Building a Tree-Bank of Modern Hebrew Text

Khalil Sima'an* — **Alon Itai**** — **Yoad Winter**** —
Alon Altman** — **Noa Nativ*****

* *Induction of Linguistic Knowledge, Tilburg University*
and Computational Linguistics, University of Amsterdam
Spuistraat 134, 1012 VB Amsterdam, The Netherlands

khalil.simaan@hum.uva.nl

** *Department of Computer Science*
Technion – Israel Institute of Technology,
Haifa 32000, Israel

{*itai,winter*}*@cs.technion.ac.il*

*** *Hebrew University, English and Computer Science Departments*

ABSTRACT. *This paper describes the process of building the first tree-bank for Modern Hebrew texts. 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, the joint utility of an automatic morphological analyzer, a probabilistic parser and a small manually annotated tree-bank was explored.*
An initial tree-bank that consists of 500 annotated sentences from a daily newspaper is described. The annotation scheme that underlies the analyses in the tree-bank integrates morphology and syntax. An existing morphological analyzer and a language-independent probabilistic parser were applied to this tree-bank. Based on the results of some experiments with these tools, a semi-automatic procedure for future enlargement of the tree-bank is outlined. This procedure starts out with the manual segmentation of words into morpheme sequences, then continues with automatic POS tagging and parsing of these morpheme sequences, followed by manual correction of the resulting parse-trees. The proposed procedure is expected to reduce the cost of the next annotation stages in this corpus.

RSUM. *A dfinir par la commande* `\resume{...}`

KEYWORDS: *Modern Hebrew, Corpus, Tree-Bank, Syntactic Analysis, Morphological Analysis, Probabilistic Parsing, Semi-Automatic Annotation*

MOTS-CLS : *A dfinir par la commande* `\motscles{...}`

1. Introduction

In recent years, the availability of corpora and linguistically annotated language resources sparked new momentum into research towards computational solutions for the ambiguity problem in language processing. Among these resources, tree-banks have proved especially useful for the development of part of speech taggers, parsers and other modules for language processing. Unfortunately, this development has been restricted mostly to languages with large communities or to communities with suitable resources for corpus collection and linguistic annotation. The cost of developing such resources is by large the main prohibitive factor. This is one of the reasons that for Modern Hebrew (as well as many other languages) large annotated corpora are not currently 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 acquisition of an initial environment that allows us to reduce the manual annotation cost by applying automatic language processing modules in the annotation loop. A serious problem in achieving this goal is the absence of reliable, broad-coverage parsers for Hebrew. The alternative, general purpose probabilistic parsers, is also not directly applicable in this case, because these require an initial training tree-bank. To overcome this "deadlock", this work explores the utility of a small Hebrew tree-bank for developing a tool for further semi-automatic morphological and syntactic annotation.

The morphology of Modern Hebrew is quite complex. Hebrew words often consist of more than one morpheme and include various affixes for agreement, pronouns, prepositions and other linguistic items. Therefore, considerable ambiguity exists already at the word level. This situation is further aggravated in the written language since the common writing system of Hebrew omits most vowels and other phonetic disambiguation clues. Hence, morphological analysis is a principal element of any sentence processing system for Modern Hebrew. We used an earlier work by [SEG 00], who developed a morphological analyzer for Hebrew using a small set of manually analyzed words, complemented with an automatic learning scheme and heuristic rules.

The annotation was done by manually correcting the analysis chosen by the morphological analyzer, followed by manual syntactic annotation. The result is a small tree-bank of 500 sentences featuring a single annotation scheme, where morphology and syntax are integrated into parse-trees. The most costly task in this process is the syntactic annotation. When expanding this small tree-bank, this cost would be significantly reduced if the present tree-bank allowed us to train a usefully accurate probabilistic parser. We address this question by experimenting with various settings for utilizing the probabilistic *Tree-Gram parsing model* of [SIM 00] and Segal's morphological analyzer of Hebrew, for the future semi-automatic annotation of new texts from the same domain. In each setting, we allowed a different amount of detail of morphological analysis to accompany the sentence that is fed as input to the probabilistic parser, varying from a mere segmentation of words into morphemes to a full fledged morphological analysis, including part of speech (POS) tagging. These settings simulate some of the viable alternatives for proceeding with the semi-automatic annotation of new text. We report encouraging empirical results with some of these settings and select one particular setting as the most promising. This setting cosists of three steps: (1) manual segmentation of the input words into morphemes, (2) POS tagging and parsing of the resulting morpheme sequences using the probabilistic parser with help of the morphological analyser, and (3) manual correction of the resulting parse-trees. The experiments exhibit the usefulness 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, summarizes the annotation process and gives some figures about the tree-bank. Section 4 reports on the experiments with the application of the probabilistic Tree-Gram parser to the small tree-bank. Section 5 presents our conclusions towards further semi-automatic annotation of new material.

2. Hebrew morphological analysis – background

2.1. *The problem*

Morphological ambiguity is prevalent in written Modern Hebrew. Most Hebrew texts are unvocalized. This means that most vowels are totally missing from the texts. The rich agreement system of the language, the fact that many affixes (such as agreement, prepositions, determiners, 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 1. The data was obtained by analyzing 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] (=“and-when-I-saw”), pronounced *ukšera’iti* or *vekšera’iti*, 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’).

The above string is unambiguous. But most Hebrew words have more than one analysis. As an example for a multiply ambiguous word, string *lmwrh*, which has (at least) the following possible analyses:

– Preposition *l*=“to” + feminine noun *mwrh*=“teacher”:
“to a female teacher”

– Preposition *l*=“to” + masculine noun *mwrh*=“teacher”:
“to a male teacher”

– Preposition *l*=“to” + definiteness marker *h* (unvocalized) + feminine noun *mwrh*=“teacher”:
“to the female teacher”

– Preposition *l*=“to” + definiteness marker *h* (unvocalized) + masculine noun *mwrh*=“teacher”:
“to the male teacher”

– Noun *lmwr*=*lemur* + female suffix *h* :
“a female lemur”

– Noun *lmwr*=*lemur* + possessive feminine suffix *h* :
“her male lemur”

2.2. *Existing morphological analyzers for Hebrew*

2.2.1. *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

1. In 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 regularity of Hebrew morphology, context-independent morphological analysis is feasible, and both academic and commercial analyzers are available.

Among the commercial systems, most noteworthy is the *Rav Millim* commercial analyzer.[2] Within a Machine Translation project, IBM Haifa Scientific Center, developed a morphological analyzer [BEN 92] 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 [SEG 00].

2.2.2. *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 [CHO 85] 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 is the correct one in all their co-occurrences. This method leaves open the question of how to automatically choose the correct analysis of the first occurrence of the word in such a pair.

Orly Albeck [ALB 92] 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. [LEV 95] gathered statistics on similar words in order to estimate these distributions. For each analysis of a morphologically ambiguous word they searched a large corpus in order to find words that differ only in one feature (such as gender) from the original analysis. They assumed that the distributions of the original and the perturbated words were the same. Thus, occurrence statistics of the perturbated 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 [LEV 92] enhanced this system with a short context filter that looked for agreement and several other local features.

Segal [SEG 00] 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. The probabilities of lexical entries are estimated using Levinger's "similar words" method. 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 rate. To reduce the error rate, Segal employed two more heuristics:

2. http://www.cet.ac.il/rav-milim/

1. Correction rules were automatically acquired from the corpus by a method resembling Brill's transformation based method [BRI 95]. The program considered 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 considered all possible rules and adopted those rules that improved the performance of the morphological analyzer on the training set.

2. Segal used a rudimentary deterministic parser that used some lexical dependency rules. This part of the system was completely heuristic and was not acquired from the corpus.

For each morphological analysis of a word the system assigned a score, which was initialized to the probabilities given by Segal's version of the similar words method. The two heuristics were then employed to modify the score. The analysis with the largest score was 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 (i.e. 4% error-rate).

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 concerns English (see [MAN 99] for some references). Large tree-banks of English text, notably the Penn Tree-Bank [MAR 93], are available. In corpus-based processing of English, morphological analysis has been of minor concern because it can be assumed that words rather than morphemes constitute the atomic building blocks of sentences.[3] Clearly, statistics over word-occurrence is more prone to sparse-data problems than morpheme occurrence. However, for English, it seems that the existence of large syntactic tree-banks and extended POS tag-sets, combined with the less complex nature of English word-structure, have been instrumental in avoiding the need for a level of morphology in the tree-banks.

Corpora of morphologically rich languages are currently being developed for Czech ([HAJ 98, BEM 99]), Turkish ([TUR 99]) and Japanese [KUR 98]. 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 syn-

3. In some cases, however, impoverished forms of morphological analysis are consulted in order to deal with unknown words (see [COL 97, CHA 99]).

tactic analysis. Currently, there are no tree-banks for Modern Hebrew, but 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 and syntactic analysis.

As discussed in the preceding section, morphological ambiguity is very common in Hebrew texts, and syntactically important morphemes such as 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 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. Consequently, both Hebrew words *lamemšala* ("to-the-government") and *lememšala* ("to-a-government") are spelled *lmmflh*. However, the morphemes in the representation of the word under the first reading are *l-h-mmflh*. 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 'covert morpheme' or using a feature is not always straightforward, and involves considerations of consistency and ease of annotation. For instance, the word *bo*, spelled *bw*, is a prepositional phrase

g (gender):	Z=masculine, N=feminine, 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*

with the meaning "in-him". For reasons of consistency with the tagging of other 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 such as 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' plan"

For linguistic arguments supporting this analysis, see [ENG 99] (but compare with [BOR 84]).

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 *?et* (spelled *at*) which appears as a separate word in written Hebrew. For example:

finh/VN-ZY3V	at/AT	h/H	mdiniwt/NN-NY
changed	ACC	the	policy

3. **CDT**: Numerals in determiner position. These 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".

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 *rak* ("only") and *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.

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*

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

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*

set are the following:

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 pre-posed 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 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

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*

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 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 is used as their node mother 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 only in the present (e.g. *hu=hwa*="is"), which is the unmarked tense in Hebrew. The distribution of the two particles is quite different. This 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". A parallel example with a present tense copula would be ungrammatical.

```
(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*

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 $Y_1, ..., Y_n$. 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 different than CC (coordinating conjunction) is a head constituent of X.

3. Otherwise: if $X, Y_1, ..., Y_n$ are all of the same category then $Y_1, ..., Y_n$ 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 apposition (‘*tmura*’) constructions. 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 a definite whereas in (3b) it is indefinite, although the head (the second NP) is indefinite in both cases.

(3) a. (NP-NY-H (NNT-NY nbxrt) (NP-ZY-H (H h) (NN-ZY nwer)))
team (construct state) the-youth
“the youth team”

b.

(NP-ZR (NNT-ZR ciwni)	(NP (NN-NY drk)))
marks (construct state)	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 shows agreement with this subject. *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 the three kinds of empty categories.

(4)

rpwbliqaim/NN-ZR	h/REL	*T*/-NONE-	nwjim/MD-ZRAH
republicans	who	*T*	tend

lhskim/VB-M	at/IN	hm/PRP-ZR3
to-agree	with	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
Tukey	received	only	52

alp/CDT-BR	*NONE*/-NONE-
thousand	*NONE*

“Tukey received only 52 thousand (dollar)”

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 [SEG 00], which was used to obtain a preliminary segmentation, POS tags and agreement features of words in the corpus.

– The SEMTAGS graphical tool of Bonnema [BON 97], 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 [SEG 00] into a preliminary POS tag set. This translation is not one-to-one, because the tag set that is used in Segal’s 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 input for this stage consisted of morphological annotations of 250 sentences that were manually analyzed by Segal and the choices of his morphological analyzer ([SEG 00]) for the words in additional 250 sentences. These morphological representations were 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 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 manual annotation was done by two students with undergraduate training in Linguistics. 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 [MAR 93] was 10 times higher. We believe that there are two main reasons for this large difference:

– The annotation of the Penn tree-bank relied on much 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 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 shows a graph of 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

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 (context-free) grammar rules	
lexical rules without (with) features	3431 (3515)
once-occurring *lexical* rules without (with) features	2363 (2439)
non-lexical rules without (with) features	1106 (1580)
once-occurring *non-lexical* rules without (with) features	728 (1012)

Table 6. *The tree-bank in numbers*

“h” (the), various prepositions e.g. “b” (in) and “l” (to) and punctuation marks (e.g. the comma).

Other interesting 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[4] 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 constitute 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.

4. Statistical standard deviation.

Figure 4 shows 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 some other 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 Context-Free Grammar rules which constitute the parse-trees are 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 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 (unary rules) to ten; the number of different rules per rhs length is respectively as follows: 78, 296, 290, 204, 139, 49, 37, 6, 5, 1. With so many different rules containing 2-6 symbols on the right-hand side, there is reasonable chance that many more will appear in parse-trees of future, novel sentences.

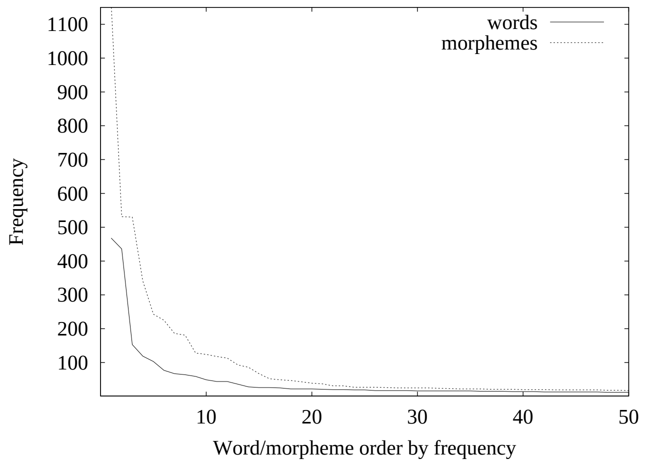

Figure 3. *Frequency counts of the 50 most frequent words/morphemes*

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 and syntactic analysis. 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

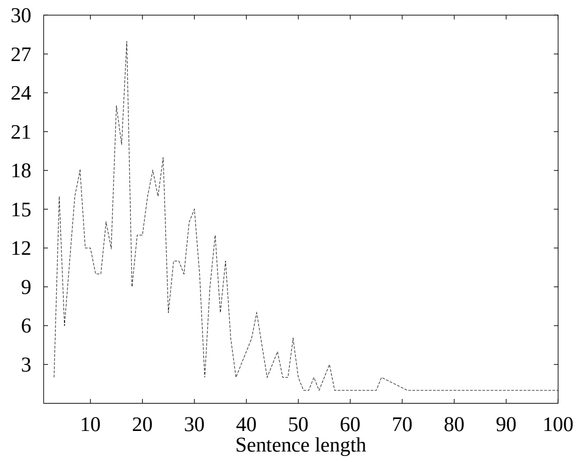

Figure 4. *Counts of sentences as a function of their length*

the currently available tree-bank of 498 analyzed sentences in order to obtain a more efficient *semi-automatic* annotation process.

The small size of the tree-bank raises at least two problems for probabilistic parsers. In the first place, the notorious sparse-data problem: because of the small tree-bank, many phenomena will not have occurred in it, and the distribution of phenomena that did occur will probably not be representative. Once occurring phenomena provide good examples of this sparseness problem. As Table 6 shows, once-occurring morphemes (67% of all morphemes) are dramatically less ambiguously represented at the POS tag level than other morphemes, and the many once-occurring grammar rules (66%) signify, most probably, non-representative distributions over syntactic structures, but possibly also annotation errors. As a result, any probabilistic parser that can be generated using this corpus is expected to suffer from coverage problems. 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 future, unseen 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 [SIM 00] and the morphological analyzer of Segal [SEG 00]. 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, will be very useful for semi-automatic syntactic annotation of a larger Hebrew corpus.

4.1. *Tree-gram probabilistic parsing*

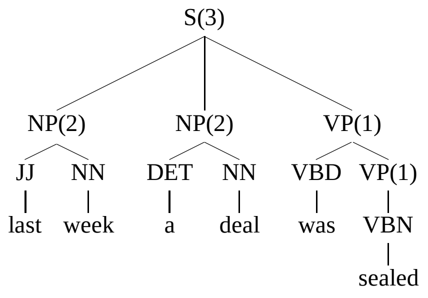

Figure 5. *An example parse-tree. The number of the head-child of a node is specified between brackets (e.g. the third child (VP) of the node labeled S carries the head).*

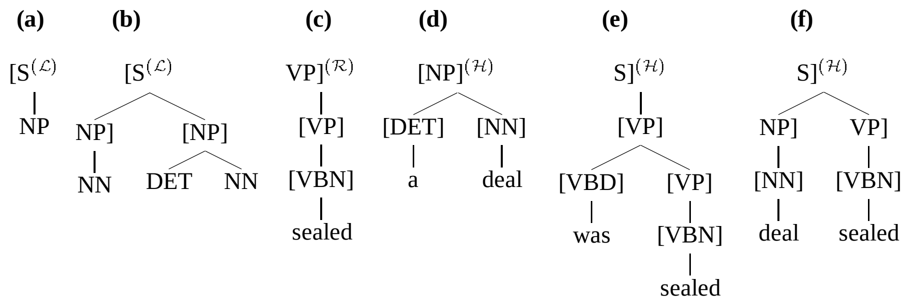

Figure 6. *Some T-grams extracted from the tree in figure 5: the superscript on the root label specifies the* T-gram role,. *e.g. the left-most T-gram is in the LEFT role. Non-leaf nodes are marked with "[" (left-STOP) and "]" (right-STOP) to specify whether they are complete from the left/right or both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

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, a tree-bank is used for inducing probabilistic grammars e.g. [SCH 90, BOD 92, MAG 95, BOD 95, COL 97, CHA 99, SIM 00].

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 $\langle rule, history \rangle$ 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

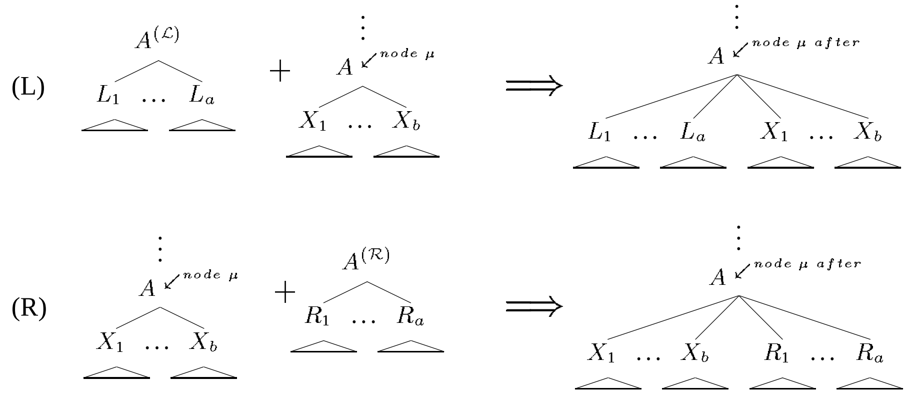

Figure 7. *A T-gram is generated by attachment at μ in a partial parse-tree. The T-gram being generated is marked with $\mathcal{L}$ and $\mathcal{R}$ to denote its role. In (L) a LEFT Tree-gram and in (R) a RIGHT Tree-gram is generated. Note that node μ must be non-complete.*

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 [SIM 00], which combines aspects from Data-Oriented Parsing (DOP) [SCH 90, BOD 95, SCH 99, SIM 99] with aspects from Bilexical-Dependency Markov-Grammars [COL 97, CHA 99]. We now provide a brief overview of the Tree-gram model. Further details of the Tree-gram model can be found in [SIM 00].

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 special symbol “STOP” both at its its left and right hand 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 the left or right hand sides of a node (or both), the node is incomplete. In this case, the subtrees that it dominates may be extended with additional 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. When a Tree-gram’s root dominates its head-child[5] (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 [COL 97, SIM 00]).

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 entirely of complete nodes. We assume that the derivations are statistically independent, thus 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 $argmax_d P(S,d)$, rather than the parse-tree T that solves $argmax_T P(S,T)$. We choose to do so, simply because the latter problem is known to be NP-Complete [SIM 96], while the first one is solvable in time cubic in sentence-length e.g. [SIM 99].

4.2. *Tree-gram parsing of the Hebrew tree-bank*

The widely used Penn tree-bank [MAR 93], which is based on English articles from the Wall Street Journal (WSJ), consists of approximately 50,000 syntactically analyzed sentences. The availability of such large annotated corpora for English is the one of the main reasons that most existing parsing models were applied to this language. Morphology does not play a major role in English tree-banks. Evidence

5. The *head child-node* of a given node is the child that carries its head-word.

for this observation comes from various works: many models that were applied to the Penn tree-bank collected sufficiently reliable statistics over joint-occurrences of syntactic and lexical phenomena, where the words are not segmented into morphemes, e.g. [COL 97, CHA 99, SIM 00]. Furthermore, a single probabilistic model is often used for both part-of-speech tagging (POS tagging) and syntactic parsing.

At this stage, our tree-bank is too small (1% of the Penn tree-bank) to allow the full integration of both morphological and syntactic analysis into one model. In fact, it is unclear whether the direct application of existing parsing models to Hebrew, with its complex morphology, can be as successful as for English. Clearly, this question will become relevant only when the size of the tree-bank allows better estimation of language models. Therefore, the first short-term objective of the project is to minimize the cost of the 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 requires additional integration between the morphological analyzer and the Tree-gram parser, which has not been achieved yet.

The following settings are explored 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 consists only of segmentation of the words into morphemes: this may involve human corrections. The parser is applied to morpheme sequences and aims both at POS tagging and parsing. The main problem under this setting is to deal 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 the morphological analyzer is used for helping to determine the unknown morphemes. The analyzer is run on some text from the corpus. The morphemes and POS tags that it produces are gathered in a “sublexicon” which serves the parser as an ambiguous, imperfect resource on 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. This stage may combine automatic annotation by means of the existing morphological analyzer [SEG 00] together with a human annotator that 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.

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. *Setup of the experiments*

In this section we outline the general setup for the experiments. We highlight the limitations of our current implementation, explain how we deal with sparse-data problems and define the evaluation methodology.

4.3.1. *The non-terminals in the tree-bank:*

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.

4.3.2. *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 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 backoff smoothing model [KAT 87, CHE 98] 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 [GOO 53] 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.

4.3.3. *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.

4.3.4. *Evaluation methodology:*

In each experiment, the trained parser is applied to the sentences found in the corresponding test-set. The parser's output is compared to the test-set trees on the PARSEVAL measures [AL. 91]: labeled bracketing precision and recall. In PARSEVAL, a parse-tree is considered as a set of constituents:[6] 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 in the output of the parser is compared to the corresponding test-set tree T by computing $|P \cap T|$, i.e. the number of matching constituents (brackets and labels)[7]. *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 more measures: the *tree Exact-Match (ExM) 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 (PTP)* (the number of correct tags to total number of tags in parser-output, including features) and recall (PTR) (the 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 assume that the input of the parser is a correct segmentation of words into morphemes, and explore two versions of this segmentation: (1) without marking of word-boundaries, and (2) including word-boundaries. 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

6. POS tags are not included as constituents because they are trivial constituents. POS tagging is evaluated separately.
7. The output parse-trees does 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*.

LU	# sen.	LBR	LBP	No Cros.	ExM	PTR	PTP
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 (LU). The 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 parentheses.*

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

4.4.1. *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 Penn tree-bank). This reduces the total number of POS tags (lexical-categories with feature-sequences) from 199 to only about 30.

4.4.2. *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, 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.

4.4.3. *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 [OED 93]. The parsing of word-graphs is described in [SIM 99].

4.4.4. *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, 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 [SEG 00] on a corpus containing 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*

LU	# sen.	LBR	LBP	No Cros.	ExM	PTR	PTP
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 (LU) 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.[8] For the unknown morphemes that it did not find in the sublexicon, the parser assumed all 199 POS tags. 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 less than 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,

8. Strictly 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.

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 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 set. 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 highly 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, it is reasonable to apply the same strategy to once occurring morphemes. This is what we did in a second experiment maintaining, otherwise, the same setting as the 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 a more (limited) choice which, on average, will prove beneficial. Table 9 shows the results of this experiment.

LU	# sen.	LB Rec.	LB Prec.	No Cros.	ExM	PTR	PTP
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 at most (LU) 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.*

There is an improvement 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 difference in results does not test significant, even at $p = 0.1$ (in a paired t-test we have $t(4) = 1.5$), we believe that it is still encouraging that there is chance of improvement using such simple means[9].

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

LU	# sen.	LBR	LBP	No Cros.	ExM
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 (LU): averages and standard deviations over 5-fold cross-validation.*

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%).

4.6.1. *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.

4.6.2. *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

9. Significance 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 is sufficient 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" for acquiring the two systems is identical.

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. [DAE 97].

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.

4.6.3. *Empirical results:*

Table 10 lists the results of this experiment. The table shows per sentence of length at most (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 labels, excluding the 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 sentences of length at most 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 decreases, the success rate rises.

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 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 of 448 training 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 as its input is made less ambiguous. This suggests that the parser can be expected to achieve better results if its input is further enriched manually with some parsing clues, e.g. brackets on the S and $SBAR$ levels. Therefore, we think that the parser should 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 manually correction. When parsing morpheme-sequences of length up to 30 morphemes (about 80% of all sentences), the POS tags are 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, manually correcting the POS tags can be followed by a simple automatic procedure for correcting the features up at the constituent nodes accordingly; we conjecture that 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 easy 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

$P(morpheme|POStag)$, 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 long sentences, e.g., morpheme-sequences longer than 30 morphemes, the performance results of the parser become significantly worse than for shorter sentences. It is reasonable to assume that long sentences are combinations of two or more shorter sentences (involving constructs such as conjunctions, relative clauses and sentential complements). Segmenting these long sentences (about 20% of all sentences) manually is not a particularly demanding task but could enhance the parser's labeled-bracketing results by about 10%.

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

– Segmentation of words into morphemes, could be possibly done by manually correcting the output of the Hebrew morphological analyzer.[10]

– Segmentation of long sentences (over 30 morphemes) into sub-sentences, e.g. at the S and $SBAR$ levels, by placing (labeled) brackets.

– Manual correction of the parse-trees obtained from the combination of the probabilistic parser and the morphological analyzer.

Semi-automatic annotation will take place in 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 [ENG 96].

10. As the size of the tree-bank grows, we can retrain the analyzer for improving its accuracy and coverage.

standard	?	b	g	d	h	v	z	x	t	i	k	l	m	n	s	@
adopted	a	b	g	d	h	v	z	x	t	i	k	l	m	n	s	e
standard	p	c	q	r	š	t										
adopted	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. In addition to the syntactic work, which is comparable to the annotation of English corpora, the process of annotating Hebrew corpora involves morphological analysis of words and their systematic translation into unambiguous sequences of POS tags. In this paper we described the construction process of a small Hebrew tree-bank and the experiments we performed on this tree-bank. The results suggest that the costly manual syntactic annotation can be significantly aided by using a probabilistic parser. The utility of a well-annotated small tree-bank for training probabilistic parsers is larger than initially expected. 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. 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 interest for corpus linguistics and statistical natural language processing.

A. Appendix I: The writing system in the corpus

In the corpus the writing system used is a non-standard transcription of Hebrew letters into English 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.

Acknowledgements

This work was made possible thanks to a grant by the Israeli Ministry of Science, Culture and Sport, no. 9484–2–98.
The second and third authors were partle supported by the fund for the promotion of reserach at the Technion, research number 120-015.

The participation of the first author in the project is thanks to a visiting grant from the Computer Science Department – Technion and support from The Netherlands Organization for Scientific Research (NWO).

B. References

[AL. 91] BLACK ET AL. E., "A procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars", *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*, 1991.

[ALB 92] ALBECK O., "*Formal analysis by a restriction grammar on one of the stages of Modern Hebrew*", Israel Science and Technology Ministry, 1992, In Hebrew.

[BEM 99] BEMOVA A., HAJIC J., HLADKA B., J.PANEVOVA, "Morphological and Syntactic Tagging of the Prague Dependency Treebank", *Proceedings of ATALA Workshop*, 1999.

[BEN 92] BEN-TUR E., ANGEL A., BEN-ARI D., LAVI A., "*Computerized analysis of Hebrew words*", Israel Science and Technology Ministry, 1992, In Hebrew.

[BOD 92] BOD R., "A Computational Model of Language Performance: Data Oriented Parsing", *Proceedings COLING'92*, Nantes, 1992.

[BOD 95] BOD R., *Enriching Linguistics with Statistics: Performance models of Natural Language*, PhD thesis, ILLC-dissertation series 1995-14, University of Amsterdam, 1995.

[BON 97] BONNEMA R., "Data Oriented Semantics", Master's thesis, University of Amsterdam, 1997.

[BOR 84] BORER H., *Parametric Syntax: Case Studies in Semitic and Romance Languages*, Foris, Dordrecht, 1984.

[BRI 95] BRILL E., "Transformation-Based Error-Driven Learning and Natural Language Processing: A case study in Part-of-speech Tagging", *Computational Linguistic*, vol. 21, 1995, p. 784-789.

[CHA 99] CHARNIAK E., "A Maximum-Entropy-Inspired Parser", *Report CS-99-12*, Providence, Rhode Island, 1999.

[CHE 98] CHEN S., GOODMAN J., "An empirical study of smoothing techniques for language modeling", *Technical report TR-10-98*, Harvard University, August 1998.

[CHO 85] CHOUEKA Y., LUSIGNAN S., "Disambiguation by short context", *Computers and the Humanities*, vol. 19(3), 1985.

[COL 97] COLLINS M., "Three Generative, Lexicalized Models for Statistical Parsing", *Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the EACL*, Madrid, Spain, 1997, p. 16–23.

[DAE 97] DAELEMANS W., VAN DEN BOSCH A., WEIJTERS A., "IGTree: using trees for compression and classification in lazy learning algorithms", *Artificial Intelligence Review*, vol. 11, 1997, p. 407–423.

[ENG 96] ENGELSON S., DAGAN I., “Minimizing Manual Annotation Cost in Supervised Training from Corpora”, *Proceedings of the 34th Annual Meeting of the ACL (ACL’96)*, 1996, p. 319–326.

[ENG 99] ENGELHARDT M., “The Syntax of Nominalized Properties”, PhD thesis, Hebrew University, 1999.

[GOO 53] GOOD I., “The population frequencies of species and the estimation of population parameters”, *Biometrika*, vol. 40, 1953, p. 237–264.

[HAJ 98] HAJIC J., HLADK B., “Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset”, *Proceedings of the 36th Annual Meeting of the ACL and the 17th ICCL*, 1998.

[KAT 87] KATZ S., “Estimation of probabilities from sparse data for the language model component of a speech recognizer”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35(3), 1987.

[KUR 98] KUROHASHI S., NAGAO M., “Building a Japanese Parsed Corpus while Improving the Parsing System”, *Proc. of The First International Conference on Language Resources and Evaluation*, 1998.

[LEV 92] LEVINGER M., “Morphological disambiguation in Hebrew”, Master’s thesis, Computer Science Department, Technion, Haifa, Israel, 1992, In Hebrew.

[LEV 95] LEVINGER M., ORNAN U., ITAI A., “Morphological Disambiguation in Hebrew Using A Priori Probabilities”, *Computational Linguistics*, vol. 21, 1995, p. 383-404.

[MAG 95] MAGERMAN D. M., “Statistical Decision-Tree Models for Parsing”, *Proceedings of the* 33^d *Annual Meeting of the ACL*, 1995.

[MAN 99] MANNING C. D., SCHÜTZE H., *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, Massachusetts, 1999.

[MAR 93] MARCUS M., SANTORINI B., MARCINKIEWICZ M., “Building a large annotated corpus of English: The Penn treebank”, *Computational Linguistics*, vol. 19, 1993, p. 313–330.

[OED 93] OEDER M., NEY H., “Word graphs: An efficient interface between continuous-speech recognition and language understanding”, *ICASSP Volume 2*, 1993, p. 119–122.

[SCH 90] SCHA R., “Language Theory and Language Technology; Competence and Performance (originally in Dutch)”, DE KORT Q., LEERDAM G., Eds., *Computertoepassingen in de Neerlandistiek*, Almere: LVVN-jaarboek (http://www.hum.uva.nl/computerlinguistiek/scha/IAAA/rs/cv.html#Linguistics), 1990.

[SCH 99] SCHA R., BOD R., SIMA’AN K., “Memory-Based Syntactic Processing”, *Special Issue on Memory-Based Processing, Journal of Empirical and Theoretical Artificial Intelligence (JETAI)*, vol. 11 (3), 1999.

[SEG 00] SEGAL E., “Hebrew Morphological Analyzer for Hebrew undotted texts”, Master’s thesis, Computer Science Department, Technion, Haifa, Israel, 2000, http://www.cs.technion.ac.il/∼erelsgl/bxi/hmntx/teud.html.

[SIM 96] SIMA’AN K., “Computational Complexity of Probabilistic Disambiguation by means of Tree Grammars”, *Proceedings of COLING’96*, vol. 2, Copenhagen, Denmark, August 1996, p. 1175–1180.

[SIM 99] SIMA’AN K., *Learning Efficient Disambiguation*, A PhD dissertation. ILLC dissertation series 1999-02 (Utrecht University / University of Amsterdam), Amsterdam, March 1999.

[SIM 00] SIMA'AN K., "Tree-gram Parsing: Lexical Dependencies and Structual Relations", *Proceedings of the* 38^{th} *Annual Meeting of the Association for Computational Linguistics (ACL'00)*, Hong Kong, China, 2000, p. 53–60.

[TUR 99] TUR D. H., OFLAZER K., TUR G., "Design for a Turkish Treebank", *Linguistically Interpreted Corpora: EACL Post-conference workshop*, 1999.