A Maximum Entropy Model For Parsing

Adwait Ratnaparkhi *Salim Roukos* *R. Todd Ward*
<adwait|roukos|tward>@watson.ibm.com

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

1. Introduction

Traditional natural language parsers are based on rewrite rule systems developed in an arduous, time-consuming manner by grammarians. Most of the grammarian's work is devoted to the disambiguation process, first hypothesizing rules which dictate constituent categories and relationships among words in ambiguous sentences, and then seeking exceptions and corrections to these rules. This grammar refinement process is time-consuming and difficult, and has not yet resulted in a grammar which can be used by a parser to analyze *accurately* a large corpus of unrestricted text.

As an alternative to writing grammars, one can develop corpora of hand-analyzed sentences (*treebanks*) with significantly less effort. With the increased availability of treebanks of annotated sentences, one can view NL parsing as simply *treebank recognition* where the methods from statistical pattern recognition can be brought to bear. In [1], we proposed an automatic method for acquiring a statistical parser from a *treebank* of manually bracketed sentences. This method takes advantage of some initial linguistic input, but avoids the pitfalls of the iterative and seemingly endless grammar development process. It relies on data to extract a reliable model for parsing. We used a set of statistical decision trees to assign a probability distribution, $p(T \mid w_1^n)$, on the space of parse trees given the input sentence w_1^n of n words. These decision trees consider various facts about the current constituent, the two constituents to its immediate left[1], and the two constituents to its immediate right. To parse a sentence, we perform a stack decoder search for the most probable tree:

$$T^* = \underset{T \in \mathcal{T}(w_1^n)}{\arg\max}\ p(T \mid w_1^n) \tag{1}$$

where the maximization is over all trees that span the n-word sentence. The resulting parser we believe is the *state-of-the-art* parser: using the PARSEVAL [4] measures, we achieve a constituent recall rate of 79% with a precision rate of 80% on AP newswire sentences of 30 words or less.

[1]See the definition of derivation order for identifying neighboring nodes.

The facts used in the decision tree parser above do not include the detailed tree structure subsumed by a node; rather, it uses the non-terminal label, head word, and head tag of the node to achieve its remarkable results. In this paper, we present a method where more of the tree structure is used in the parsing model. We define a set of features that capture long distance dependency such as parallelism in coordination. These features are then integrated with a Maximum Entropy model into an overall probabilistic model for parsing.

We introduce the decision tree parser in Section 2, describe the Maximum Entropy model in Section 3, describe the feature extraction algorithm in Section 4, give experimental results in Section 5, and present our conclusions in Section 6.

2. Decision Tree Parser

2.1. Derivation history

A parse tree is a n-ary branching tree with each node labeled by either a non-terminal (called a label) or a part-of-speech (called a tag). We introduce a new definition for a derivation of a parse tree by using Figure 1 for the noun phrase "the Enter key". The derivation order defines the order in which the labeled edges of the parse tree are constructed. Each edge from the child to the parent has a composite name that combines the direction and the name of the parent of the node: we use right (r), left (l), up (up), or unary (un) for direction and the actual tag or nonterminal names. r means that the node combines with others to its right to make a constituent, l means that it combines with others to its left, *up* corresponds to interior edges of a constituent, and *un* corresponds to a renaming of a node. The edge from node 2 in Figure 1 is rN (an edge combining to the right to form an N constituent) whereas the edge from node 3 is upN. We only consider bottom-up derivations where a node is extended only after all of the nodes beneath it are extended. The bottom-up leftmost derivation for the tree in Figure 1 is:

unAT rN unVVC unNn upN unNN1 lN.

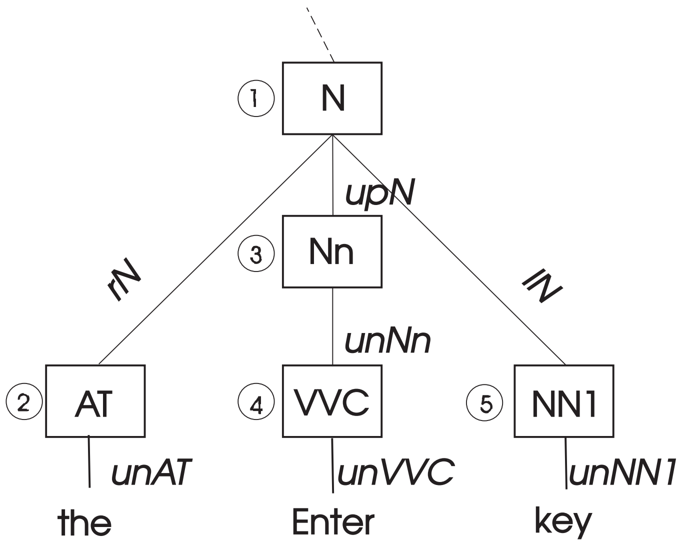

Figure 1: Representation of constituent and labeling of extensions.

2.2. Parsing Model

Denote by e_i the edge at the i-th step in the bottom-up derivation of parse tree T. The probability of a parse tree is the product of the probabilities of the extensions (edges) in the derivation of the tree:

$$p(T \mid w_1^n) = \prod_{0<i\leq|d|} p(e_i \mid context(e_0^{i-1})),$$

where e_0^i is the derivation up to the i-th step in the d-long derivation of T. We use either of two models for an extension: a tagging model when the extension of a node corresponds to a tagging step or a labeled-extension model when a constituent is being formed.

Node Representation We do not use all the subtree information rooted at a node N to condition our probabilistic models. Instead, we represent a node by its node name. Additionally, we use for each constituent node, a word, along with its corresponding part-of-speech tag, that is selected to act as a lexical representative for the constituent. This lexical representative corresponds loosely to the linguistic notion of a head word. As an example, in the noun phrase of Figure 1, the node N is represented by the node name "N", the word "key" and the tag "NN1".

Notation w_i and t_i refer to the word corresponding to the ith token in the sentence and its part-of-speech tag, respectively. N^k refers to the 4-tuple of feature values at the kth node in the current parse state, where the nodes are numbered from left to right. N_l^k, N_w^k, N_t^k, and N_e^k refer, respectively, to the label, word, tag, and extension feature values at the node k. N^{c_j} refers to the jth child of the current node where the leftmost child is child 1. $N^{c_{-j}}$ refers to the jth child of the current node where the rightmost child is child 1.

Tagging Model The tag feature value prediction is conditioned on the two words to the left, the two words to the right, and all information at two nodes to the left and two nodes to the right.

$$p(t_i \mid context) \approx p(t_i \mid \; w_i w_{i-1} w_{i-2} w_{i+1} w_{i+2} t_{i-1} t_{i-2} \\ t_{i+1} t_{i+2} N^{k-1} N^{k-2} N^{k+1} N^{k+2})$$

Labeled-Extension Model The extension feature value prediction is conditioned on the node information at the node being extended, all information from two nodes to the left and two nodes to the right, and the two leftmost children and the two rightmost children of the current node (these will be redundant if there are less than 4 children at a node).

$$p(N_e^k \mid context) \approx p(N_e^k \mid N_w^k N_t^k N_l^k N_c^k N^{k-1} N^{k-2} \\ N^{k+1} N^{k+2} N^{c_1} N^{c_2} N^{c_{-1}} N^{c_{-2}})$$

3. Maximum Entropy Modeling

A feature, f, is a function which detects the presence of a pattern in a parse tree, returning 1 if the pattern exists and 0 otherwise. One simple class of features, level-1 features, detect patterns on depth 1 subtrees. For example, the level-1 feature Nn $\rightarrow$ VVC returns 1 for the tree shown in Figure 1. A more general level-1 feature is N $\rightarrow$ AT $*$ where the wildcard $*$ allows for any string of nonterminals. This latter feature covers all noun phrases that begin with an article.

Similarly, a typical level-2 feature is given by the production:

$$\text{V} \rightarrow [\text{V VVC\_TAG N V}]\ \text{CC\_TAG}\ [\text{V VVC\_TAG N V}].$$

This level-2 feature (which would cover the parse tree of "type a 6 (print) and press the Enter key") captures a certain type of parallelism found in coordination. A more general level-2 feature is:

$$\text{N} \rightarrow \quad [\text{N AT}\mathit{1}\text{\_TAG} * \text{N}]\ ,\text{\_TAG}\ [\text{N AT}\mathit{1}\text{\_TAG} * \text{N}]\ ,\text{\_TAG} \\ \text{CC\_TAG}\ [\text{N AT}\mathit{1}\text{\_TAG} * \text{N}].$$

This feature would include the parse of "a message, a note, or a short document". Other classes of useful features will be described later. We turn now to the presentation of the exponential model for parse trees.

Given a set of features $\{f_i\}$, and a parse tree T, we can define an exponential model by

$$p(T|w_1^n) = \frac{q(T|w_1^n)\, \exp\left(\sum_{i=0}^{k} \lambda_i f_i(T, w_1^n)\right)}{\sum_{T' \in \mathcal{T}} q(T'|w_1^n)\, \exp\left(\sum_{i=0}^{k} \lambda_i f_i(T', w_1^n)\right)},$$

where $q(T|w_1^n)$ is the probability of T given by the decision tree model described earlier and $\mathcal{T}$ is the set of all trees spanning w_1^n. The feature weights λ_i are chosen so that the model's expected feature values are constrained to the expected values $\tilde{\mathrm{E}} f_i$ of those features using the empirical distribution of the training data, $\tilde{\mathrm{p}}(T, W)$:

$$\tilde{\mathrm{E}} f_i = \sum_{T,W} \tilde{\mathrm{p}}(T, W) f_i(T, W).$$

The feature weights can be computed by using the iterative *Generalized Iterative Scaling* algorithm of Darroch and Ratcliff [2].

As discussed in [3], the above exponential model is obtained by the ME principle: find the model closest to q in the Kullback sense subject to k linear constraints on the expected values of the k feature functions, where the Kullback distance between p and q is given by

$$D(p, q) = \sum p \ln(p/q).$$

4. Feature Selection

We define a large pool of features, denoted by $\mathcal{P}$, that apply to parse trees. Since these features are automatically generated, many are redundant, and others have minimal linguistic content. To identify the most useful features, we proceed as follows:

1. Initialize $\mathcal{M}$ the set of features in the model to be the empty set.
2. Select the best feature from $\mathcal{P}$ using Delta-Likelihood rank
3. Add it to $\mathcal{M}$
4. Train Maximum Entropy Model, using features in $\mathcal{M}$
5. repeat from (2)

Delta-Likelihood At step (2) in the search, we rank all features in $\mathcal{P}$ by estimating their potential contribution to the log-likelihood of the training set. Let q now denote the conditional probability distribution of the model with the features currently in $\mathcal{M}$. Then for each $f_i \in \mathcal{P}$, we estimate, by computing only λ_i and holding the other λ's fixed, the probability distribution p_i that results when f_i is added to the ME model:

$$p_i(T|W) = \frac{q(T|W)\, e^{\lambda_i f_i(T,W)}}{\sum_{T \in \mathcal{T}_N(W)} q(T|W)\, e^{\lambda_i f_i(T,W)}}$$

Ideally, we have to normalize the model by summing over all trees that span sentence W; but since this is impractical, we truncate the sum to the top N trees proposed by the decision tree parser for sentence W. In the same way, we sum over only the top N trees when estimating the feature weights in the full ME model.

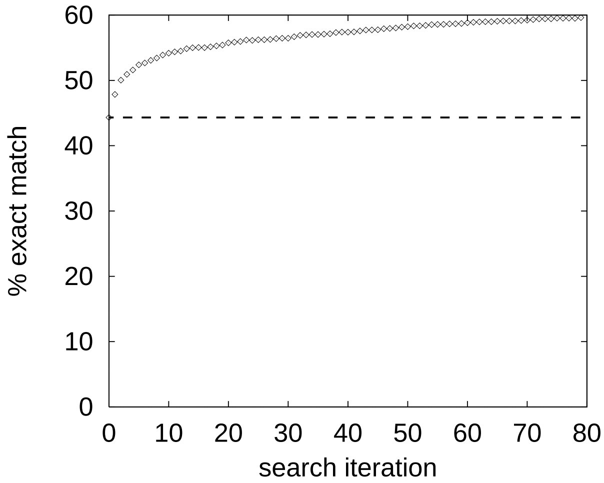

Figure 2: Exact match of ME parser on training set (dashed line is decision tree parser).

We then compute the increase in (log) likelihood for each new model:

$$\delta L_i = \sum_{T,W} \tilde{\mathrm{p}}(T, W) \ln p_i(T|W) - \sum_{T,W} \tilde{\mathrm{p}}(T, W) \ln q(T|W).$$

and choose the feature with the highest δL. Features redundant or correlated to those features already in $\mathcal{M}$ will produce a zero or negligible δL, and will therefore be outranked by genuinely informative features. The chosen feature is added to $\mathcal{M}$ and used in the ME Model. We may also add a block of b features instead of adding one at a time to reduce computational complexity.

5. Experimental Results

Task Domain We have chosen computer manuals as a task domain. A corpus was treebanked by the University of Lancaster. The Treebank uses 17 non-terminal labels and 240 tags[2].

Experiment The Treebank of 32,000 sentences was divided into a set of 16k for building the decision tree parser, a set of 14k for the ME training, and a set of 2k for testing. The set of 14k was parsed and the top N parses (N ranging from 80 to 300) for each sentence were produced. This set was used to find a subset of features and to estimate the exponential model. The features were selected from a pool that included:

- The level-1 and level-2 features described earlier.

[2] we have projected the tag set to 193

```
J --> J CC_tag J
TOP --> [Fa CS_tag * Fa] ._tag
V --> VM_tag [V V CC_tag V V]
nounAttach( authority, to)
V --> [V VVI_tag N P V] CC_tag
      [V VVI_tag N P V]
Fa --> CS_tag [S S CC_tag S S]
```

Figure 3: Sample of selected features.

- Word bigram features to capture linguistic phenomena such as verb subcategorization; for example, in a verb phrase, use a feature for the main verb and preposition as $verbAttach(select, from)$.
- A variety of features that we identified as problematic for the decision tree parser.

A parse produced by the parser is judged to be correct under the "Exact Match" criterion if it agrees with the Treebank parse structurally and all nonterminal labels and tags agree.[3]

Figure 2 shows how the parser accuracy improved on the training set as the number of features that are added is increased. The cumulative number of features selected is shown in Table 1 which also shows the corresponding performance on the test set. Figure 3 shows examples of some of the features that are selected from the pool of features. The ME model improves performance by a modest 14% on the test data. We are surprised by the big disparity in performance between the training and test set even when only 688 features are added. The training set improvement is more than double the test set's. In addition, the approximation on the top N parses leads to a biased estimate of the denominator. While we have not yet implemented a careful unbiased estimator, we tried the following "hack". We estimate the weights of the ME model by using a uniform initial model on the top N parses obtained by the decision tree.

[3] A sample of 5000 sentences (a training set of 4000, a development test of 500, and an evaluation test of 500) is available by request from roukos@watson.ibm.com.

Search Iterations	# features	Exact Match
0	0	35.9%
10	388	39.3%
20	653	41.1%
40	1113	41.0%
80	1930	40.7%

Table 1: Exact Match on test set as a function of the number of features.

When parsing test data, we use the decision tree score with the new weights. This leads to an improved Exact Match of 42.9% for the 1930 feature model. This corresponds to an improvement of almost 20%. We are still exploring how to more correctly solve the parameter estimation problem.

6. Conclusion

We presented a "linguistically" naive parsing model that has a parsing accuracy rate that we believe is state-of-the-art. We introduced a model that allows the incorporation of traditional rules into the parsing model. The rules are selected automatically from a corpus. The resulting model leads to a small improvement in parsing accuracy. However, we still need to develop an unbiased estimator for the weights and expect the unbiased parameters to yield a better improvement in performance.

Acknowledgment

This work was supported in part by ARPA under ONR contract No. N00014-92-C-0189.

References

1. Jelinek, F., Lafferty, J., Magerman, D. M., Mercer, R., Ratnaparkhi, A., and Roukos, S., "Decision Tree Parsing Using a Hidden Derivation Model." *Proceedings of the ARPA Workshop on Human Language Technology*, Princeton, New Jersey, March 1994.
2. Darroch, J.N. and Ratcliff, D., "Generalized Iterative Scaling for Log-Linear Models", *The Annals of Mathematical Statistics,* Vol. 43, pp 1470–1480, 1972.
3. Berger, A., Della Pietra, S., Della Pietra, V., "Maximum Entropy Methods in Machine Translation," *Manuscript in preparation.*
4. Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F. Kalvans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., Strzalkowski, T., "A Procedure for Quantitatevily Comparing the Syntactic Coverage of English Grammars," *Proceedings of the DARPA Workshop on Speech and Natural Language*, Pacific Grove, California, February 1991.