A Statistical Syntactic Disambiguation Program and what it learns

Murat Ersan and Eugene Charniak

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-95-29
October 1995

A Statistical Syntactic Disambiguation Program and What It Learns

Murat Ersan and Eugene Charniak*
Department of Computer Science, Brown University
Providence, RI 02912-1910

October 10, 1995

Abstract

We describe a program that uses statistical information on word-usage to perform syntactic disambiguation, and show that the use of this information significantly improves performance. The bulk of the paper, however, attempts to answer the question: what did the program learn that would account for this improvement? We show that the program has learned many linguistically recognized forms of lexical information, particularly verb case frames and prepositional preferences for nouns and adjectives. We also show that viewed simply as a learner of lexical information the program is also a success, performing slightly better than hand-crafted learning programs for the same tasks.

1 Introduction

Work in statistical language processing typically gathers statistics on some aspects of natural language use, applies the statistics to a natural-language task, and then shows that using the statistics improves task performance. Thus such research fits within a broad construction of language-learning work — a program is exposed to some data and improves its performance thereby. From a learning point of view, however, one is left less than completely satisfied: if this is learning, it seems to be of a trivial sort, even though the tasks and programs may be complicated indeed.

*This research was supported in part by NSF grant IRI-9319516.

Some of the reasons behind the dissatisfaction can be exposed by asking exactly what the program has learned. There are simple answers to this question. Obviously the program learned how to perform some task better; or again, it learned the particular statistical properties of language decreed by the model employed. But these answers seem to beg the question. Is there some non-trivial way to characterize what has been learned?

In this paper we describe a particular program that gathers statistics on word usage and uses these statistics to help syntactically disambiguate sentences. We briefly describe this program and how the statistics help it pick the correct syntactic structure. However, the bulk of the paper is an attempt to give a nontrivial description of what the program has learned. We show that it has learned many of the sorts of lexical facts linguistics knows and loves: verb case frames and noun/adjective prepositional preferences in particular. Furthermore, we show that, simply viewed as a learning program, its performance is comparable to or perhaps slightly better than the best special-purpose verb case-frame learner of which we are aware. (We know of no programs for extracting noun/adjective preposition preferences with which to compare our system.)

2 Statistical Syntactic Disambiguation

English sentences typically have many legal syntactic parses. The problem of syntactic disambiguation is picking from these the parse intended by the author. In this section we describe a system for syntactic disambiguation. We keep the description simple because our concern here is not the system per se, but rather what the system learned that made possible its improved performance. See [6] for a more complete description of the program and its underlying theory.

From a statistical point of view syntactic disambiguation is formalized as finding the most probable parse of the sentence. One comparatively simple version of this ideas uses the notion of a probabilistic context-free grammar (PCFG). A PCFG is a grammar in which each rule has an associated probability that is to be interpreted as the probability of using that particular rule given that one needs to expand a constituent of the corresponding type (i.e., the probability of using $\mathsf{vp} \rightarrow \mathsf{v}\ \mathsf{np}$ given one needs to expand a vp). The probability of a complete parse in such a system is simply the product of the probabilities of all the rules used in the parse. There are efficient algorithms for finding the parse with the highest probability, and one might

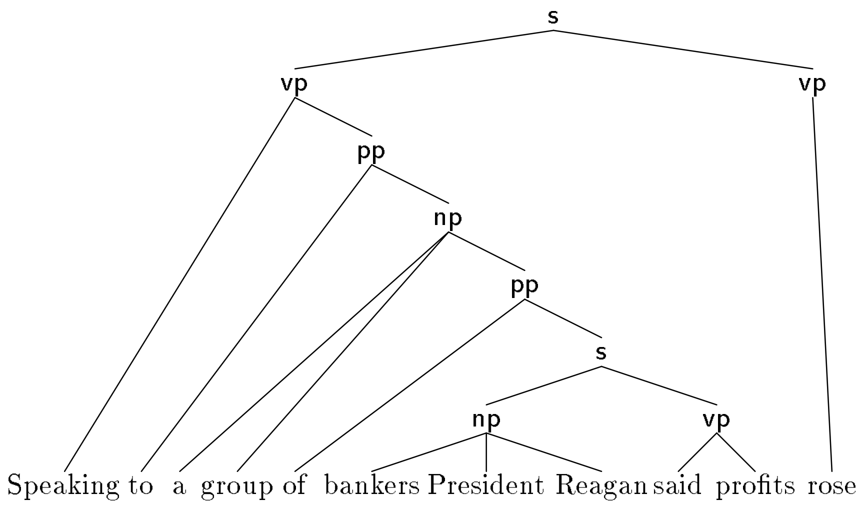

Figure 1: A sentence badly parsed by a PCFG

then choose that parse as one's best guess for the correct (intended) parse.

Unfortunately, PCFGs serve only to distinguish between common and uncommon constructions. They do not offer much help if things get at all complicated. For example, Figure 1 shows what one of our PCFGs produced as its analysis of the sentence "Speaking to a group of bankers President Reagan said profits rose." While many things are wrong here, let us consider two in particular. First, notice the prepositional phrase (pp), "of bankers President Reagan said profits." A key mistake was to create a pp consisting of the preposition "of" followed by a sentence (the "complement" of the pp). Now some prepositions do take sentences as their complements ("because" is a good example) but "of" hardly ever does so. Unfortunately, our program does not know this. It only knows that in general sentential pp's are less common than nominal ones. Presumably the system would have a better chance of finding the correct parse were it to know, not just the relative probability of pp rules, but their probability for a given preposition, or, as we shall call it, the probability of the rule given the head of the phrase.

Similarly, note in Figure 1 the putative np "bankers President Reagan." We know that while "President" is a plausible modifier of "Reagan," "bankers" is not. Again, if the system knew this it would presumably parse better.

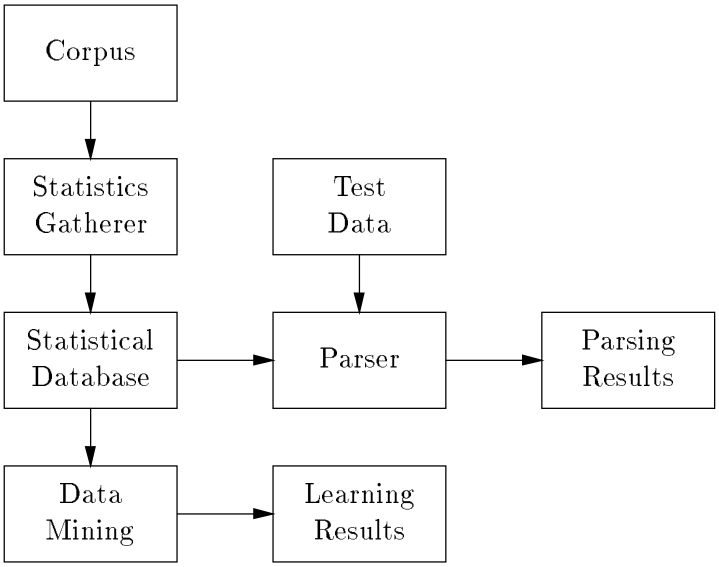

Figure 2: Gathering, using, and analyzing lexical statistics

2.1 The Model

What we observed in the previous section, then, is the need to augment our PCFG grammar with more detailed lexical information. To do this we devised the system shown in Figure 2. Starting with a body of text (the corpus), we collect statistics on how individual words are used to create the "statistical database." These statistics are then used by a parser to parse new sentences (the "test data") and the "parsing results" tell us how well our new parser works. (For the moment we ignore the lower portion of the diagram.)

Now let us consider in slightly greater detail the kinds of statistics collected in the database. Figure 3 gives a parse for the sentence "Sheep dogs pulled the sleds over rocks." Each constituent, such as the np for "Sheep dogs," is marked with both its part of speech (in this case np) and its *head*. The head of a constituent is the "central" word of the phrase, "dogs" in the case at hand. More generally, the head of a noun phrase is typically the main noun in the phrase, the head of a verb phrase is the main verb, the head of a prepositional phrase is the preposition, etc. So besides the np we just looked at, we see that the head of the verb phrase "pulled sleds over rocks" is the main verb "pulled," while the head of the prepositional phrase "over rocks" is the preposition "over."

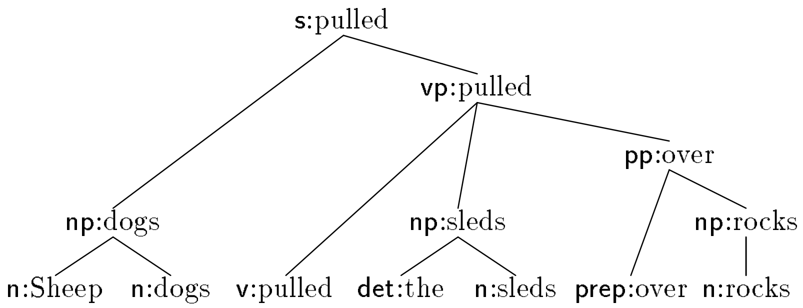

Figure 3: Phrase marker for "Sheep dogs pulled the sleds over rocks."

In our comments about the parse in Figure 1 we noted that we could have done better had we kept track of the kinds of rule used to expand individual prepositions; to use our new terminology, we want to keep track of the rule used for each possible head of the constituent. Again, when we noted that "bankers" is an unlikely modifier of "Reagan," we were commenting that "bankers" is a poor subconstituent of a constituent headed by "Reagan." Thus our statistics keep track of such things. Figure 4 shows some statistics gathered from the example in Figure 3. We see there that the preposition "over" is the head of a pp that uses the rule pp $\rightarrow$ prep np and that the noun "sleds" is a subconstituent of a phrase headed by "pulled." The "count" in Figure 4 indicates how often we have seen the relationship in question. In our example each relation was seen exactly once. When processing a large corpus the counts are summed over all of the sentences.

Actually, things are more complicated than Figure 4 lets on. First, the data collected by this system is more detailed than shown here. For instance, distinctions between subjects and objects of verbs, ignored in our example, are captured in the real data. Also, we have assumed here that there is only one parse for this sentence, while in reality sentences have multiple parses. However, techniques for handling this obstacle are well known (see, e.g., [7]) and we do not go into them here.

2.2 The Experiment

The system was run using data collected by parsing 36 million words of *Wall Street Journal* (WSJ) text provided by the ACL Linguistics Data Collection Initiative (ACL-LDCI). (For more on this data, see [11].) The grammar used throughout the experiment is a context-free grammar developed from

Head	Rule/Subhead	Count
pulled		
	s → np vp	1
	vp → v np pp	1
	dogs	1
	sleds	1
	over	1
over		
	pp → prep np	1
	rocks	1

Figure 4: Portion of the statistics for "Sheep dogs pulled the sleds over rocks."

work on context-free grammar induction described in [4, 5].

To test the model we used sentences from parsed ACL-LDCI WSJ text, none of which is used in training. We restricted consideration to sentences that use the 5400 most common words from this corpus, which were those that appeared in the corpus 500 or more times.

We take as the principal measurement of parsing success bracket-crossing accuracy (henceforth called simply *accuracy*): the percentage of bracketing found in the model's most probable parse that do not cross the bracketing in the LDCI parses. We like this measure because it is relatively insensitive to disagreements about the structures a grammar should assign. However, we are cognizant of its deficiencies; when used alone it can be made artificially high by certain tricks. Thus to keep ourselves honest we also report *recall*: the percentage of bracketing found in the LDCI parse that were also in the model's most probable parse. However, we judge the significance of the results solely in terms of the accuracy measure.

We compare these measures for three models:

1. The full model as described earlier ("complete")
2. The model using only the PCFG probabilities ("PCFG")
3. A model assuming uniform right-branching parses ("right branching").

In a right-branching model each word is assumed to start a new constituent that ends at the end of the sentence, e.g., (The (girl (went (to (the park))))),

Table 1: Syntactic disambiguation results

Probability Model	Percent Accuracy	Error Reduction	Percent Recall
Complete	91.1	41.1	80.8
PCFG	84.9		76.1
Right Branching	57.2		67.2

which has an accuracy of .75. English is close to being completely right branching, so "right branching" is generally considered a good "dumb" benchmark.

The major parsing results are shown in Table 1. The picture painted by these figures is reasonably clear. While assuming pure right branching only gives an accuracy of only 57.2%, we get 84.9% accuracy for a PCFG using none of the word statistics, and 91.1% accuracy for the model using all of our statistical information. This latter figure represents a 41.1% reduction in the number of bracket-crossing errors when compared to the pure PCFG. This is consistent with other work using statistics on lexical information such as [1, 9].

3 Verb Case Frames

We now turn to the question of what the system described above actually learned. We took as our initial hypothesis that it had, in fact, learned at least some of the lexical information commonly of interest in linguistic discussions. To test this hypothesis we applied "data-mining" techniques to the statistical database that drove the parsing system, as outlined in Figure 2. That is, if the system had in fact learned this information, it ought to be reasonably easy to extract it from the statistical data already extant. Or, to put it another way, we ought to be able to reuse our system as an automatic collector of lexical information.

In our first experiment we attempted to collect verb case-frame information automatically from our data. Verb case frames are defined as the types of syntactic arguments a verb can take. These syntactic arguments can be in the form of objects, prepositional phrases, infinitives, etc. For example, the verb *abandon* is followed either by a noun phrase or by a noun phrase

Table 2: The sixteen verb case frames employed in our system

Number	Case Frame	Explanation
1	iv	Intransitive verb
2	np	Transitive verb
3	dtv	Ditransitive verb
4	that	That complement
5	np_that	Object followed by that complement
6	wh	Wh-clause complement
7	np_wh	Object followed by wh-clause
8	inf	Infinitive complement
9	np_inf	Object followed by infinitive
10	ing	-ing participle complement
11	np_ing	Object followed by -ing participle
12	adj	Adjective used as a complement
13	np_adj	Object followed by adj complement
14	adj_inf	Adj complement followed by inf
15	p	Prepositional phrase
16	np_p	Object followed by a preposition

and a prepositional phrase.

Yesterday they abandoned the project. (Case frame: np)
He abandoned himself to despair. (Case frame: np_p)

There have already been a few attempts at learning verb case frames from corpora. Brent [2, 3] has tried to extract this information from both untagged and tagged corpora. Similarly, Manning [10] generated case-frame lists for verbs from untagged corpora. To make our results compatible, we tried to follow Manning whenever possible. In particular, since different dictionaries supply different kinds of case frames for verbs, we used the verb case frames Manning used as specified in Table 2.

3.1 Identification of Case Frames

The system first identifies the verbs, next gathers the statistics on various case frames, and finally identifies which case frames for a verb appear to be statistically significant.

Because we parse the corpus, we can identify a word as a verb if it is the head of a verb rule (i.e. a rule that expands a verb phrase vp.) This is a straightforward and simple procedure when compared with Brent's technique for verb detection by which anything that occurs both with and without the suffix -ing in the text is considered a potential verb, and if a potential verb is not preceded by a determiner or a preposition other than *to*, it is taken as a verb.

The grammar used to parse the WSJ corpus contains 1209 rules for expanding verb phrases. These rules are mapped into our verb case frames. The verb rules that contain adverbs are treated as if they contained no adverbs, e.g. the rule vp $\rightarrow$ v adv pp is mapped to the case frame in which a verb takes a prepositional phrase, i.e. p. If a rule contains punctuation marks or connecting words such as "and", only the part of the rule up to the punctuation mark or connecting word is taken into consideration, e.g. the rule vp $\rightarrow$ v np and vp is classified as a transitive verb rule. Because our case frames do not have anything after a preposition, the verb rules having nonterminals after a preposition are treated as if the preposition were the last nonterminal of the rule. Pronouns that appear in the rules are considered nouns, e.g. the rule vp $\rightarrow$ v pron np is mapped to the rule vp $\rightarrow$ v np np, i.e. ditransitive. Finally, some rules cannot be associated with our case frames, because of either the complexity of the rules or errors in the grammar. The probability mass of these rules, however, is less than 2% of all the rules.

We then go through our statistical data looking at each rule associated with a particular verb, as in Figure 4. If we have been able to associate that rule with a particular case frame, the count for that rule is added to the count for the appropriate verb/case-frame pair. When all of the statistics for a particular verb have been processed, the data is filtered to determine which case frames appear to be statistically significant.

3.2 Filtering

Some of the case frames identified by our program are the actual case frames of the verb, while some are wrong ones due to a mistake in the tagger or parser or to other causes, such as prepositional phrases that are not actual arguments of the verb. So the raw results have to be filtered and the actual case frame assignments distinguished from the wrong ones.

The filtering method used here is the one proposed by Brent [2]. This method assumes that B_s is the estimated upper bound on the probability that the program assigns a wrong case frame to a verb token. Assume that a verb occurs m times in a corpus, and n of those times is classified as having a certain case frame. The B_s values are used to calculate the probability that all these n assignments are wrong. This probability is bounded by

$$\sum_{i=n}^{m} \binom{m}{i} B_s^i (1 - B_s)^{m-i} \tag{1}$$

In our case, n indicates the counts for each case frame, m is the total number of occurrences of the verb assigned to any case frame, and $\binom{m}{i}$ is the i-combination of m elements.

If the probability that all n assignments are false is low, then the probability that at least one of them is correct is high. So if the above sum is less than some confidence level (in our system $C = 0.02$) then we assume that the case-frame assignment is correct.

Each case frame has its own B_s value, since the probability that a case-frame assignment is wrong changes from frame to frame. For example, the tagger and parser we used are more likely to make mistakes that generate extra p frames than any other frame. All the B_s values have been set empirically.

3.3 Evaluation

The *Oxford Advanced Learner's Dictionary* (OALD) [8] is used for testing the results of the program. The case frames learned by the program and the case frames in OALD do not have a direct correspondence, so OALD's 51 case frames are mapped into our 16 case frames. Then the machine-readable version of OALD is used to extract all the verbs and their case frames automatically. That version of OALD has separate entries for the different forms of verbs (e.g., abandon, abandoned, abandoning, and abandons), so each verb form that appears in our data can be directly compared to the dictionary.

The comparison program outputs, for each verb in our data, the correct case frames that our system generated, the extra ones (those not in the OALD but generated by our program) and the missing ones (those in OALD but not generated by our program).

Our system can be evaluated using two kinds of measurements: *precision*, the ratio of the correct case frames generated by our system to all the case frames generated by our system, and *recall*, the ratio of the correct case

Table 3: Comparison with previous verb-case-frame work

	precision (%)	recall (%)
combined verb forms	92	52
original program	87	58
Manning's system	90	43

frames generated by our program to all the case frames the dictionary supplies. Among the most common verbs of the WSJ corpus, 30 were chosen randomly to be used in the evaluation. These verbs are different from those used by Manning, because the contexts of the corpora are different.

We achieved 87% precision and 58% recall. As mentioned previously, different forms of a verb are evaluated separately. Then the information about them is combined. To do this, we first grouped the different forms of every verb. Next, for each group the aggregate rule counts were calculated. As seen in Table 3, this had the effect of increasing precision and decreasing recall. What happened, intuitively, is that combining verb forms increased the number of times we saw any particular verb so the number of times we needed to see a case frame went up as well. It seems that many case frames were only observed in one (or a few) forms of the verb and these were deleted by the binomial filter.

Table 3 compares the results of our system to those generated by Manning. It can be seen that both systems achieve almost the same precision, and our system has slightly better recall. Brent's systems, on the other hand, was able to learn only six case frames (np, dtv, that, np_that, inf, np_inf). Table 4 shows the case frames our system generated for the group of verbs used in the evaluation: the table lists the number of correct case frames generated by our program, the number of wrong case frames, and the number of case frames of the verb. Additionally, the final column shows the incorrect case frames generated by our program.

4 Prepositions

The grammatical rules of English require particular prepositions after particular nouns and adjectives, some of which are obligatory and some optional. Examples of these kinds of noun-preposition and adjective-preposition pairs

Table 4: Comparison of verb case frame results with OALD

Verb	# Right	# Wrong	Out of	Wrong case frames
abandon	2	0	2	
admit	3	1	7	iv
agree	2	0	5	
aim	1	0	5	
announce	2	0	3	
ask	4	0	9	
calculate	3	0	5	
decide	4	0	8	
delay	3	1	4	np_p
determine	3	0	7	
employ	1	1	2	p
engage	2	0	7	
fear	3	0	6	
gain	2	1	5	np_inf
hear	5	0	9	
join	3	0	4	
learn	3	0	7	
look	2	0	7	
make	4	0	9	
measure	3	1	4	wh
pick	2	0	3	
plunge	2	0	4	
prepare	3	0	4	
project	3	0	4	
provide	2	0	4	
retire	3	0	3	
rise	1	2	3	np, np_inf
study	3	0	6	
watch	5	0	8	
withdraw	3	0	4	
TOTAL	82	7	158	

are: “frontier between”, “head of”, “interested in”, “glad about”, etc. We tried to determine both obligatory and optional prepositions, though not the distinction between obligatory and optional.

The program that finds obligatory or optional prepositions attached to nouns starts by identifying the prepositions and nouns. We identify nouns as words that head noun rules, and similarly for prepositions. As mentioned previously, we already have the information about which preposition is attached to which word in the corpus. (Figure 4 shows the preposition “over” as attached to the verb “pulled.” The principle is the same for prepositions attached to nouns and adjectives.) Using this information, we find the prepositions attached to each noun and their counts. Next this raw data is filtered to get rid of the rare prepositions and errors that could have occurred during the tagging and parsing process.

The filter used at this stage is the same as the one used in filtering verb case frames, i.e., the binomial filter. However, this time all prepositions are assigned the same B_s value, 0.01. This value is found experimentally. In the binomial filter m is the number of total occurrences of a particular noun and n is the number of times it appears with a specific preposition. Those that pass the filter are considered correct noun-preposition pairs.

The same procedure is followed to determine the prepositions for adjectives.

4.1 Evaluation

To evaluate the results of the programs, a group of nouns and adjectives was chosen randomly among the most common nouns and adjectives that take prepositions. This time, however, the evaluation was done by hand, since no on-line source on this type of prepositional information could be found. We used the OALD (which does not have the information on prepositions on-line) to evaluate our results. A problem arose in the evaluation because the dictionaries were not concise and complete in listing the prepositions that can be used with nouns and adjectives; instead the dictionary lists prepositions that can be attached to the nouns but in example sentences new prepositions are introduced. Also, some prepositions that we believe correct do not appear in the dictionary: e.g. price *of*, offer *of*, house *of*. Therefore we make two different evaluations: one takes into account only those prepositions that appear in the dictionary, and another (called *self evaluation*) also allows those prepositions we believe correct. Table 5 displays the precision and recall for both nouns and adjectives.

Table 5: Precision and recall according to the dictionary and self evaluation

	precision (%)	recall (%)
self evaluation (nouns)	87	55
dictionary evaluation (nouns)	70	45
self evaluation (adjs)	80	71
dictionary evaluation (adjs)	65	67

We believe that a high precision for both nouns and adjectives has been achieved. Tables 6 and 7 display our results on the small set of nouns and adjectives used in the evaluation. In these tables we show for each noun/adjectve the number of correct (as judged by the self-evaluation method) prepositions generated by our program, the number and types of wrong prepositions generated by our program, and the toal number of prepositions for the word. The prepositions that do not appear in the dictionary but were classified as correct in the self-evaluation results are listed in the final column.

5 Conclusion

We described a system that learned to better determine the correct parse of a sentence by parsing a large amount of text and gathering statistics on the properties of particular lexical items. The use of these statistics resulted in a 41% decrease in the number of parsing errors.

We then looked more closely at exactly what the system had learned by collecting these statistics and found that it had, at least in part, learned many of the lexical relations that linguists consider important. In particular, we looked at the statistical database produced by the original system and extracted from it verb case-frame information. The resulting data is comparable to that provided by special-purpose systems for collecting this data. We also extracted data on the prepositions used by various nouns and adjectives. In this case we did not have a prior hand-written system against which we could compare, but the results seem convincing nevertheless.

Table 6: Comparison of noun-preposition couples with OALD

Nouns	# Right	# Wrong	Out of	Wrong prepositions	Assumed correct
account	1	1	3	for	
acquisition	2	0	2		
agreement	2	0	5		on
amount	1	0	1		
bank	2	0	2		in
bid	1	0	2		
board	1	0	1		
business	2	0	4		in
chairman	1	0	1		
companies	3	1	3	that	in
control	1	0	3		
court	2	0	2		in
decline	2	0	2		
demand	2	0	3		
exchange	2	1	3	in	
group	1	0	1		
growth	2	0	2		
head	1	0	5		
index	1	0	1		
line	3	0	6		
lot	1	0	3		
market	2	0	2		
meeting	3	0	4		in, with
member	1	0	1		
number	1	0	2		
offer	2	0	2		
operations	2	0	3		in
part	1	0	2		
president	1	0	1		
price	3	0	4		in
quarter	1	1	1	from	
sale	2	0	3		in
share	2	1	3	from	
stake	1	0	2		
unit	1	1	1	in	
value	1	1	15 3	at	
TOTAL	47	7	86		

Table 7: Comparison of adjective-preposition couples with OALD

Adjective	# Right	# Wrong	Out of	Wrong prepositions	Assumed correct
adequate	1	0	2		
afraid	1	0	2		
available	2	0	3		
aware	1	0	2		
capable	1	0	1		
cautious	1	1	2	in	
compatible	1	0	1		
consistent	1	0	1		
different	3	0	4		in
difficult	1	0	1		
due	1	1	3	out	
eager	1	1	1	in	
enthusiastic	1	0	2		
familiar	1	0	2		
fearful	2	0	2		
guilty	1	1	2	in	
highest	3	1	3	for	of, since, in
impossible	1	0	2		
larger	2	0	2		of, than
responsible	1	0	1		
same	1	2	1	for,in	
skeptical	2	0	2		
suitable	1	0	2		
typical	1	1	1	in	
TOTAL	32	8	45		

References

[1] Bod, A. Rens, Using an annotated language corpus as a virtual stochastic grammar. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Menlo Park: AAAI Press/MIT Press (1993) 778-783

[2] Brent, Michael R., Automatic acquisition of subcategorization frames from untagged text. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics* (1991) 209–214

[3] Brent, Michael R. and Berwick, R. C., Automatic acquisition of subcategorization frames from tagged text. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop* (1991) 342–345

[4] Carroll, Glenn and Charniak, Eugene, Two experiments on learning probabilistic dependency grammars from corpora. In *Workshop Notes, Statistically-Based NLP Techniques*, AAAI (1992) 1-13

[5] Charniak, Eugene and Carroll, Glenn, Context-sensitive statistics for improved grammatical language models. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Menlo Park: AAAI Press/MIT Press (1994) 728-733

[6] Charniak, Eugene, *Parsing with context-free grammars and word statistics.* Technical Report CS-95-28, Department of Computer Science, Brown University (1995)

[7] Charniak, Eugene, *Statistical Language Learning.* Cambridge: MIT Press (1993)

[8] Hornby, A. S., *Oxford Advanced Learner's Dictionary of Current English.* Oxford: Oxford University Press. 3rd ed. (1985)

[9] Magerman, David M., Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* (1995) 276-283

[10] Manning, Christopher D. Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics* (1993) 235-242

[11] Marcus, Mitchell P., Santorini, Beatrice, and Marcinkiewicz, Mary Ann, Building a large annotated corpus of English: the Penn treebank. In *Computational Linguistics* **19** (1993) 313-330