Reranking Probabilistic Parses with Supervised Learning

Chirag Nagpal
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania
chiragn@cs.cmu.edu

Abstract

Algorithms like CKY can be used along with Probabilisitc Context Free Grammar to carry out parsing of Natural Language sentences. However, CKY like algorithms are greedy, as they produce parses only with the highest probability with respect to the training grammar. In this study, we explore the use of supervised learning, to rerank parses based on prior evidence of more likely parses.

1 Introduction

CKY like dynamic algorithms can be used along with Probabilistic grammars to generate parses over a given Natural Language sentence, however these techniques, being greedy dynamic programs, are deterministic and generate only the most probable parse for a given sentence.

There have been improvements in parsing, by using techniques like grammar markovisation(Klein and Manning, 2003), which lend more statistical support and local context while carrying out parsing, However, the inherent algorithm, CKY is still deterministic with respect to the Grammar.

A good compromise was the use of coarse-to-fine pruning(Charniak et al., 2006), that employs a coarse grammar to prune parses with low probability and then fine grammar. Intuitively, this can be thought of as a technique that allows only very natural parses, while high scoring special cases are pruned out as outliers due to lack of statistical support.

Reranking, on the other hand treats parsing as a supervised learning problem, considering that the best parses of a sentence all share certain common structural and statistical properties that are not unique to the sentence they parse. In other words, one can assume that good parses across sentences look similar.

The motivation behind this generic assumption, can be easily understood with a parse of a sentence that is right branching as opposed to centrally expanding. Intuitively one would assume the more central parse to be better, than one that is completely right branching.

In this study we experiment with training supervised classifiers on features extracted from the top k-parses using the Berkeley parser.(Petrov et al., 2006)

2 Corpus Statistics

	Tr=15,Te=15	Tr=inf,Te=40
Training Trees	9753	39832
Test Trees	421	1578

Table 1: Corpus Statistics

3 Feature Extraction

Since we treat reranking as a supervised learning problem, the performance is highly sensitive to the extracted features from the trees. We extract features that are shared globally across multiple parse trees. A list of features as extracted is below

- Position of Gold Tree in K-Best List
- Length of Sentence
- Size of Parse Tree
- Size of Longest Right Branch
- Difference of Longest Right Branch & Size of Parse Tree
- The Rule present in the Grammar (Label ->Children)
- Right Child and PoS Tag of Span and Rule
- Left Child and PoS Tag of Span and Rule
- Right Context of Span and Rule

- Left Context of Span and Rule
- Span Structure

These features are as described in the paper (Hall et al., 2014)

It is interesting to note how the decoding time changes with change in training dataset size. From Figure 2 it is clear that the decoding time increases with the increase in training data and then decreases. This behaviour can be explained easily by the use of cache table in our model.

For models built on small datasets, large number of trigrams are unseen, and the model backsoff to a lower order model, the decoding for which is fast. One the other hand as the training dataset increases, the model not only has to take into account the higher level ngram model, but also recursively decode lower level model, increasing the decoding time. As the dataset size increases, so does the repetition of certain trigrams, thus the model would have higher number of cache table entries and the probability of a cache miss will decrease, speeding the decoding time for the model.

4 Classification

We experiment with 3 different Classifiers, 1) the simple Perceptron model to learn a linear decision boundary, 2) a Maxent Classifier and Finally 3) a Max Margin SVM classifier.

4.1 Linear Perceptron

Perceptron is one of the simplest techniques to train a linear decision boundary to perform binary classification. To train perceptron, we use the K-Best Parses as training data along with the Gold trees.

We iterate over all the K-best Trees and find the one which scores the highest. If the one with the highest score is same as the Gold Tree, we ignore and move to the next set of K Best Parses, else we update our weights.

This is carried out iteratively, till the perceptron converges. In practice however, most data is never linearly separable and hence, the perceptron is never guaranteed to converge.

	Prec	Rec	F1	Time
Tr=40, Te=40	83.59	84.5	84.04	5932

Table 2: Linear Perceptron

4.2 Max Entropy

Max Entropy or Logistic Regression Models aim to reduce the Log-Loss on the Dataset. MaxEnt models also generally use L2 regularisation on the weights.

In order to perform Max Ent classification, we used Stochastic Gradient Descent on the Log Loss of the training Data. The dimensionality of our dataset can be very large in terms of the weights. However, features extracted from a tree are sparse.

Thus, we perform the SGD on Loss function, without any regularisation.

	Prec	Rec	F1	Time
Tr=40, Te=40	77.29	85.18	81.04	4694

Table 3: Max Ent

4.3 Max Margin Classification

In our experiments Max margin classification with a 0/1 Loss outperformed Linear Perceptron and Max Ent models. For Max Margin classification, we iterate over all samples in the KBest List and find the one that violates the decision function the most and use it for the training by assigning it a 1 Loss.

	Prec	Rec	F1	Time
Tr=40, Te=40	86.47	85.88	86.17	19106

Table 4: Max Margin Classifier

5 Conclusion

In this study, we implemented 3 different supervised classifiers to rerank parses generated from the berkeley parser. In our implementation, the Max Margin classifier performed the best, with a score of 86.17 F1.

Our Max Entropy model performed the worst, one of the reasons for the same could be lack of regularisation (which we avoid due to computational complexity), thus leading to overfitting.

In all of the experiments, parameter tuning was found to be challenging. However, for the perceptron, the parameter tuning was most challenging. It is possible that our perceptron probably did not converge to an optimal solution, generally, in such high dimensional, sparse feature spaces, data is linearly separable, allowing perceptron to perform decently, however, our perceptron could not outperform the 0-best baseline.

More number of iteration, along with a decaying learning rate, could perhaps address some of these challenges for the perceptron.

Acknowledgments

We would like to convey our gratitude to Professor Taylor Berg-Kirkpatrick, Wanli Ma and Kartik Goyal for all the help for this assignment.

References

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R Shrivaths, Jeremy Moore, Michael Pozar, et al. 2006. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 168–175. Association for Computational Linguistics.

David Leo Wright Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *ACL (1)*, pages 228–237.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.