only the tree-bank: We assume that the morphological analysis only consists of segmentation of the words into morphemes. The parser is applied to morpheme sequences and aims both at POS tagging and parsing. The main problem will be dealing with *unknown morphemes*: initially we will assume no prior knowledge on unknown morphemes.

POS tagging + Parsing - using the morphological analyzer for unknowns: The setting here is the same as before, except that now we use the morphological analyzer for the *unknown-morphemes*. The analyzer is run on some text from the corpus. The morphemes and POS tags which it produces are gathered in a "sublexicon" which serves

the parser as an ambiguous resource about unknown-morphemes. We also explored a direct generalization of this setting by allowing the POS tag-sets of the *once-occurring morphemes*, in the parser’s training set, to be expanded by their tag-set as found in the sublexicon, just like unknown-morphemes.

Syntactic parsing: We assume a more advanced morphological analysis which does all the work necessary at the word level: it may take place by a combination of automatic annotation by means of the existing morphological analyzer [Segal, 2000] together with a human annotator which further disambiguates and corrects the POS tags. Hence, the parser applies to disambiguated correct POS tag sequences and aims at syntactic parsing only. Unknown POS tags are resolved through an impoverished implementation of the notion of “similarity” to known POS tags.

The results of these experiments warranted two simple observations: (1) the small tree-bank allows the probabilistic parser to achieve surprisingly useful results in parsing POS tag sequences, (2) the problem of sparse-data in parsing morpheme-sequences (e.g. unknown and infrequent morphemes) can be avoided relatively successfully when the Hebrew morphological analyzer is allowed to complement the probabilistic parser. The latter observation will provide us with enough reason to believe that future semi-automatic syntactic annotation of new sentences will be possible: it will proceed from the morpheme-level, a reasonably comfortable point, given our current situation where almost all annotation takes place manually. Next we describe the general experimental conditions followed by the details and results of the experiments pertaining to each of the above listed settings.

4.3 Limitations, smoothing, methodology and evaluation

Tree-bank non-terminals: In the tree-bank, a non-terminal symbol consists of the conjunction of a syntactic category (e.g. NP) and a sequence of features (gender, number, person, tense and definiteness). Currently, we do not percolate the features from the lower nodes (manually annotated features) to the upper nodes of the tree-bank trees for two reasons: 1) to avoid further complications with sparse-data problems, and 2) because we expect the Tree-grams to capture many of the feature agreements through their probabilities of joint occurrence of constituents. We also employ only a single head-child for every node in a tree-bank parse-tree: whenever there are multiple head-children for a node, we take the left-most to be the only head-child. This decision is inherent to the current implementation and has no theoretical justification.

Katz smoothing: Naturally, due to the small size of the tree-bank, there are many non-terminals (conjunctions of syntactic categories and features) and Tree-grams that did not appear. A Tree-gram’s probability is conditioned on some portions of the context in which it occurred. In this case, the context of a Tree-gram consists of, among others, the label of its root-node (N, F), where N is the syntactic category and F is the sequence of features. Tree-grams that did not appear in such (relatively) complex contexts will receive probability zero. To avoid the degradation of coverage of the parser, it is necessary to simulate unseen Tree-grams. Under this setting we apply the Katz back-off smoothing model [Katz, 1987, Chen and Goodman, 1998] to the Tree-grams that were found in the Tree-bank. A set of Tree-grams that did not occur in a given context is simulated by “backed-off” Tree-grams: backed-off Tree-grams are obtained from the Tree-grams that occurred in the Tree-bank by

removing the features (and any other information) from their conditioning-context, thereby leaving only the syntactic category as conditioning context. Hence, the probabilities of backed-off Tree-grams are conditioned on simpler contexts. In Katz smoothing, the probabilities of the original Tree-grams are *discounted* according to Good-Turing [Good, 1953] and the probability mass saved from this discounting is redistributed between the backed-off Tree-grams relative to their probabilities conditioned on their simplified root-node labels.

Cross-validation experiments: For each experiment, we employ 5-fold cross-validation blind testing: we randomly split the tree-bank into a training-set of 448 trees and 50 test trees; we repeat this procedure 5 times with different random partitions. The test-trees are not involved in any training activity.

Evaluation methodology: In each experiment, the trained parser is applied to the sentences found in the corresponding test-set. The parser-output is compared to the test-set trees on the PARSEVAL measures [Black et al., 1991]: labeled bracketing precision and recall. In PARSEVAL, a parse-tree is considered as a set of constituents³: each constituent is a triple $\langle i, j, L \rangle$, where i and j are indices into the words dominated by the constituent and L is its phrasal-label. A tree P output by the parser is compared to the corresponding test-set tree T by computing $|P \cap T|$, i.e. the number of matching constituents⁴ (brackets and label). *Labeled Bracketing Recall* (LBR) is the mean (over the test-set sentences) of $\frac{|P \cap T|}{|T|}$ and *Labeled Bracketing Precision* (LBP) is the mean of $\frac{|P \cap T|}{|P|}$ (precision is defined to be zero when $|P|$ is empty). We also report two the *tree Exact-Match measure* (percentage of parser-output trees that exactly match their test-set counterparts) and the *No Crossing measure* (percentage of parser-output trees that contain only brackets that do not cross any of the test-tree brackets). When relevant, we also report the *POS tagging precision* (number of correct tags to total number of tags in parser-output, including features) and recall (number of correct tags to total number of tags in test-parse, including features). This evaluation procedure is employed in all experiments reported in the sequel.

4.4 Input: morpheme-sequences; Output: POS tags and parses

In this experiment we evaluate the utility of the combination of the Tree-gram parser with the small tree-bank on the combined task of POS tagging and parsing. We use from the tree-bank only the word-segmentation into morphemes. Thus, the input of the parser consists of the correct morpheme sequences. The boundaries each sequence of morphemes that correspond to a single word in the corpus can or cannot be marked. We explore two versions: 1) word-boundaries are not supplied to the parser, and 2) word-boundaries are supplied to the parser. We note that our expectations from this experiment should be modest given the available statistics over morphemes; with about 3130 different morphemes and a total of 10866 morpheme-occurrences in the whole tree-bank, every morpheme is expected to occur

³POS tags are not included as constituents because they are trivial constituents. POS tagging is evaluated separately.

⁴The output parse-trees do not specify any of the label-extensions {SBJ,OBJ,COM}. Furthermore, the parser's output specifies gaps and features (only on the labels where the tree-bank annotation provides the features, i.e. no percolation). However, gaps and features are *not included* in the current evaluation on the syntactic level (i.e. beyond POS tags); *when present, POS tag features are always included in the evaluation of POS tagging recall and precision.*

LUB	# sen.s	LBR	LBP	No Cros.	Ex. match	PTag Rec.	PTag Prec.
10	9.4 (2.0)	58.7 (7.3)	58.6 (8.0)	72.7 (21.6)	7.8 (7.2)	82.3 (3.9)	83.1 (3.9)
20	25.4 (4.6)	54.0 (4.4)	54.5 (5.0)	43.3 (6.2)	3.2 (3.1)	84.8 (3.4)	85.0 (3.2)
30	39.0 (4.9)	51.0 (2.7)	51.7 (2.6)	29.8 (6.5)	1.9 (1.9)	83.6 (2.7)	84.3 (1.7)
40	44.8 (2.6)	48.7 (3.8)	50.0 (3.2)	27.3 (6.1)	1.7 (1.8)	81.8 (3.3)	83.8 (2.0)
∞	50.0 (0.0)	46.7 (4.2)	48.3 (3.2)	25.2 (5.2)	1.6 (1.6)	80.3 (3.6)	83.6 (1.7)
10	9.4 (2.1)	61.8 (3.7)	61.4 (3.5)	65.8 (14.5)	6.3 (6.0)	78.4 (4.5)	78.4 (4.5)
20	25.4 (4.6)	58.2 (2.0)	59.1 (3.7)	40.9 (8.0)	2.5 (2.3)	80.1 (2.4)	81.0 (3.4)
30	39.0 (4.9)	54.8 (2.5)	56.2 (2.9)	28.6 (6.8)	1.5 (1.3)	80.1 (1.9)	81.2 (2.3)
40	44.8 (2.6)	53.3 (3.0)	54.6 (2.2)	25.4 (6.7)	1.3 (1.2)	79.3 (2.3)	80.7 (2.0)
∞	50.0 (0.0)	48.7 (4.8)	53.2 (2.9)	25.6 (6.2)	1.2 (1.1)	73.7 (4.8)	80.6 (1.9)

Table 7: Results of parsing morpheme sequences to length upper-bound (LUB). First five rows pertaining to morphemes without word-boundaries in the input, the last five rows to morphemes with word-boundaries. The averages and standard deviations are taken over a 5-fold cross-validation experiment. The standard deviations (std) are reported between brackets.

in a sentence about 3.5 times only. In reality, about 67% of all morphemes occur only once in the whole tree-bank.

Removing the features: Due to the small size of the tree-bank, it turned out that on average about 21% of the morphemes in every test-set were unknown in the training-set. With an average sentence length of around 23.5 (std of 2.3) morphemes, there are on average about 4.7 unknown morphemes per sentence. Moreover, 95.2% of all test-sentences contained at least one unknown morpheme. To avoid run-time memory problems, especially with so many unknown morphemes, we had to remove all features from the annotation, leaving bare phrase-structure symbols (pretty much similar to the WSJ tree-bank). This reduces the total number of POS tags (lexical-categories with feature-sequences) from 199 to only about 30.

Training procedure: For this experiment, we extract from each of the (featureless) training tree-banks a Tree-gram parser. We allow morphemes to lexicalize only Tree-grams that have a root labeled with a POS tag category: that is to say that only Tree-grams of depth 1 are lexicalized. Furthermore, we limited the size of the Tree-grams by limiting their depth to 5 and their number of incomplete nodes to 4. This limits the number of Tree-grams drastically (to around 50,000). The conditioning history of a Tree-gram probability consisted of its root-label together with a choice of one of two possibilities: if the root-node of the Tree-gram has a subcat-frame, this is used as conditioning material; otherwise, we employed a first-order Markov process conditioning on the label of preceding sister node of a LEFT/RIGHT Tree-gram.

Unknown morphemes: For every unknown morpheme, we allowed all open-category POS tags in the training-set (on average 29.6 POS tags) to be supplied to the parser with a uniform distribution (signifying our state of complete ignorance as to the category of the morpheme). Hence, the input to the parser, when there are unknown morphemes, is a graph, where for every morpheme, there are several alternative POS tags. For the known morphemes, there

were on average about 1.15 POS tags in the training-set; a very small number, probably due to the size of the tree-bank. This means that the average number of POS tags-sequences per average sentence of 20 morphemes is about $29.6^{4.7} * 1.15^{15.3} \approx 9 \times 10^6$ sequences. These POS tag sequences are packed into a so called “word-graph” (Finite-State Machine) representation, which originates from work on speech-understanding [Oeder and Ney, 1993]. The probabilistic parsing of word-graphs is described in [Sima'an, 1999].

Empirical results: The first five rows of table 7 list the average and standard deviation results over the five blind tests on parsing morpheme-sequences *without word-boundaries*. The effect of unknown morphemes clearly shows on these results. With 95% of the sentences containing at least one unknown morpheme, contextual syntactic information is a weak means for POS tagging: 80.3% mean-recall and 83.6% mean-precision. The same conclusion applies to syntactic (featureless) parsing where only less than half the nodes is completely correct (label and bracket) 46.7% labeled recall and 48.3% labeled precision. As expected, in general, it is easier to score better on shorter sentences.

The last five rows of table 7 list the results of the same setting but now add the word-boundaries to the input. Word-boundaries were exploited by distinguishing between four mutually exclusive kinds of morphemes and their POS tags: at the beginning of a multi-morpheme word, at the end of a multi-morpheme word, in the middle of a multi-morpheme word or simply corresponding to a one-morpheme word. This partitions the morphemes and the POS tags: we added a corresponding 4-value feature to morphemes and POS tags and allowed Katz back-off on this feature. The results show that this way of exploiting word-boundaries, on the one hand, degrades POS tagging results, but on the other hand, improves syntactic parsing results. The partitioning of the morphemes and POS tags seems to degrade the probabilities of lexicalized Tree-grams (which consist of lexical rules *POStag* \rightarrow *morpheme*). For the probabilities of the non-lexicalized syntactic Tree-grams, adding word-segmentation knowledge to the POS tags seems to provide useful clues as to constituent labels.

4.5 Input: morpheme-sequences + sublexicon; Output: POS tags and parses

In this experiment, we changed the following elements from the preceding setting (i.e. parsing morpheme-sequences):

1. we do *not* strip off the features, rather we keep them on the non-terminals (employing Katz back-off for avoiding coverage problems),
2. we employ an automatically acquired ambiguous “sublexicon” for supporting the analysis of (most of) the unknown and low-frequency morphemes. This sublexicon was extracted by running the morphological analyzer [Segal, 2000] on a corpus subsuming ours (consisting of 526 sentences) and collecting a set of morpheme-POS tag pairs, with on average 1.4 POS tags per morpheme (and standard-deviation of 1.0).

It is important to note that the sublexicon contains morphemes that were obtained by automatic segmentation of the words. This automatic segmentation agreed in most cases with the segmentation in the tree-bank. However, for some words, the two segmentations did not agree; therefore, there were morphemes in the test-sets that were not found in the sublexicon (about 2.5% of all unknown morphemes, or approximately 0.5% of all test morphemes).

4.5.1 Applying the sublexicon to unknown morphemes only

LUB	# sen.s	LBR	LBP	No Cros.	Ex. match	Ptag Rec.	Ptag Prec.
10	9.4 (2.1)	81.4 (4.4)	78.4 (6.4)	78.3 (8.2)	21.9 (20.0)	89.1 (1.1)	89.1 (1.1)
20	25.4 (4.6)	77.3 (1.6)	75.0 (1.8)	43.3 (7.5)	10.5 (9.9)	89.6 (2.3)	89.6 (2.3)
30	39.0 (4.9)	73.5 (1.9)	71.5 (1.7)	29.6 (6.6)	6.7 (6.5)	90.0 (1.0)	90.0 (1.0)
40	44.8 (2.6)	71.5 (1.5)	70.0 (1.2)	26.3 (6.8)	6.0 (5.9)	89.0 (1.2)	89.7 (0.7)
50	48.0 (0.7)	69.0 (3.7)	68.4 (1.5)	25.4 (6.1)	5.8 (5.7)	87.5 (3.5)	89.6 (0.6)
∞	50.0 (0.0)	65.5 (4.3)	68.0 (1.5)	26.0 (5.5)	5.6 (5.5)	83.7 (4.5)	89.7 (0.7)

Table 8: Results of parsing morpheme sequences to length upper-bound (LUB) with a morphological-analyzer applied for obtaining a sublexicon for unknown morphemes. No word-boundaries used. The averages and standard deviations are taken over a 5-fold cross-validation experiment.

For this experiment, the parser consulted the sublexicon every time it encountered an unknown morpheme. For the unknown morphemes that it did not find in the sublexicon, the parser assumed all 199 POS tags.⁵ In both cases it assumed a uniform distribution over the POS tags that are possible for an unknown morpheme.

Table 8 exhibits the results of this experiment. Most notably, the LB recall and precision for all sentences (65.5% and 68%) are at least 15% better than the best results in the preceding experiment (row 5 in table 7: 48.7% and 53.2% respectively). Furthermore, POS tagging recall and precision (83.7% and 89.7%) are respectively 3% and 6% better than the preceding experiment (80.3% and 83.6%). For sentences of length smaller or equal to 40 morphemes (about 89.6% of all test sentences), the POS tagging result (89.0% and 89.7%) is 6-9% better than the results on the same sentences of the preceding experiment (row 4 of table 7); on the same sentences, LB recall and precision (71.5% and 70%) are about 20% better than the results of assuming no knowledge on the unknown morphemes (48.7% and 50% respectively).

Clearly, although the morphological analyzer is an ambiguous POS tagging source, and although it is imperfect, still it is a very useful source on unknown morpheme POS tags. It enables limiting the number of POS tags per unknown morpheme from 199 to 1.4 (on the average), enabling the Tree-gram parser to improve POS tagging and labeled-bracketing by a significant percentage.

4.5.2 Applying the sublexicon to morphemes with frequency ≤ 1

As mentioned earlier, due to the small size of the tree-bank, about 67% of all morphemes occur only once. The POS tag-set for each once-occurring morpheme is a singleton. Given that more frequent morphemes have an ambiguity POS tag-set of average size 2.4 (0.9) (i.e. on average 2.4 POS tags per morpheme), it is probable that once-occurring morphemes will miss many of their possible POS tags in new contexts. In the light of the relative success of supplementing the parser with the (imperfect) sublexicon for resolving unknown morphemes, one cannot help but apply the same strategy to once occurring morphemes also. This is indeed what we did in a second experiment maintaining, otherwise, the same setting as the

⁵Strictly speaking, it is possible to reduce this ambiguity a bit by eliminating potential POS tags of closed categories that do not match the morpheme. However, the improvement that can be obtained in this way is negligible.

preceding one. For once-occurring morphemes, we supplemented their singleton POS tag-sets with the set of POS tags found in the sublexicon, as assigned by the morphological analyzer. We maintained a uniform distribution over the morphemes originating from the sublexicon. This raises the ambiguity of once-occurring morphemes at all levels, from POS tagging to syntax. However, this, hopefully, gives the parser more (limited) choice which, on average, will prove beneficial. Table 9 shows the results of this experiment. There is an improvement

LUB	# sen.s	LB Rec.	LB Prec.	No Cros.	Ex. match	Ptag Rec.	Ptag Prec.
10	9.4 (2.1)	84.3 (6.4)	83.2 (6.4)	91.5 (5.1)	20.6 (14.8)	89.6 (3.1)	91.5 (2.7)
20	25.4 (4.6)	77.1 (1.3)	75.5 (1.9)	47.1 (9.8)	9.6 (7.6)	89.3 (2.3)	89.7 (2.4)
30	39.0 (4.9)	74.9 (2.1)	73.2 (1.6)	34.7 (7.0)	5.8 (4.4)	90.3 (0.6)	90.5 (0.8)
40	44.8 (2.6)	73.3 (1.7)	71.5 (1.8)	30.3 (7.4)	5.2 (4.1)	89.9 (0.8)	90.1 (1.0)
50	48.0 (0.7)	71.0 (3.4)	69.6 (2.7)	28.8 (7.3)	5.0 (4.1)	89.1 (1.9)	90.0 (0.8)
∞	50.0 (0.0)	66.7 (4.8)	69.2 (2.6)	29.6 (6.4)	4.8 (3.9)	84.1 (4.8)	90.0 (0.9)

Table 9: Results of parsing morpheme sequences to length upper-bound (LUB) with a morphological-analyzer applied for obtaining a sublexicon for unknown and once-occurring morphemes. No word-boundaries used. The averages and standard deviations are taken over a 5-fold cross-validation experiment.

of about 1.5% LB recall and precision on all sentences, and even 3% on shorter sentences; a similar level of improvement can be seen on POS tagging recall and precision; and even a 3-5% improvement on non-crossing parse-trees (on sentence up to ten morphemes there is a 13% improvement, mostly due the small number of such sentences). These improvements come at the price of a slight degradation in tree exact-match (0.5-0.8% degradation corresponds to, on average, less than a “half parse-tree” in a test-set). Although, the improvement in e.g. LB recall does not test significant even at e.g. $p = 0.1$ level (in a paired t-test we have $t(4) = 1.5$), we feel it is still encouraging that there is chance of improvement using such simple means⁶. In any case, next we will see that supplying only the correct POS tag per morpheme in the input of the parser improves the results significantly.

4.6 Input: Correct POS tag-sequences only; Output: Parses

LUB	# sen.s	LBR	LBP	No Cros.	Ex. match
10	9.4 (2.0)	86.5 (4.3)	86.9 (5.1)	75.8 (6.4)	39.0 (21.3)
20	25.4 (4.5)	83.0 (2.2)	83.0 (2.4)	50.2 (5.6)	19.4 (9.9)
30	39.0 (4.9)	79.3 (1.0)	79.8 (1.2)	37.7 (3.8)	14.3 (7.0)
40	44.8 (2.6)	78.5 (1.6)	79.1 (0.77)	33.5 (3.7)	12.6 (6.4)
∞	50.0 (0.0)	76.7 (2.2)	76.4 (1.8)	30.0 (3.7)	11.6 (6.2)

Table 10: Results of parsing POS tag sequences without word-boundaries to length upper-bound (LUB): averages and standard deviations over 5-fold cross-validation.

⁶Significance testing at the $p\%$ level implies a $p\%$ probability for the difference in results actually occurring given the null hypothesis (i.e. that the two settings are actually “the same”). However, in our case, it seems even enough to *suspect* that the better result is not due to chance but due to a better system, in order for it to better qualify for use in semi-automatic annotation; this is especially because the “price” – means, time and space – for acquiring the two systems is identical.

In this experiment we evaluate the utility of the combination of the Tree-gram parser with the small tree-bank on the sole task of parsing POS tag sequences. Hence, we assume that a stage of morphological analysis exists, possibly with human intervention, which segments the words into morphemes and provides a single (correct) POS tag per morpheme. The parser is expected to deliver syntactic parses *including the feature annotation of each phrasal-label*. However, the parser is evaluated only on the phrasal-labels excluding the features. We ran two experiments, once with and once without the word-boundaries. The experiment with the word-boundaries resulted in slightly lower LB recall (2%).

Training procedure: Again, the parser was trained on each of the five training-sets (separately, of course) and applied to the corresponding test-set sentences. The upper-bound on the depth of a Tree-gram was set at 5 and the upper-bound on the number of incomplete nodes was also set at 5.

Unknown POS tags (a nearest neighbor strategy): There were on average only 5 unknown POS tags per test-set (containing on average about 1110 POS tag occurrences, i.e. sentence length about 22.4 POS tags). This time, instead of assuming all possible POS tags, we employed a “nearest neighbor” strategy: when a POS tag was unknown, we compared it to all known POS tags according to a simple distance measure and substituted the single “closest” known POS tag and provided that as input to the parser. The distance measure is very simple and consists of two steps: reordering and distance-computation:

Reordering: the POS tag components were reordered into the following rank order (from more important to less important): category, gender, number, person, tense and definiteness. This order has been selected (almost) ad hoc and has no theoretical justification. A better reordering can be applied taking the information-gain of each feature into consideration as done in e.g. [Daelemans et al., 1997].

Distance-computation: The distance between two such ordered sequences is the total weighted distance on each component. If the two values of the same component (category or feature) in the two ordered sequences are exactly the same, the distance is zero. Otherwise, the distance is a function of the inverse of the rank order of the component, i.e. disagreement on a more important component results in larger distance than disagreement on a less important one.

Empirical results: Table 10 lists the results of this experiment. The table shows per sentence length upper-bound (10- ∞), the average (std) number of sentences among the 50 test-sentences, and the other evaluation measures. For all sentences, an average LB recall/precision of 76.7%/76.4% is achieved, i.e. around 3/4 of all constituents (excluding the root node which is trivial) of a parse-tree are exactly the same (brackets and label including features) as in the test-set. Furthermore, about 30% of the parser-output trees did not cross at all with the test-set trees and (only) 11.6% matched exactly. For sentence length upper-bound 30 POS tags about 78% of all sentences are included and the parser delivers around the 80% LB recall and precision. As the sentence upper-bound is lowered, the success rate rises (for example, about 39% match exactly and 75% do not cross at all when we consider only sentences of length 10 POS tags or less).

It is clear that the mean results of LB recall and precision in this experiment are much better than the preceding one (respectively about 10% and 7% improvement on all sentences). In a paired t-test, e.g. the mean LB recall in this experiment is significantly higher than the respective mean in the preceding experiment (i.e. with sublexicon “treatment” of the unknown and once-occurring morphemes), $t(4) = 6.99$, $p < 0.01$. Hence, providing the correct POS tags to the parser reduces the amount of ambiguity significantly. Furthermore, this reduces the sparse-data effects, e.g. the number of unknown and once-occurring POS tags is much smaller than the number respective numbers for morphemes. The cost of this level of performance, however, is larger: unambiguous, (manually) corrected POS tags must be provided for every sequence of morphemes.

4.7 Discussion

The preceding experiments warrant the following conclusion with respect to the utility of the small tree-bank (training only on 448 parse-trees):

1. the tree-bank allows probabilistic language models to perform syntactic parsing of POS tag sequences with 76% of all constituents being retrieved correctly,
2. the tree-bank allows probabilistic language models combined with an ambiguous, imperfect lexical source (in the form of a morphological analyzer), to perform syntactic parsing of morpheme-sequences with 65-70% of the constituents and 83-89% of the POS tags being retrieved correctly.

We find it encouraging that such a small tree-bank of morphologically and syntactically analyzed sentences allows to achieve these results on new unseen morpheme/POS tag sequences. Note that in the three sets of experiments, the accuracy of the Tree-gram parser consistently increases with the amount of the input annotation. This suggests that despite the small size of the corpus, it provides a reliable indication of the behavior of the parser on larger datasets. The parser can therefore be useful as a tool for semi-automatic annotation as discussed in the next section.

5 Semi-automatic annotation methods

The possibility of parsing morpheme-sequences seems the most attractive alternative, because it assumes a less complex input but achieves relatively good results. For parsing morpheme-sequences, we assumed: (1) that the words are correctly segmented into morphemes, which is a major element of Hebrew language analysis, and (2) that the tree-bank contains a detailed analysis of the morphemes, thereby exposing the major features that demand agreement at the syntactic level. While the first assumption has a reasonable expected manual cost, the second one is truly demanding.

For the next stage of the project, however, the second assumption (i.e. feature-annotation) may be relaxed considerably through allowing the syntactic parser to suggest syntactic annotations (including features) followed by manual correction. When parsing morpheme-sequences of length up to 30 morphemes (about 80% of all sentences), the POS tags will be expected to be correct (including the features) about 90% of the time, while constituent nodes will be correct about 70-75% of the time. Hence, manual correction of the POS tags can be followed by a simple automatic procedure for correcting the features up at the constituent nodes

accordingly (heuristic rules can be developed for this task with reasonable accuracy). The utility of this semi-automatic annotation process will depend mostly on the availability of annotation tools that provide an easy environment for tree-correction (e.g. POS tag correction, node-label correction, bracket correction).

It is important to stress that it is easily possible to improve on our results with a few simple modifications:

Probabilities from morphological analyzers: In the current implementation, we assumed uniform distributions over a set of POS tags for the unknown morphemes. If the morphological analyzer could provide a probability estimate $P(\text{morpheme}|\text{POS-tag})$, however, this probability can be combined with the probabilities of the Tree-gram model in a simple fashion. This probability encodes lexical knowledge which the parser can not acquire from a small tree-bank.

Percolation of features and backoff: It is possible to percolate the morphological features up the parse-trees for providing stronger symbolic means for agreement. For this, one would need better implementations of the Katz backoff. Rather than implementing the backoff over the features in a single step (as we currently do), one could allow a multiple-step backoff, each time on another feature. This seems a practical method for the estimation of probabilities of feature-grammars without entering the zone of complex and time-consuming training methods that demand extensive training material.

Manual sentence segmentation: It is clear that for very long morpheme-sequences (longer than 40 morphemes) the performance of parsing and POS tagging degrades. It is expected that long sentences are combinations of two or more shorter sentences (using conjunction, relative clauses, sentential complements etc.). Manual segmentation of the long sentences (about 10% of all sentences) is not an especially demanding task but could enhance the parser's labeled-bracketing results by about 10-15%.

To summarize, optimally, it seems that the next phase of semi-automatic tree-bank annotation should consist of the following manual activities:

- word segmentation into morphemes, possibly by only correcting the output of the Hebrew morphological analyzer.
- long sentence (over 30 morphemes) segmentation into sub-sentences at the S level by placing brackets.
- correction of the parse-trees output by the joint-system of probabilistic parser and morphological analyzer.

Semi-automatic annotation will take place by cycles: a new sequence of sentences (say 500) are automatically annotated and manually corrected; the parser (and possibly the Hebrew morphological analyzer) is trained on the union of the newly acquired tree-bank and the old tree-bank, and the cycle of annotation starts again. At a certain point, it might be possible to use automatic methods for sampling sentences from the corpus, that have a particularly high expected utility for the parser's accuracy, [Engelson and Dagan, 1996].

standard	?	b	g	d	h	v	z	x	t	i	k	l	m	n	s	@	p	c	q	r	š	t
adopted	a	b	g	d	h	v	z	x	t	i	k	l	m	n	s	e	p	c	q	r	f	t

Table 11: transcription of Hebrew letters

o	u	yyCM	yyCLN	yyLRB	yyQUOT	yyDOT
%	”	,	:	(”	.
yyDASH	yyRRB	yyEXCL	yyQM	yySCLN	yyELPS	
-)	!	?	;	...	

Table 12: transcription of symbols

6 Conclusions

Manual morpho-syntactic annotation of a Hebrew corpus is a demanding task. Compared to the annotation of English corpora, an additional bottleneck in the process of annotating Hebrew corpora involves morphological analysis of words and its systematic translation into unambiguous sequences of POS tags. In this paper we reported on the construction of a small Hebrew tree-bank and on experiments performed on this tree-bank. The results suggest that the utility of a well-annotated small tree-bank for training probabilistic parsers is larger than initially expected and that a parser that is generated in this way can be useful for the annotation of the corpus. In this sense, the general learning model of Tree-gram parsing is useful for specific problems that emerge in Hebrew due to the absence of a robust parser that could help in the annotation process. Moreover, we have seen that dedicated morphological tools are useful for the initial processing of the input: the experiments described in section 4 indicate that the parsing results improve dramatically when the morphological analyzer complements the parser’s sublexicon on unknown and low-frequency morphemes. We believe that these conclusions hold for a variety of morphologically rich languages, and are therefore of general importance for corpus linguistics and statistical natural language processing.

A Appendix The writing system in the corpus

The writing system used in the corpus is a non-standard transcription of Hebrew letters into Latin letters, according to table 11. Additional symbols are denoted as in table 12. The reason for this special notation is merely technical, since some of the tools we used expect only English letters.

References

- [Albeck, 1992] Albeck, O. (1992). *Formal analysis by a restriction grammar on one of the stages of Modern Hebrew*. Israel Science and Technology Ministry. In Hebrew.
- [Bemova et al., 1999] Bemova, A., Hajic, J., Hladka, B., and J.Panevova (1999). Morphological and syntactic tagging of the Prague Dependency Treebank. In *Proceedings of ATALA Workshop*.

- [Ben-Tur et al., 1992] Ben-Tur, E., Angel, A., Ben-Ari, D., and Lavi, A. (1992). *Computerized analysis of Hebrew words*. Israel Science and Technology Ministry. In Hebrew.
- [Black et al., 1991] Black et al., E. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*.
- [Bod, 1992] Bod, R. (1992). A computational model of language performance: Data Oriented Parsing. In *Proceedings COLING'92*, Nantes.
- [Bod, 1995] Bod, R. (1995). *Enriching Linguistics with Statistics: Performance models of Natural Language*. PhD thesis, ILLC-dissertation series 1995-14, University of Amsterdam.
- [Bonnema, 1997] Bonnema, R. (1997). Data oriented semantics. Master's thesis, University of Amsterdam.
- [Borer, 1984] Borer, H. (1984). *Parametric Syntax: Case Studies in Semitic and Romance Languages*. Foris, Dordrecht.
- [Brill, 1995] Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistic*, 21:784–789.
- [Charniak, 1999] Charniak, E. (1999). A maximum-entropy-inspired parser. In *Report CS-99-12*, Providence, Rhode Island.
- [Chen and Goodman, 1998] Chen, S. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. In *Technical report TR-10-98*, Harvard University.
- [Choueka,] Choueka, Y. Rav millim. Technical report, The Center for Educational Technology in Israel. <http://www.cet.ac.il/rav-milim/>.
- [Choueka and Lusignan, 1985] Choueka, Y. and Lusignan, S. (1985). Disambiguation by short context. *Computers and the Humanities*, 19(3).
- [Collins, 1997] Collins, M. (1997). Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the EACL*, pages 16–23, Madrid, Spain.
- [Daelemans et al., 1997] Daelemans, W., Van den Bosch, A., and Weijters, A. (1997). Igtree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.
- [Engelhardt, 1999] Engelhardt, M. (1999). *The Syntax of Nominalized Properties*. PhD thesis, Hebrew University.
- [Engelson and Dagan, 1996] Engelson, S. and Dagan, I. (1996). Minimizing manual annotation cost in supervised training from corpora. In *Proceedings of the 34th Annual Meeting of the ACL (ACL'96)*, pages 319–326.
- [Good, 1953] Good, I. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264.

- [Hajic and Hladk, 1998] Hajic, J. and Hladk, B. (1998). Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 36th Annual Meeting of the ACL and the 17th ICCL*.
- [Katz, 1987] Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3).
- [Kurohashi and Nagao, 1998] Kurohashi, S. and Nagao, M. (1998). Building a Japanese parsed corpus while improving the parsing system. In *Proc. of The First International Conference on Language Resources and Evaluation*.
- [Levinger, 1992] Levinger, M. (1992). Morphological disambiguation in Hebrew. Master's thesis, Computer Science Department, Technion, Haifa, Israel. In Hebrew.
- [Levinger et al., 1995] Levinger, M., Ornan, U., and Itai, A. (1995). Morphological disambiguation in Hebrew using a priori probabilities. *Computational Linguistics*, 21:383–404.
- [Magerman, 1995] Magerman, D. M. (1995). Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33th Annual Meeting of the ACL*.
- [Manning and Schütze, 1999] Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.
- [Marcus et al., 1993] Marcus, M., Santorini, B., and Marcinkiewicz, M. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).
- [Oeder and Ney, 1993] Oeder, M. and Ney, H. (1993). Word graphs: An efficient interface between continuous-speech recognition and language understanding. In *ICASSP Volume 2*, pages 119–122.
- [Scha, 1990] Scha, R. (1990). Language Theory and Language Technology; Competence and Performance (in Dutch). In de Kort, Q. and Leerdam, G., editors, *Computertoepassingen in de Neerlandistiek*, Almere: LVVN-jaarboek.
- [Scha et al., 1999] Scha, R., Bod, R., and Sima'an, K. (1999). Memory-Based Syntactic Processing. *Special Issue on Memory-Based Processing, Journal of Empirical and Theoretical Artificial Intelligence (JETAI)*, 11 (3).
- [Segal, 2000] Segal, E. (2000). Hebrew morphological analyzer for Hebrew undotted texts. Master's thesis, Computer Science Department, Technion, Haifa, Israel. <http://www.cs.technion.ac.il/~erelsgl/bxi/hmntx/teud.html>.
- [Sima'an, 1996] Sima'an, K. (1996). Computational Complexity of Probabilistic Disambiguation by means of Tree Grammars. In *Proceedings of COLING'96*, volume 2, pages 1175–1180, Copenhagen, Denmark.
- [Sima'an, 1999] Sima'an, K. (1999). *Learning Efficient Disambiguation*. A PhD dissertation. ILLC dissertation series 1999-02 (Utrecht University / University of Amsterdam), Amsterdam.

- [Sima'an, 2000] Sima'an, K. (2000). Tree-gram Parsing: Lexical Dependencies and Structural Relations. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, pages 53–60, Hong Kong, China.
- [Tur et al., 1999] Tur, D. H., Ofazzer, K., and Tur, G. (1999). Design for a Turkish treebank. In *Linguistically Interpreted Corpora: EACL Post-conference workshop*.