Posterior Decoding for Generative Constituent-Context Grammar Induction

Chuong Do
Department of Computer Science
Stanford University
chuong.do@stanford.edu

Abstract

In this project, we study the problem of natural language grammar induction from a database of sentence part-of-speech (POS) tags. We then present an implementation of the EM-based generative constituent-context model by Klein and Manning. We also present two posterior decoding approaches to be used in conjunction with the constituent-context model and evaluate their performance against regular Viterbi parsing on a subset of the sentences from the Penn Treebank.

1 Introduction

Grammar induction refers to the general task of learning the structure of a natural language grammar given only examples of sentences generated by the grammar itself. In essence, by looking at a representative sample of a language, a program must be able to deduce the syntactic structure of the language. While the goal of completely understanding a language through unsupervised learning techniques over a restricted number of training examples is extremely difficult, methods that provide at least semi-decent analyses have uses as first steps in the production of larger grammars or treebanks. Sometimes, an automatically induced grammar may even be the only alternative available when the language being learned is unknown.

Conceptually and practically, the process of inducing a grammar may be split into two parts: (1) inferring the nonterminal classes that may be needed, and (2) deducing the grammar production rules that define the hierarchical structure of the language. The first task, more commonly known as part-of-speech induction, can be thought of as a distributional clustering problem, for which good results have previously been attained [5]. In this paper, we adopt the philosophy of Klein and Manning and regard the induction of part-of-speech tags as a largely solved problem [3].

The second task, which is often considered the more difficult of the two challenges, is essentially completion of the grammar by specification of the grammar production rules, given the assumption that part of speech tags for all words in the training examples are already known. In general, attempts at solving this latter task have had far less success, owing to the complicated hierarchical structure of natural languages [2]. Most systems that have had any success at the grammar induction task are forced to fix at least some aspect of the problem and then work on optimizing another. For instance, many methods based on probabilistic context-free grammars (PCFGs) begin by first fixing grammar production rules and then using the inside-outside algorithm to derive locally optimal probabilities for each production rule [1]. Other more ambitious approaches attempt to include some sort of structural search in tandem and rely on the minimum description length (MDL) heuristic as a guide for generating grammars [6].

Here, we explore a simplification of the grammar induction problem in which we consider only

the generation of the phrasal structure of a sentence based on only the part-of-speech labels for a sentence. This model does not attempt to provide labels for the identified phrases but rather leaves them anonymous. We discuss our implementation of the generative constituent-context model first described in [4] and discuss some attempts at improving the performance of the model on a test set derived from the Penn Treebank.

2 The generative constituent-context model

In the generative constituent-context model, let $S = \{s^1 \dots s^m\}$ denote the set of training examples where each s^i is a list of part-of-speech tags for a sentence. For a particular training example s^i of length n, let $s^i_1 \dots s^i_n$ represent the part-of-speech tags for the n words in each sentence, where $s^i_j \in D$, the set of all part-of-speech tags. Also, let $spans(s^i) = \{(j,k) : 1 \leq j \leq k \leq n\}$ denote the set of all subranges in the sentence. For a particular span (j,k), then $\alpha_{jk} = s^i_j \dots s^i_k$ is called the *yield* of the span, and the ordered pair $x_{jk} = (s^i_{j-1}, s^i_{k+1})$ consisting of the parts-of-speech tags on either side of the span is called the *context*. To ensure that contexts are well-defined, we may add an implicit symbol to specify the sentence boundaries.

We may then associate each sentence s^i with a phrasal tree structure by defining a map $B : spans(s^i) \mapsto \{c, d\}$ known as the *bracketing* for s^i. Given a rooted phrase tree in which the leaves of the tree correspond to parts-of-speech tags from the sentence s^i, we call a span (j,k) a *constituent* if the yield for some node in the tree is exactly $s^i_j \dots s^i_k$; otherwise, the span is called a *distituent*. The phrase tree is then represented as bracketing by assigning a value to $B_{jk} \in \{c, d\}$ (where c and d stand for constituent and distituent, respectively) for each span (j,k) in the sentence. In the generative constituent-context algorithm, we are concerned with producing binary tree phrase structures, and it is important to note that not all bracketings correspond to valid binary tree structures in which every non-root node has exactly one parent. However, it is the case that every binary phrase tree has a unique bracketing.

Given these concepts, we now define a joint probability distribution specifying the probability of generating a sentence s^i and its associated bracketing B. First, a bracketing B is selected from a multinomial distribution $P(B)$ over all possible bracketings for sentences of the desired length. In the case of the generative constituent- context model, we take $P(B)$ to be the uniform distribution over all valid binary trees in which all bracketings corresponding to non-valid trees are given zero probability mass. Second, for every span (j,k), we select the yield α_{jk} and the context x_{jk} independently. α_{jk} is selected from one of two multinomial distribution over all possible yields, $P(\alpha_{jk}|B_{jk} = c)$ or $P(\alpha_{jk}|B_{jk} = d)$, depending on whether the span appears as a constituent or a distituent in the bracketing. Similarly, x_{jk} is selected from one of two multinomial distributions over all possible contexts, $P(x_{jk}|B_{jk} = c)$ or $P(x_{jk}|B_{jk} = d)$. Note that the distributions for α_{jk} and x_{jk} are taken to be the same for each span of the sentence; thus, the parameters of the model are completely specified by four multinomial distributions.

3 The EM algorithm

Having defined the generative constituent-context model, we now describe how expectation-maximization (EM) may be used to derive values for the parameters of the model. The method used is essentially a variant of the inside-outside algorithm that runs in time cubic in the length of the sentence.

In the expectation step of the EM algorithm, we must compute the expected number of times each of the production rules for yields or contexts are used. For a particular training example s^i, the marginal probability of generating that particular sentence may be found by summing over all possible bracketings for the sentence. Thus, the expected number of times a production rule is used for a sentence s^i is

$$\frac{\sum_B (\#\text{ of times the production rule is used in bracketing } B) \cdot P(S, B)}{\sum_B P(S, B)}.$$

Using the inside-outside algorithm directly to calculate the probability of each yield/context rule is difficult, since the probabilities $P(S, B)$ are dependent on generating the yield and context for every span in the tree, not simply for those nodes that are constituents. Since the observed sentence is fixed during the training process, however, the yield and context for every span in the sentence are also fixed. Thus, we can simply (1) divide each $P(S, B)$ through by the probability for generating *every* span as a distituent and then (2) run the inside-outside algorithm using $\frac{P(rule|B_{jk}=c)}{P(rule|B_{jk}=d)}$ as the probabilities for generating nodes. As a further note, step (1) is unnecessary since the same value is factored out from both the numerator and denominator of each expectation given in the previous paragraph. Thus, we can compute the inside and outside probabilities for generating each span of the sentence as usual, where the probabilities computed must be multiplied through by the probability of generating the yield and context for every span as a distituent in order to give the actual values for $P(S, B)$.

In the maximization step, we compute the maximum likelihood estimates for the probabilities of each production rule; for smoothing, we added 10 counts to each $E[\alpha|B_{jk} = c]$ and $E[x|B_{jk} = c]$, added 50 counts to each $E[\alpha|B_{jk} = d]$ and $E[x|B_{jk} = d]$, and adjusted the number of "observed" total number of rule uses appropriately. This uneven smoothing is identical to that suggested by Klein and Manning and reflects the disproportionate likelihood of random spams to be distituents rather than constituents.

4 Posterior decoding as an alternative to Viterbi parsing

To assess the performance of the grammar induction, we must be able to compute for a given sentence the most likely bracketing based on the model parameters. To do this, we may use the same trick of converting the probabilities for rules as described for the EM algorithm and then use a cubic-time Viterbi algorithm for determining the most likely binary tree bracketing that gives rise to the sentence; essentially, this is the method used by Klein and Manning for evaluating the induction program.

While Viterbi parsing does return the most likely binary tree for a particular sentence, the posterior probability that a given constituent label in the tree is correct may vary over the various constituents in the Viterbi parse. In some cases, the existence of alternative explanations (i.e. alternative conflicting parses) for a sentence based on the grammar induction may mean that Viterbi loses the information from one of the parses altogether, since the parses are mutually exclusive and cannot occur in the same optimal tree.

Based on these observations, it seems reasonable then that the parse returned should not necessarily be the maximum likelihood binary tree but rather should focus on getting constituent labels right. To accomplish this, we tried two different methods as alternatives to Viterbi parsing:

1. Using the inside/outside probabilities as calculated in the EM procedure, we can determine the posterior probability that any constituent labeling is correct given the model parameters. Thus, we can perform a cubic time search for the best parse tree using these posterior probabilities as scores instead of the probability ratios used in the Viterbi parsing; the score of the best tree may then be interpreted naturally as the expected number of constituent labels that are correct, and the procedure will seek to maximize this score.

2. Alternatively, we can compute these posterior probabilities for constituent labels again, but rather than restricting ourselves to a binary parse tree, simply select all constituent labels with a posterior probability above $\gamma = 0.5$, that is, all labels that are more likely to be correct than incorrect. Note that the labels selected in by this method do not necessarily form a binary parse tree.

Essentially, the first method seeks the highest accuracy parse as opposed to the most likely parse and the second method abandons the binary tree requirement to use only labels for which the algorithm is fairly certain.

5 Results and Discussion

To test our systems, we collected the 7422 sentences from the Wall Street Journal section of the Penn Treebank with no more than ten words after POS tags corresponding to punctuation or no text were removed (WSJ-10). For each sentence, only the POS tags were kept, and all lexical information was discarded. We then ran 20 iterations of the EM training procedure and evaluated the system for accuracy in reproducing the bracketings given in the Penn Treebank using the Viterbi parsing and the two posterior decoding methods described. Clearly, the validity of the tests are dependent upon the accuracy of the parsings in the Penn Treebank, but we did not pursue this issue any further.

An important point that should be made is that the generative constituent-context model uses a completely delexicalized training set; in many cases, though, lexical information can make a large difference in affecting the accuracy of a phrase structure parser. Given that the number of parameters that must be estimated by the generative constituent-context model is already quite large, incorporating lexical information would be difficult and is still an open problem for study. Also, it may be considering "cheating" that we were given POS tag information from the Penn Treebank for testing, rather than using POS tags derived automatically from distributional clustering. Klein and Manning tried the algorithm with induced parts of speech and were still able to beat the right-branching heuristic [4].

For comparison, we implemented and tested four other parsers:

1. RANDOM returns a random binary parse tree where the decision of where to split at each constituent node is made by choosing from a uniform probability distribution over all valid possible splittings.
2. LBRANCH always returns the left-branching tree for a sentence.
3. RBRANCH always returns the right-branching tree for a sentence.
4. UBOUND returns the tree given in the Penn Treebank, converted into a binary tree (this adds constituent labels that are not present in the Penn Treebank labelings and so gives an upper bound on the maximum attainable precision for a binary tree parser).

To assess the accuracy of the test system, we computed *precision* as the proportion of predicted constituents that were labeled correctly over the entire training set, and *recall* as the proportion of all true constituents that were labeled correctly over the entire training set. Only nontrivial constituents (i.e. constituents spanning more than one word but less than the entire sentence) were considered. The F_1 value reported is the harmonic mean of the precision and the recall.

The results are given in the table below (note that CCM-POST1 refers to posterior decoding in which the highest accuracy binary tree parse is found, whereas CCM-POST2 is not restricted to finding only binary trees as it selects labels with constituent probabilities greater than 0.5):

Table 1: Performance of various parsing methods on WSJ-10

METHOD	PRECISION	RECALL	F_1
LBranch	11.3	15.2	13.0
Random	18.0	21.4	19.5
RBranch	46.5	62.3	53.2
CCM-Viterbi	53.0	71.0	60.7
CCM-Post1	53.1	71.1	60.8
CCM-Post2	53.9	70.4	61.0
UBound	74.6	100.0	85.5

The obtained values differ from those reported by Klein and Manning, but the ordering of the results and the separation between the techniques are approximately the same, leading us to believe that the difference rests in the method of evaluation and not incorrect implementation of the algorithms themselves. Essentially, the various constituent-context models are all very competitive with one another and all clearly beat the right-branching heuristic in both precision and recall.

As a second test, we evaluated the performance numbers when the value of γ was varied for CCM-Post2. The results are shown below:

Table 2: Performance of CCM-Post2 on WSJ-10 at varying values of γ

γ	PRECISION	RECALL	F_1
0.001	27.2	98.1	42.6
0.01	34.5	93.7	50.5
0.1	44.1	83.7	57.8
0.25	48.6	76.1	59.3
0.5	53.9	70.4	61.0
0.75	58.8	62.5	60.6
0.9	66.7	54.3	59.9
0.99	84.1	29.4	43.6
0.999	94.7	5.7	10.7

Essentially, the goal of this test was to show assess the performance of the constituent-context model as the required expected accuracy is varied. In comparing the values of the γ's to their corresponding precisions, it is clear that the posterior probabilities are indeed correlated with the expected accuracy, although the relationship is nonlinear; thus, the posterior probabilities are good indicators of accuracy, but cannot be used directly to yield an expected accuracy. Determination of the proper fitting function would give a more robust way for estimating the expected precision of a particular constituent label than simply using the posterior probability itself; in fact, such a function would be useful for CCM-Post1 which tries to optimize the expected accuracy of the binary tree returned. We have not performed such a fitting, but this would be an interesting direction for future study. It is also worth noting that the break even point, $\gamma = 0.5$, yields the optimal F_1 value as expected: a label should be selected as a constituent whenever it is more likely to be a constituent

than not.

Finally, we evaluated the performance for CCM-Viterbi, CCM-Post1, and CCM-Post2 at $\gamma = 0.5$ as the number of EM iterations was varied:

Table 3: Performance of CCM-Viterbi, CCM-Post1, and CCM-Post2 on WSJ-10 after varying numbers of EM iterations

ITERATIONS	F_1 **for CCM-Viterbi**	F_1 **for CCM-Post1**	F_1 **for CCM-Post2**
5	44.4	45.8	45.5
10	59.5	59.8	60.1
15	60.4	60.6	60.8
20	60.7	60.8	61.0
25	60.6	60.8	61.0
30	60.4	60.6	60.7
35	60.8	60.8	60.9
40	60.1	60.2	60.3
45	58.5	58.5	54.4
50	54.3	54.4	43.9

Note that the three approaches all used the same training and differ only in their usage of the derived model parameters. Here then, we can observe that after approximately 20 iterations, the EM algorithm is prone to overfitting the training data, resulting in decreased performance of all algorithms. The relative performance of different algorithms is also very consistent with the posterior probability based binary tree parser, CCM-Post1, consistently outperforming the Viterbi parser. Interestingly, when the overfitting is strongly set in at greater than 45 iterations, CCM-Post2 is hurt the most as its probabilities are no longer meaningful predictors of constituency, and there is no binary tree constraint to guide its choice of constituents.

Based on the results given, CCM-Post2 with $\gamma = 0.5$ is the best performing scheme for parsing using the model parameters of the constituent-context model; as long as the EM algorithm has not caused overtraining of model parameters, then this method yields the best parses; for performance that is reliable even when overtraining has occurred, CCM-Post1 may be used as it completely dominates Viterbi parsing with CCM-Viterbi.

As shown by the parsing results, the constituent-context model does provide a sensible and well-performing method for automatically finding likely phrase structures for sentences. Klein and Manning offer a variety of linguistic explanations as to why the constituent-context model itself may be sensible [3]. In the case of posterior probability algorithms for parse finding, (1) the desire to find high expected accuracy alignments and (2) the ability to find constituents that are not restricted to following a binary tree hierarchy provide the linguistic motivations for why such posterior decoding schemes tend to outperform Viterbi parsing.

Use of such posterior decoding techniques is applicable in a wide variety of situations and is not unique to the constituent-context model. Correlating expected accuracy with posterior probabilities generated gives an interesting avenue for future research in improving the performance of expected accuracy maximization techniques.

References

[1] Baker, J.K. (1979) Trainable grammars for speech recognition. *Speech Communication Papers for the 97th Meeting of the ASA*. 547-550.

[2] Carroll, G., Charniak, E. (1992) Two experiments on learning probabilistic dependency grammars from corpora. *Working Notes of the Workshop Statistically-Based NLP Techniques*. AAAI Press. 1-13.

[3] Klein, D., Manning, C.D. (2001) Natural Language Grammar Induction Using a Constituent-Context Model. *Advances in Neural Information Processing Systems 14 (NIPS-2001)*.

[4] Klein, D., Manning, C.D. (2002) A Generative Constituent-Context Model for Improved Grammar Induction. *Proceedings of the 40th Annual Meeting of the ACL*.

[5] Schutze, H. (1995) Distributional part-of-speech tagging. *EACL 7*. 141-148.

[6] Stolcke, A., Omohundro, S.M. (1994) Inducing probabilistic grammars by Bayesian model merging. *Grammatical Inference and Applications: Proceedings of the Second International Colloquium on Grammatical Inference*. Springer-Verlag.