CS395T Project 2: Shift-Reduce Parsing

1 Problem Statement

In this work, I implemented shift-reduce parsing algorithms to create dependency trees for English sentences. The two main approaches were: a greedy model that used local features in a multiclass logistic regression, and a global model that used the beam-search algorithm to parse trees, with a structured perceptron making decisions. The global beam model showed promise for achieving the best results, and given enough time and space resources this could have been tested. Nevertheless, the best achieving algorithm was still the greedy model. The best performing greedy model achieved a unlabeled attachment score of 79.42. I also investigated the performance of each model on various languages other than English.

2 Models

2.1 Greedy

2.1.1 Implementation

The greedy model performed decoding by making local action decisions at each step of the shift-reduce framework until reaching a finished state. I used a multiclass logistic regression classifier (aka perceptron with softmax activation) to make local decisions based on the provided set of extracted features. The decision with the highest probability was selected. In order to prevent illegal actions, I hard-coded the following restrictions on the model: if stack size is less than 2 then shift, if buffer size is zero then don't shift, if root is the second stack item, don't left arc. For training, I implemented an AdaGrad optimized version of stochastic gradient ascent (SGD) using predicted softmax probabilities to find expected feature values and gradients.

2.1.2 Configuration

Stochastic gradient ascent was run for 3 epochs using the provided AdaGrad optimizer. The lambda parameter was set to 1e-8, which showed to improve performance of the default 1e-5 from 77.49 UAS to 79.42 UAS. Weights were initialized to zeros.

2.2 Global Beam

2.2.1 Implementation

The global beam model parsed sentences by making 2n predictions, and using the beam to keep track of the best predictions made thus far. In a similar manner to the greedy model, actions were determined by taking the argmax of the output from a generalized perceptron, as seen in (Zhang and Clark, 2011). For most of the trials, I used the averaged perceptron model as a form of regularizing the final weight vector, which showed to slightly improve performance (Zhang and Clark, 2011). I also enforced the same legal action restrictions discussed in Section 2.1.1. Global training was done by decoding a whole sentence, and using the difference between accumulated features and accumulated gold features as the gradient update to the weights. Unlike for the greedy model, I did not get around to implementing AdaGrad for the structured perceptron training in the global beam model.

2.2.2 Configuration

Stochastic gradient ascent was run for 3 epochs using a constant learning rate of 1.0. Weights were initialized to zeros.

2.3 Extensions

2.3.1 Beam Manipulation

After reading some analysis presented in (Zhang and Nivre, 2012), I decided to investigate how manipulating the beam size between training and testing could influence results. In (Zhang and Nivre, 2012), they found that providing larger beam sizes at test time did not improve performance, and any mismatch between training and testing beam sizes significantly hurt performance. In this analysis, I used smaller beam sizes due to computational limits.

2.3.2 Non-English Languages

As another extension, I experimented with other non-English languages in order to observe which languages perform worse or better, as well as which model does better at each language. To avoid using too much memory in the feature cache, I truncated the number of sentences in the Chinese training set from roughly 59,000 to 40,000.

3 Results

3.1 Main Results

As Table 1 indicates, the greedy model with local decisions performed better than the global beam model. However, as expected, performance of the beam model got slightly better and better with larger and larger beams.

Model	Learning	UAS
Greedy	SGD+AdaGrad	**80.15**
Global beam=1	SGD+avg perceptron	73.65
Global beam=5	SGD+avg perceptron	74.10
Global beam=5	SGD+perceptron	73.55
Global beam=8	SGD+avg perceptron	74.96

Table 1: Main results. For model descriptions Global beam=x, x indicates the beam size.

Nonetheless, it was not feasible to experiment with large beams, as a beam size of 8 took 40-50 minutes per epoch. In addition, the average perceptron method as a form of regularization seemed to do well, as indicated by the performance bump from the third to second row. Finally, it should also be noted that the greedy model was trained with the AdaGrad optimizer, whereas the global model used a constant learning rate of 1.0.

3.2 Beam Manipulation

As Table 2 illustrates, the results were consistent with (Zhang and Nivre, 2012). Any experiment with mismatching training and testing beam size did significantly worse than the others.

Train beam size	Test beam size	UAS
1	1	73.65
1	5	59.19
5	1	40.23
5	5	74.10

Table 2: Results for manipulating the beam sizes at training and testing times.

3.3 Non-English Results

Table 3 shows the results on non-English languages. Surprisingly, both models did better on Chinese and Japanese relative to English. This trend differs that presented in (Zhang and Clark, 2011), though this is likely due to custom model tuning for English data.

Language	Sentence count	Model	UAS
Arabic	1460	Greedy	70.50
Arabic	1460	Global	67.60
Chinese	40000	Greedy	83.50
Chinese	40000	Global	79.85
Japanese	17044	Greedy	89.70
Japanese	17044	Global	81.70

Table 3: Results for various non-English languages.

Moreover, across all languages, the greedy model outperformed the global beam model, by roughly the same margin for each language.

4 Discussion

Although the UAS marginally increases with bigger and bigger beams, the gains found in Table 1 are not overwhelming, particularly given that the greedy model still does significantly better. This is also partially an unfair comparison, as the greedy model parameters were tuned using AdaGrad whereas the global beam model was tuned using a constant learning rate of 1.0. It would be beneficial to compare the models when training hyperparameters are more or less consistent. Given the speed of training and testing on the global model with large beams, the greedy model seems like the optimal algorithm for dependency parsing.

In addition, I found that mismatching training and testing beam sizes led to relatively terrible performance. This is despite the fact that, for example, a globally trained model with a beam size of one is given more information during decoding of testing examples with a beam size of five. This may be a result of the weights of the structured perceptron over-fitting or at least specializing to a particular beam size.

As discussed by (Marton et al., 2013), morphologically rich languages tend to use feature like word order less in syntactic modeling. This could explain why the results obtained for Arabic are

much worse than that of English, as our features primarily rely on word order and part of speech. This could also explain why Chinese and Japanese were easier for the models presented in this analysis. Both of those languages rely less on word morphology and tenses, and primarily use word ordering to establish syntactic dependencies.

References

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of modern standard arabic with lexical and inflectional features. *Computational Linguistics* 39(1):161–194.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics* 37(1):105–151.

Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *COLING (Posters)*. pages 1391–1400.