Learning Probabilistic Dependency Grammars from Labelled Text

Glenn Carroll and Eugene Charniak*
Department of Computer Science
Brown University
Providence RI 02912

Abstract

We present the results of experimenting with schemes for learning probabilistic dependency grammars[1] for English from corpora labelled with part-of-speech information. We intend our system to produce wide-coverage grammars which have some resemblance to the standard[2] context-free grammars of English which grammarians and linguists commonly exhibit as examples.

Introduction

The aim of our current research is to produce a system capable of learning probabilistic grammars for English from labelled text. Our approach is pragmatic, rather than theoretically pure; simply put, we are interested in delivering results without clinging to some particular model or method. As far as English syntax goes, it is our position that any construct which appears frequently in our training text is indeed a "good" construct, and deserves to be parsed by our grammar, regardless of what a grammarian or linguist might say about it. We do not expect our system to produce grammars which will necessarily parse every sentence given to them. After all, grammatical errors do occur, and we would not want to commit ourselves to parse sentences that might well be little more than gibberish. We will discuss below which sentences will be labelled as errors or exceptions by our system.

Our system has a dynamic orientation to it, although it is not, strictly speaking an online system (yet). We take this approach for two reasons: first, because it appears more practical for learning grammars–we can first learn simple grammars and then extend them to more complex ones. Second, we would like to be able to track grammars that might be moving targets. Human speech and writing are continuously adapting and changing. We would like our system to continuously and efficiently re-tune its grammar to track a target text whose style might be slowly changing over time.

We assume the reader is familiar with the inside/outside[3] algorithm (**Baker [1982]**), and dependency grammars. Briefly put, the inside/outside algorithm trains a probabilistic grammar so as to minimize the cross entropy of the grammar given the training corpus. For us, it provides the foundation for unsupervised learning of probabilistic grammars. For more information, see (**Baker [1982]**), or what is perhaps a better description in (**Jelinek, Lafferty & Mercer [1991]**).

Dependency grammars are a highly restricted form of context-free grammar, in which only rules of the form $\overline{X} \rightarrow \alpha X \beta$ are allowed. Here X is a non-terminal, α and β are possibly empty strings of non-terminals. For each terminal in the language, there is exactly one corresponding non-terminal, which by convention is given the name of the terminal with an overlying bar.

We are not the only researchers to notice that dependency grammars and the inside/outside algorithm combine very nicely to produce a system which shows some promise of learning probabilistic grammars for natural languages. Where appropriate below, we will compare and contrast our system with those other systems which have been reported.

The rest of this paper is laid out as follows: first we describe the basic algorithm. While this is not terribly subtle, we had to make choices which are worth clarifying. Second, we describe the constraints we have applied to the problem in order to make it learnable. Third, we discuss enhancements to the learning system which have helped our performance on artificial grammars, and show some promise of helping us on our task of learning English grammar. Finally, we describe the visible next steps in our research program.

*We would like to thank Mark Johnson for many useful discussions. This research was supported in part by NSF contract IRI-8911122 and ONR contract N0014-91-J-1202.

[1] A.k.a. X-bar grammars or headed grammars.

[2] i.e. narrow-coverage.

[3] A.k.a. the Baum-Welsh algorithm (**Baum [1972]**). Our particular version of this algorithm is extended to allow context-free grammars which need not be in CNF.

```
istream trainingStream;
Grammar grammar;

while( !eof( trainingStream )
{
        Sentence sentence( trainingStream );

        if( !grammar.parse( sentence ) )
            grammar.constructRules( sentence );

        retrain( grammar, trainingStream );
}
```

Figure 1: The basic algorithm in C++ pseudo-code.

Algorithm basics

We give C++ code for our basic algorithm in figure 1. Before any of this is executed, we sort the input text, so that sentences are presented to the learning system in order of increasing sentence length. We will discuss why below. In step 1, we make our grammar, which is initially empty; we will be adding rules to this as necessary as we move through the corpus, and we will be retraining it periodically as well. In the loop, we read in a sentence from the training corpus, and try to parse it. If the parse fails, we construct new rules for the sentence. Since we are using a dependency grammar, the number of rules which could possibly be used to parse this sentence is finite. Our procedure for constructing rules is unsophisticated, constructing *all* possible rules which could be used to parse the sentence, and producing a grammar which is the union of this set of rules and the current set of rules. As the last step in the loop, we retrain the grammar. Our retraining function uses the inside/outside algorithm iteratively to refine the probabilities of the rules, minimizing the cross-entropy of the grammar relative to the text seen so far. Since the inside/outside algorithm needs to review the entire corpus seen so far, our algorithm is not so purely dynamic as this code sketch makes it appear. We do, indeed, back up and re-read all the sentences whenever we actually have to retrain. By caching parse information, we can actually avoid retraining for many iterations through this loop, but that is largely a performance issue.

The primary difference between our algorithm and others is that we start with an empty grammar, construct rules as necessary and retrain iteratively. Taking a batch approach, which is so far the universal choice aside from our system, we would be compelled to construct all candidate rules for every sentence in the entire corpus, and then train this vast set of rules over the entire corpus. This is impractical. It is also, we believe, unnecessary, as this learning problem lends itself to an incremental approach. Learning small, simple grammars, namely those that cover the shorter sentences in the corpus is a much simpler task, and the grammar acquired is a good headstart on the more complex grammar required for longer sentences.

Pereira and Schabes in (**Pereira & Schabes [1992]**) start with all possible rules initialized to some very low probability. In order to make the system tractable, they use much stronger constraints to reduce the size of the set of all possible rules, and rely on the much stronger information provided by bracketed text.

Briscoe and Waegner, (**Briscoe & Waegner [1992]**) come closer to our scheme, but they also have a batch-oriented system. In effect, they begin with a set of high-probability rules which have been selected by hand, and the rest of the set of all possible rules initialized to some fixed, low probability. Again, in order to render this trainable, they must reduce the size of possible rule set, and they do this by using much stronger constraints than we do. It is unclear whether these constraints leave all reasonable candidates for rules of English within the possible set of rules. If so, then we could improve our performance by borrowing these constraints; if not, then it is no longer possible for the system to learn the best English grammar.

There is another, more subtle, problem with both of the previous approaches, and any batch approach. By resorting to strictly batch operation, they exacerbate what amounts to an unsolved problem for the inside/outside algorithm: how to initialize the probabilities. A moment's reflection shows that the very best initial probabilities are actually the very best final probabilities, and clearly, if one knew these, one wouldn't have to run the inside/outside algorithm at all. This would not be a problem if the inside/outside algorithm were insensitive to initial probabilities, but unfortunately, this is not the case. Viewed as an algorithm for searching through a space whose coordinates are a vector of probabilities, the algorithm is quite sensitive to initial conditions, and it is prone to getting stuck in local minima. For a problem on the scale of an English grammar, there are a great many local minima in which to get stuck.

In our earliest experiments, we tried using near-uniform probabilities and a more batch-oriented approach[4] and so we became interested in the question of just how many local minima there were to get stuck in. We ran an experiment, in which we took a set of rules our system had learned, and which covered an artificial, pseudo-English corpus. We randomly initialized the probabilities of these rules, and then we retrained, repeating the inside/outside algorithm until it converged. We were interested in how many times this procedure would produce duplicate grammars, namely ones stuck in the same local minima. In 300 runs, we did

[4]To be precise, we processed the corpus on a section-by-section basis, with each section consisting of the set of all sentences of a given length.

not land in the same local minimum once; the grammars differed in their rankings of rules, they differed significantly in the probability of individual rules, and they commonly differed on the measure of net cross entropy with respect to the corpus. In no measure could they be said to be very nearly the same. Given the vastly larger set of rules which will be necessary for English, and the still larger set necessary when a batch approach is taken, the number of local minima would appear to be astronomical. Starting with uniform probabilities is, so far as we know, equivalent to drawing one of these local minima out of a hat. Chances of selecting the global minimum, or something closely resembling it, are small.

Like the approaches above, we cannot solve the problem of how to initialize the rule probabilities. We have, however, both experimental evidence that our dynamic approach works better than a batch one (which is why we abandoned our own batch scheme), and an argument for why our dynamic approach ameliorates the problem of initialization. Since we sort the input corpus, the initial sample presented to our algorithm contains only very short sentences, which can only be generated by very short rules. Our initial search, then, is among a small set of very short rules. This search space is many orders of magnitude smaller in both size and dimensionality than any of the search spaces discussed above. That being the case, we have a much better chance of landing on the correct local minimum, or a good neighbor, by chance. Like everyone else, we are reduced to initializing new rules using some fixed, low probability[5]. However, we do not re-initialize old rules. In effect, our algorithm extends solutions in the reduced space (where sentences, and hence, rules are shorter) to solutions in the expanded space (where sentences, and some rules, are longer).

Constraining the Problem

Relatively few learning problems are tractable in their rawest, most unconstrained form. Our first experiment in learning probabilistic dependency grammars, reported in (**Carroll & Charniak [1992]**), failed to produce a reasonable grammar precisely because it used no explicit constraints. We had been optimistic that simply using dependency grammars, which provide powerful constraints as compared with CFG's, would suffice. We then added the explicit constraints described below, and, as we reported in the same paper, succeeded in learning the target artificial grammar.

[5] Actually, we initialize the count of a new rule to one, indicating that it is tentatively used once in the new sentence, and we compute its probability from that. By initializing counts and not probabilities, we automatically weight old information appropriately. For example, let us pick some non-terminal, say $\overline{X}$. If the corresponding terminal X appears seldom in the corpus, then the sum of the counts of all rules which have $\overline{X}$ on the left-hand side will be low, and therefore a new rule for such a terminal will have a relatively high probability.

Dependency Grammars

As mentioned above, we regard our use of dependency grammars as a problem constraint because they are a severely restricted subset of CFG's. Their restrictions provide several critical properties for our endeavor:

- first, they allow us to write a procedure for constructing all possible rules which could be used to parse a sentence. If the set of non-terminals were unbounded, this would be impossible, as one could always invent a new non-terminal and add a rule with the new non-terminal on the left and the same right-hand side as some other rule used to parse this sentence.
- Next, they guarantee a bounded depth for the parse tree. If the number of non-terminals was bounded, but epsilon-transitions were allowed, then we could construct all the rules used for a sentence, but parse depth could be infinite, which would make training with the inside/outside algorithm impossible.
- In addition to the above, dependency grammars give us a bonus: given a valid right-hand side of a rule, we immediately know which non-terminal must be on the left-hand side. Conceivably we could live with out this, but the additional expressiveness doesn't look worth the additional problem complexity.

Explicit Rule Constraints

We have also added explicit constraints on the form of rules are allowed. In essence, we want to bias the learner, using what we already know about which English parts of speech (our non-terminals) should be allowed to head phrases. For example, determiners should not head noun phrases. For each non-terminal, we simply write down a list of what non-terminals are allowed on the right-hand side of its rules. The rule construction procedure then tests candidate rules to ensure that no forbidden part of speech appears on the right hand side of a rule.

These constraints are trivial to come up with, they capture human intuition about the structure of English grammar that would be hard to learn otherwise, and they solve a major problem with unconstrained learning of dependency grammars. The problem is that the basic algorithm described above will learn what we have dubbed a "memorizing grammar". Such a grammar has one rule for each distinct sentence, plus rules of the form $\overline{X} -> X$. For each sentence, the probability of the corresponding rule is just the number of times that sentence appears in the corpus, divided by the number of sentences in the corpus. It is provable that such a grammar is optimal, in the sense that it will have a net cross entropy as good as, or better than, that of the grammar used to generate the corpus. Clearly such a grammar reveals no useful structural information about the sentences in the corpus, and is utterly useless as far

as parsing is concerned. Therefore some constraint(s) must be used to rule it out as a solution. The explicit constraints we have described have so far proved quite efficient for this in artificial grammars. Moreover, we are interested in producing human-oriented grammars, and these constraints provide a strong push in that direction without, we feel, over-constraining the possible solutions.

Lastly, and with perhaps least justification, our system has two numerical parameters that are effectively two further constraints. Both of these are slated for removal, so we will not discuss them in detail here. The first is a threshold for rule probability; following training, we discard all rules which fall below this threshold. We originally thought this was necessary to keep the number of rules down, and therefore keep the time for retraining within bearable limits; upon re-evaluating, we believe that our system can run quite successfully without it. The second parameter is a limit on rule length, which was a first attempt at preventing the formation of memorizing grammars. It turns out that memorizing grammars degrade very gracefully, so this was not effective.

Enhancing Learning

If the above were all that was necessary to learn probabilistic dependency grammars, then we would congratulate ourselves roundly for having discovered such an elegant and straightforward solution. Unfortunately, there are (at least) two remaining problems. First, as discussed above, our dependency grammars tend to acquire a large number of overly specific rules, rather than developing a small set of general purpose ones. The constraints have greatly mitigated this problem, but they are not as complete a solution as we would like. Second, it appears that once our system has adopted a grammar that is not near optimal, it is unlikely to discover an optimal or near optimal one later on. Due to sampling error and the skew induced by our ordering, it appears impossible to prevent this from happening in all cases. Looked at another way, it appears only too likely that there will be corpora which present misleading information at some stage of the learning process; while this is not a problem for batch-mode learning algorithms, it will arise for any system committed to online learning. Based on the notion that anything that can go wrong, will go wrong, we have to plan on coping with this. Adding constraints does not seem like the right approach for either of these problems; rather, we concentrate on enhancing the learning process, both making better use of the information we have available, and pushing the learner in the desired direction of shorter, more general grammars. We begin with the second problem first, as it appears to be more critical.

Probabilistic Pseudo-Negative Examples

It is a commonplace of machine learning that learning with negative examples is much easier than learning without them. The big problem with negative examples is finding a source of them, especially an automatic source, as people are slow, unreliable, and relatively expensive. [6] In our case, we'd have to take a grammar from the system and return a sentence which the grammar recognized but was ill-formed or improbably likely. Recognizing ill-formed sentences would require the right grammar, which would obviate our learning another one. Hence, we must some how leverage off our probabilities.

Simply put, we are interested in finding sentences whose computed probability differs significantly from their observed frequency; these will be our probabilistic pseudo-negative examples. We restrict ourselves to sentences whose computed probability is too high, rather than too low, as it is easier to make use of this information. Such sentences indicate that the rules used to parse them have excessively high probabilities. In the special case in which we find a sentence whose observed frequency is zero, then we can be fairly certain (given a reasonable sample size) that some rule or rules used to parse this sentence should be eliminated.

That being said, the technique for generating and using probabilistic pseudo-negative examples is fairly straightforward. See figure 2 for a C++ code fragment. We use our grammar to generate sentences, and compute the probability of each sentence. (We generate sentences in order of decreasing probability, and cut off generation when the probability is sufficiently low that the sentences cannot be suspicious.) Once that is done, we can compute whether the observed frequency of the sentence is plausible using Chebyshev's inequality. If the sentence is deemed implausibly likely, we note all of its rules. We repeat this for all of the generated sentences, gathering up all the suspicious rules. We then iterate over each suspicious rule, eliminating it from the grammar and retraining. We allow the retraining to add rules as necessary, with the exception of the rule currently under suspicion. This allows the grammar to "patch" itself to cover the corpus, and explore an alternative solution. Retraining produces the resulting cross entropy, and we continue learning with whichever grammar shows the best cross entropy.

We will walk through a contrived example to make our use of Chebyshev's more perspicuous. Suppose the sentence

```
det noun verb .
```

[6] There is also the issue of finding a human with the necessary expertise. The thought that we might be the only ones who understood what our program required, and hence *we* would be the ones sitting in front of the computer and supplying negative examples, was sufficiently chilling that we went back to looking for an automatic source.

```
///////////////
//
// Search for a better grammar, using
// probabilistic pseudo-negative examples
//
///////////////
gramSearch( Grammar& startingGrammar,
   double& startingCrossEntropy )
{
   Grammar bestGrammar( startingGrammar );
   double bestCrossEntropy;
   bestCrossEntropy = startingCrossEntropy;
   SentenceSet allSentences;
   RuleSet     suspectRules;

   startingGrammar.generate( allSentences );
   while( !  endOfSet( allSentences ) )
   {
      Sentence sentence( allSentences.next() );
      if( implausible( sentence ) )
         suspectRules.insert( sentence.parse() );
   }
   while( !  endOfSet( suspectRules ) )
   {
      Rule suspectRule( suspectRules.next() );
      Grammar testGrammar( startingGrammar );
      testGrammar.eliminate( suspectRule );
      double testCrossEntropy;
      testCrossEntropy = testGrammar.retrain();
      if( testCrossEntropy < bestCrossEntropy )
      {
         bestCrossEntropy = testCrossEntropy;
         bestGrammar = testGrammar;
      }
   }
   startingGrammar = bestGrammar;
   startingCrossEntropy = bestCrossEntropy;
}
```

Figure 2: Code for finding probabilistic pseudo-negative examples.

(read: "determiner noun verb ."–recall that we are using labels) is assigned probability .2 by the grammar and has been observed 5 times in a sample of 100 sentences. Chebyshev's states

$$Pr(||X - \mu||) \leq 1/c^2 \tag{1}$$

For the figures above, this means that the probability of seeing 5 or fewer sentences of this form is 0.0002 . We set the plausibility cut-off at .10; that is, the likelihood of seeing this few (or fewer) examples must be .10 or higher. Since 0.0002 is less than .10, we take all of the rules used to parse this sentence and put them into our bag of suspicious sentences.

Now, any time a numerical parameter shows up in a learning algorithm, it should be viewed with grave suspicion. Fortunately, the .10 above is not a magic number, but falls into the category of trade-off parameters, allowing one to control the behavior of the learning system by trading accuracy for time. The lower we set this number, the more we are insisting on proof of error, and hence, the more we will tolerate deviations from the desired grammar. We will, however, run faster, as we avoid the cost of retraining. Setting the number higher makes us more suspicious, but costs us retraining time.

It has been suggested that the statistical information we use is so weak, and will grow weaker as grammar complexity increases, that this can't possibly buy us anything. The information *is* weak, and it does indeed grow weaker as the grammar complexity increases, since in general the likelihood of the generated sentences steadily drops with increasing numbers of rules. [7] We have two replies to this, one being experimental evidence that we get good results from this procedure, and the other being an argument that weak evidence is sufficient for our needs.

For the experimental argument, see the artificial grammar in appendix A, which was unlearnable before we adopted this procedure. In preliminary tests with English text from the Brown Corpus, we have also managed to find improved grammars using this procedure. In our experience, this procedure is far more likely to fail when called early on, when sample size is very small, rather than later, when the grammar is more complex but the sample size allows for much more confidence. Whether this will extend directly to English text is not yet settled.

For the argument as to why we need only weak information, we first note that this procedure will never move us from a better grammar to a worse one, so at the very worst we will simply be wasting our time. We view our learning as a search for the right grammar. Our current expectation is that our learned grammar will never be far from the target grammar, so that our search need never look over long distances. The problem is that even local search is impossible, without any

[7] Note, however, that larger sample sizes increase the reliability of our measure.

guidance, as the dimensionality is in the hundreds and the number of local minima is huge–see our experiment above. The guidance need not be perfect, only probably approximately correct and sufficiently restrictive that the search takes a reasonable time. Our probabilistic, pseudo-negative examples, while providing weak information, fulfill both of those requirements.

Reducing Grammar Size

Our next and last technique is aimed directly at our algorithm's tendency to develop large, highly specific grammars rather than general ones. The underlying reason for this behavior would seem to be the same as that which explains why memorizing grammars appear: the resulting grammar is optimal or near-optimal in terms of its net cross entropy, and it is apparently the easiest one for our algorithm to find. Indeed, it is possible that, for some text, no non-memorizing grammar will have as low a cross entropy. What's worse, memorizing grammars are quite robust; even in the face of rule length limits which force some generalizing rules to be included, the bulk of the grammar will remain in the form of one sentence per rule, and the cross entropy will be very near to optimal.

Our approach is to reduce the number of rules in the grammar without forfeiting any coverage of the corpus. As mentioned above in the Constraints section, we discard rules below a certain probability threshold. As we advance through the corpus, we may eventually discard a rule which is necessary to parse one or more sentences. We use this condition to trigger an attempt to squeeze the grammar, as the presence of very low probability rules which are nonetheless required for parsing generally indicates that the grammar has gotten too large. Again our search scheme is extremely simple. We first form an ordered list of rules according to probability, and then step through the list, eliminating the rule and retraining. If we can parse all sentences without the eliminated rule, we leave it out and continue. No rule construction is allowed to take place. Since for each non-terminal the sum of the probabilities of the corresponding rules must be 1.0, eliminating a rule drives up the probabilities of the remaining rules. By successive elimination, we hope to force the problematic rule above threshold, thus, by the end of the procedure, the triggering condition should no longer be true.

This has never failed to work for our artificial grammars. Whenever the procedure was invoked, it did indeed produce a smaller grammar with all rules suitably above threshold. Unfortunately, it failed almost immediately in our preliminary run on real English. This probably merely means that our threshold, 0.001, is too high for English, which will have a much larger grammar than our artifical development cases and may have rules whose probability is lower than this threshold. Nonetheless, this came as a rude surprise, and led to some reconsideration of what we were doing. Part of the appeal of this technique was that there were no parameters which explicitly stated the trade-off between grammar complexity and cross entropy; nonetheless, there *is* a parameter, namely the threshold for pruning sentences. While it would indeed be nice if there were some parameterless way for trading complexity for cross entropy, it's not obvious that it is possible. In hindsight, what we have come up with appears to be a trade-off parameter for controlling how hard to try to reduce grammar complexity, while giving up some (unspecified) amount of cross entropy in return. We have not observed any significant loss in cross entropy as a result of this procedure–further testimony to the number of local minima with near-optimal cross entropy–but it may be that a more explicit control over the trade-off would be desirable.

Future Work

Artificial grammars and corpora have many handy properties that are helpful for experiments: test cases are easy to generate, learning performance can be accurately measured by comparing the learned and target grammars, and various parameters of the problem, for example, number of terminals, rule length, and so forth, can be varied and their impact studied. Clearly they are useful creatures to keep around the ivory tower of academia, but they are not the dragon we have set ourselves to slay. We are currently aware of two modifications to our algorithm which are necessary before it will produce good results on English text. The first modification is support for handling corpora which are too large for retraining, and we intend to handle this by using a sliding window of a thousand sentences or so. The second is support for handling exception (unparsable) sentences.

In our system, exception sentences can be recognized in two different ways. The first, and far easier way, occurs when it is impossible to construct rules to parse the sentence; this implies that the sentence violates one or more of our explicit constraints, so the sentence is simply marked an exception and ignored. The second case occurs when we run into trouble reducing the grammar. It may very well be impossible to produce a grammar in which all rules are above some threshold, and this will generally indicate that some sentence(s) which occurs only once require its own special grammar rule(s). Extremely low-frequency sentences which require special grammar rules are also flagged as exceptions, and ignored, once they are detected.

Conclusions

We believe that automatically generated grammars for English (or a large fraction thereof) will be created in the not so distant future. Moreover, the grammars learned will capture some of the standard human intuitions about what rules the grammar should have and

how, broadly speaking, it is structured. We have processed about 10% of the Brown Corpus in preliminary tests, and the results are encouraging. That is, these results resemble what we got for processing about 10% of our artificial corpora; some probabilities were wildly off, and there were extraneous rules, but there are also groups of rules which have already been arranged in their final probability ranking. This evidence, coupled with the results of our earlier experiments, gives us some reason to hope that the essential pieces for our system are in place. As is always the case in probabilistic learning, there are no guarantees.

Appendix A

Our last artificial grammar.

1.0	S	→	..
.4	..	→	noun_ verb_ .
.1	..	→	noun_ .
.1	..	→	aux_ noun_ verb_ .
.1	..	→	pron_ verb_ .
.1	..	→	verb_ .
.08	..	→	that_ aux_ adj_ .
.06	..	→	noun_ aux_ adj_ .
.06	..	→	prep_ noun_ verb_ .
.2	verb_	→	verb
.1	verb_	→	verb pron_
.1	verb_	→	verb prep_
.1	verb_	→	verb pron_ prep_
.1	verb_	→	verb noun_ noun_
.1	verb_	→	aux_ verb noun_ prep_
.07	verb_	→	verb noun_
.07	verb_	→	aux_ verb noun_
.07	verb_	→	verb noun_ prep_
.03	verb_	→	aux_ verb that_
.03	verb_	→	verb that_ prep_
.03	verb_	→	verb that_
.26	noun_	→	det_ noun
.17	noun_	→	det_ noun wh_
.16	noun_	→	det_ noun prep_
.09	noun_	→	noun
.07	noun_	→	det_ adj_ noun
.07	noun_	→	det_ noun verb_
.04	noun_	→	adj_ noun and_ noun_
.04	noun_	→	noun and_ noun_
.04	noun_	→	det_ noun and_ noun_
.03	noun_	→	noun prep_
.03	noun_	→	noun wh_
1.0	pron_	→	pron
.7	prep_	→	prep noun_
.2	prep_	→	prep pron_
.1	prep_	→	prep noun_ prep_
1.0	det_	→	det
.8	adj_	→	adj
.2	adj_	→	adj adj_
.5	wh_	→	wh verb_
.5	wh_	→	wh noun_ verb_
1.0	aux_	→	aux
1.0	and_	→	and
.6	that_	→	that noun_ verb_
.1	that_	→	that aux_ noun_ verb_
.1	that_	→	that noun_ aux_ adj_
.1	that_	→	that prep_ noun_ verb_
.1	that_	→	that pron_ verb_

The output from our learning system.

1.000000	S	→	..
0.336682	..	→	noun_ verb_ .
0.133954	..	→	noun_ .
0.118590	..	→	verb_ .
0.118407	..	→	pron_ verb_ .
0.075104	..	→	that_ aux_ adj_ .
0.073713	..	→	noun_ aux_ adj_ .
0.046103	..	→	verb_ prep_ .
0.033380	..	→	prep_ noun_ verb_ .
0.015535	..	→	aux_ verb_ prep_ .
0.014961	..	→	noun_ prep_ .
0.013914	..	→	noun_ verb_ prep_ .
0.006798	..	→	aux_ verb_ .
0.005376	..	→	pron_ verb_ prep_ .
0.004610	..	→	verb_ noun_ noun_ .
0.002873	..	→	verb_ that_ .
0.328038	verb_	→	verb
0.137521	verb_	→	verb pron_
0.080784	verb_	→	aux_ verb noun_
0.075778	verb_	→	verb noun_
0.074991	verb_	→	verb prep_
0.065449	verb_	→	verb pron_ prep_
0.054192	verb_	→	verb noun_ noun_
0.035408	verb_	→	aux_ verb noun_ prep_
0.035317	verb_	→	verb noun_ prep_
0.024908	verb_	→	aux_ noun_ verb
0.023719	verb_	→	noun_ aux_ verb noun_
0.014947	verb_	→	aux_ verb that_
0.014851	verb_	→	aux_ noun_ verb pron_
0.012828	verb_	→	verb that_ prep_
0.011748	verb_	→	verb that_

0.005989	verb_	→	aux_ noun_ verb noun_
0.003530	verb_	→	aux_ noun_ prep_ verb
0.366678	noun_	→	det_ noun
0.131325	noun_	→	noun
0.124311	noun_	→	det_ noun prep_
0.088172	noun_	→	det_ adj_ noun
0.087455	noun_	→	det_ noun wh_
0.057398	noun_	→	det_ noun verb_
0.035842	noun_	→	noun and_ noun_
0.035842	noun_	→	adj_ noun and_ noun_
0.028532	noun_	→	noun prep_
0.027957	noun_	→	det_ noun and_ noun_
0.016487	noun_	→	noun wh_
1.000000	pron_	→	pron
0.689024	prep_	→	prep noun_
0.310976	prep_	→	prep pron_
1.000000	det_	→	det
0.801105	adj_	→	adj
0.192479	adj_	→	adj adj_
0.006416	adj_	→	adj_ adj
0.740398	wh_	→	wh verb_
0.259602	wh_	→	wh noun_ verb_
1.000000	aux_	→	aux
1.000000	and_	→	and
0.613636	that_	→	that noun_ verb_
0.136364	that_	→	that aux_ noun_ verb_
0.113636	that_	→	that noun_ aux_ adj_
0.063957	that_	→	that pron_ verb_
0.049680	that_	→	that pron_ verb_ noun_
0.022727	that_	→	that prep_ noun_ verb_

References

Baker, J. [1982], "Trainable Grammars for Speech Recognition," in *Speech Communication Papers for the 97th Meeting of the Acoustic Society of America*, Wolf, D. K. andJ., ed., ASA, 547–550.

Baum, L. E. [1972], "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities* III, 1–8.

Briscoe, T. & Waegner, N. [1992], "Robust Stochastic Parsing Using the Inside-Outside Algorithm," in *Workshop notes for the Statistically-Based NLP Techniques Workshop*, 39–53.

Carroll, G. & Charniak, E. [1992], "Two Experiments on Learning Probabilistic Dependency Grammars from Corpora," Brown University Dept. of Computer Science, Brown University Technical Report No. CS-92-16.

Jelinek, F., Lafferty, J. D. & Mercer, R. L. [1991], "Basic methods of probabilistic context free grammars," Continuous Speech Recognition Group IBM T.J. Watson Research Center.

Pereira, F. & Schabes, Y. [1992], "Inside-outside Reestimation from Partially Bracketed Corpora," in *Proceedings of ACL '92*, 128–135.