Active Learning for Neural Dependency Parsing

Zeyuan Hu
Computer Science Department
University of Texas at Austin
Austin, Texas
iamzeyuanhu@utexas.edu

Abstract

Constructing a corpus is a labor-intensive task. We always want to have as few annotated examples as possible while maintaining a good model performance. Active learning explore this idea by selecting the most helpful training examples to be annotated by human experts and learn from them. In this writeup, we explore the idea of active learning on the neural dependency parsing. Empirical results shows that active learning can truly be helpful to our goal.

1 Introduction

The idea of active learning for training parsers from treebank data is that we want to have as few annotated training sentences as possible while obtaining a good performance. It can be costly to construct large training corpora like the Penn Treebank (Marcus et al., 1994) in terms of human effort. Thus, we want to pick the a set of training sentences that have the maximum potential performance gain for our parser and have those sentences labeled by the human expert. In other words, active learning can be viewed as a way of minimizing the corpus size and reducing the human effort (Hwa, 2000).

Unlike traditional learning approach, active learning system not only needs to learn the parsing for each sentence but it also needs to pick training sentences from a pool of "unlabeled" training sentences. We call the process of selecting next batch of training sentences from "unlabeled" training pool as *sample selection*. The goal of sample selection is to minimize the amount of data that needs to be annotated to achieve a desired level of performance. For statistical parsing, a training instance is a sentence and the annotation is a parse tree supplied by a linguistic expert. In this paper, we implement three types of sample selection *policy* and evaulate their performance impact on dependency parsing.

2 Implementation

We use neural dependency parser from Stanford (Chen and Manning, 2014) to evaluate our sample selection policy on Penn Treebank (Marcus et al., 1994). First, we set aside a portion of corpus as our test data and we also pick a portion of data as our initial training data. The rest of corpus is assumed unlabled. Then, we train the neural dependency parser on a small randomly-selected sample of annotated training sentences to get started. Then, we feed in the trained model to the policy module to determine next batch of training sentences we pick from the "unlabeled" training pool. The newly picked training sentences are combined with the initial training set to form the new training set for the next iterations. To meausre the performance of sleection policy, we test the learned model against the test data and calculate Labeled Attachment Score (LAS) in each iteration. Our active learning procedure described above is shown in Figure 1, which is similar to Hwa (2000)'s experiment setup.

We implement four sample selection policies: *Random*, *Length*, *Raw*, and *Margin*. *Random* policy asks the system to choose a random set of sentences from the "unlabeled" training pool. *Length* chooses the batch based on the sentence length (i.e., number of words within a sentence). Intuition for this

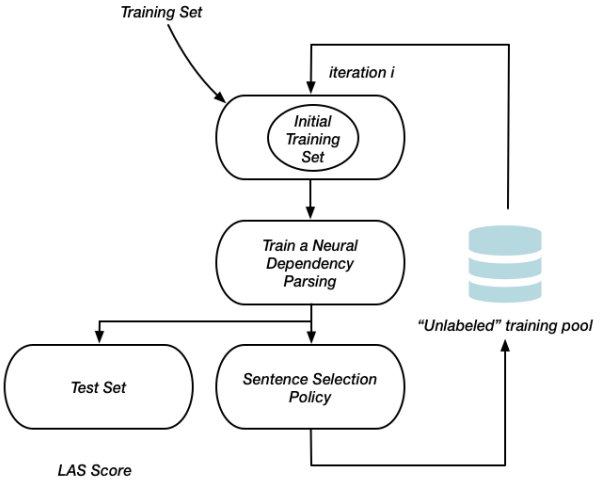

Figure 1: Active Learning Implementation Architecture

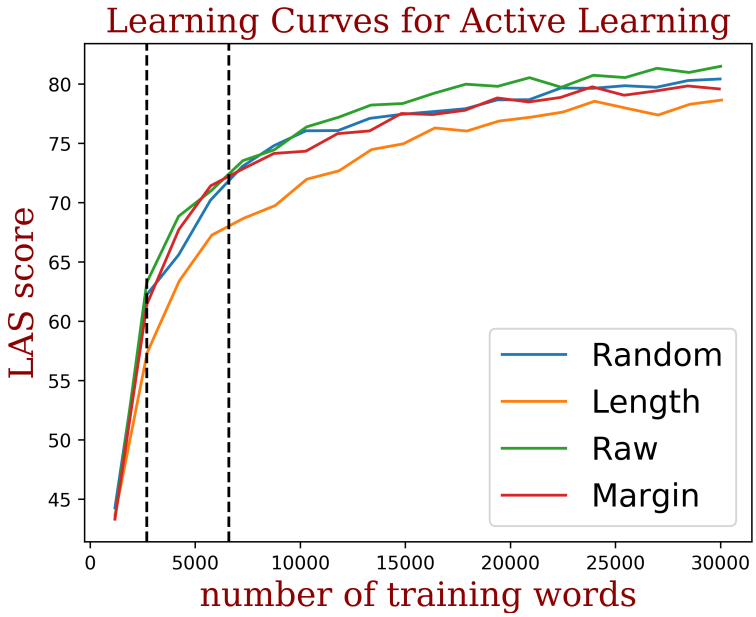

Figure 2: Learning Curves for Sample Selection Policies

policy is that long sentences are usually more syntactically complex and so their parses are inevitably less certain. We first parse the "unlabeled" training pool and both *Raw* and *Margin* select the sentences with lowest probabilities. The lower the probability, the higher the uncertainty. By obtaining feedback on the cases in which the parser is most uncertain, the parser hopes to learn more than obtaining labels on random sentences in which its existing model is perhaps already quite confident in, and from which it would therefore not learn much. *Raw* selects sentences based on *normalized* raw probability, which is calculated by taking the product of the probability assigned to every transition taken to get that parse. *Margin* select sentences based on *normalized* margin probability, which is done by taking the product of the margin between probabilities assigned to two top transitions at every step. *Normalized* means taking $2n$th (n is the length of sentence) root of probabilities to compensate the fact that longer sentences have lower probabilities due to more shift-reduce decisions that the parser has to make.

3 Experiments and Analysis

We extract first 50 sentences from section 00 to form our initial training set. For the "unlabeled" training set, we concatenate sections 01-03 of WSJ, which gives us 4485 potential training sentences (107272 words). For test set, we use WSJ section 20. In each iteration, we use policy to select approximate

Policy	LAS	Training Time (s)	Training Set (words)
Random	80.44	**19393** (0)	29976
Length	78.66	19805 (+412)	**30047**
Raw	**81.51**	20408 (+1015)	30006
Margin	79.60	19564 (+171)	29942

Table 1: Sample Selection Policy Performance

1500 words of additional training data from "unlabeled" training set. All the experiements are performed on a machine that has 4 Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz processors and 4GB of memory. We run parser for 20 iterations and we set `maxIter`, the number of training iterations to complete before stopping and saving the final model, to be 500. The rest configuration of neural dependency parser takes default values[1]. Experiment result is shown in Table 1 and corresponding learning curves are shown in Figure 2. The commands along with the parameters for the program to reproduce the experiment can be found in `Makefile` shipped with the writeup.

From the result, we see that *Raw* outperforms *Random* selection policy while *Margin* and *Length*

[1] `https://nlp.stanford.edu/software/nndep.shtml`

stay below *Random* performance. To further verify the observation, another set of experiments is run and we have *Random*: 79.43, *Length*: 78.78, *Raw*: 81.40, and *Margin*: 80.41. We see that *Raw* is still the best policy, *Random* and *Margin* stay relative close, and *Length* be the worst policy. If we take a look at learning curves, we find that active learning policies perform better than random selection in terms of learning rate. The good performance of *Raw* indicates that actively select most uncertain training sentences can truly improve the parsing performance both in terms of final LAS score and the learning rate, which majorly because of picking sentences that have highest learning values.

When comparing different policies for measuring uncertainty, we see that *Length* is not a good policy. This is probably becasue of two reasons. First, there is no strong correlation between the length of the sentence and the uncertainty of the parses. Some sentences may be long but their syntactic structure might be simple. Another possible reason is that the parser is trained bias towards long, complex sentences and fail to learn the short, simple sentences. Overall, *Length* policy leads to an offline training scheme in the sense that all the training batches can be predetermined: the parser performance on the current batch has no impact on how the next batch of sentences should be selected. *Random* and *Margin* stay very close to each other possibly due to the fact that *Random* already picks the sentence with the most gradient from a pool of random-selected sentences, which makes *Random* effectly perform active learning on the provided batch. If we take a look at Figure 1, we can see that both *Margin* and *Raw* outperforms *Random* around 5000 words, which suggests that both *Margin* and *Raw* learn much more from sentences they intentionally pick than *Random* that select sentences randomly. When compare *Margin* with *Raw*, we can see that *Raw* is a better metric of uncertainty. Imagine a sentence with two words. The first word's action is very certain with probability 1 and the second word's actions are distributed as 0.6 and 0.4 for the top two. Then, under *Raw* policy, this sentence's probability is $1 \times 0.6 = 0.6$ while for *Margin*, it is $1 \times 0.2 = 0.2$. Under *Raw*, this sentence is fairly certain while for *Margin*, this sentence is very uncertain. However, as one can see, parser is only unsure about the last action but the top action is still above 0.5, which is fairly certain. However, the parser's uncertainty get exaggerated under *Margin* policy. In other words, the low margin probability may not be a good indicator to the uncertainty of the whole parsing, which means the sentence falsely pick the sentence that has few learning value.

If we take a closer look at the learning curve, we can see that active learning is most helpful when the number of training words between 2694 and 6600 as shown in dot lines in Figure 1. Within the region, the LAS score of both *Margin* and *Raw* outperforms *Random*. The reason for this phenomenon is that the parser learns more from the sentences selected by the active learning policies than by random. After 6600 number of training words, *Raw* still outperforms *Random* but the gap between them gets closer. One possible explanation is that the "unlabeled" training pool gets smaller and even by *Random*, the sentences that have high learning values will be selected with higher and higher chances. Thus, the gain from active learning gets decreased.

When compare our result with Hwa (2000)'s experiemts' results, we can see that our experiement is closest to Hwa's first experiment setup, which experiments with different sample selection policy from an abundant supply of unlabeled data. When we compare results with Hwa's, we find that both experiments show that *Length* policy is less helpful. In addition, active learning policies have faster learning rate (i.e., the slope of learning curves) than the baseline. Hwa's result shows that tree entropy is the most effective selection policy while ours is *Random*, which share same intuition about measuring uncertainty from probabilistic point of view. Our result is different from Hwa's result in terms of difference among *Random*, *Length*, and *Raw* performances. Hwa shows that on average the score of *Random* and *Length* are the same. One furture work is to show whether Hwa's result holds in our case when we repeatedly run several trails of our experiments. Additionaly, Hwa's result shows they can use much less *brackets* when using tree entropy policy. It seems to suggest that tree entropy is a better uncertainty measurement than *Raw*. This may be true because tree entropy is calculated by considering all possible parses while *Raw* considers only the most probable parsing.

4 Conclusion and Future Work

In this writeup, we show that given a fixed amount of unlabeled training examples, active learning strategy can learn much more than the random selection baseline. We compare different ways of measuring uncertainty about training examples and show that *Random* policy performs the best. In the future, we may explore the composition of test set to see if parser can have great performance on long sentences while have subpar performance on short sentence to explore the root cause of failure of *Length* policy.

References

[Chen and Manning2014] Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics.

[Hwa2000] Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 45–52, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Marcus et al.1994] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.