Assignment 3 : Active Learning

Kartik Sathyanarayanan (EID: ks46373)

April 9, 2018

1 Introduction

The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. Since the learner chooses the examples, the number of examples to learn a concept can often be much lower than the number required in normal supervised learning. An active learner may pose queries, usually in the form of unlabeled data instances to be labeled by an oracle (e.g., a human annotator). Active learning is well-motivated in many modern machine learning problems, where unlabeled data may be abundant or easily obtained, but labels are difficult, time-consuming, or expensive to obtain.

The key hypothesis behind active learning is that if the learning algorithm is allowed to choose the data from which it learns, it will perform better with less training. Consider that, for any supervised learning system to perform well, it must often be trained on hundreds (even thousands) of labeled instances. Sometimes these labels come at little or no cost, such as the the *spam* flag you mark on unwanted email messages. In these cases, data about such labels is available for free, but for many other more sophisticated supervised learning tasks, labeled instances are very difficult, time-consuming, or expensive to obtain. Active learning systems attempt to overcome the labeling bottleneck by asking queries in the form of unlabeled instances to be labeled by an oracle (e.g., a human annotator). In this way, the active learner aims to achieve high accuracy using as few labeled instances as possible, thereby minimizing the cost of obtaining labeled data.

In this assignment, we will aim to simulate an active learning experiment for dependency parsing. The goal is to pick training examples wisely in order to minimize the amount of data that needs to be annotated to achieve a desired level of performance. In this experiment, a training instance is a sentence and the annotation is a parse tree. Initially, the system is trained on small set of randomly selected annotated instances. Using this learned model, the system selects a small batch of the most useful (or uncertain) examples for annotation, add them to the training set and retrain the model with the modified initial set of sentences. Based on what it has learned, the system repeatedly selects small batches of examples to add to the training set until the desired level of performance is reached.

2 Methodology

In this experiment, a corpus of completely annotated data from the Wall Street Journal is used to simulate active learning. A disjoint set of sentences is kept aside for testing. Another set of around 4500 sentences from WSJ is chosen as the "assumed" unlabeled set. When the system requests annotation for a particular instance, the annotation for that instance is retrieved from the dataset.

We wish to compare four different ways of actively selecting data for annotation.

1. Random : We select a fixed number of samples (1500 words) from the unlabeled set after every epoch. This will be our baseline to evaluate the effect of active learning.
2. Sentence Length : After every epoch, we select the longest sentences from the unlabeled set. Long sentences usually contain complex structures and are very hard to parse.
3. Raw score : We select sentences from the unlabeled set that the model is most uncertain (least confident) about. For each sentence, we evaluate the raw parse probability normalized by the sentence length.

   $$RawScore = \sqrt[2n]{\prod_{i=1}^{n} p_{shift}(i)p_{reduce}(i)} \tag{1}$$

   where n is the total number of words in the sentence and $p_{shift}(i)$ and $p_{reduce}(i)$ are the probabilities of i^{th} shift and reduce operations in parsing the sentence.
4. Margin score : We select sentences from the unlabeled set that the model is most uncertain (least confident) about. For each sentence, we evaluate the difference between probabilities assigned to the top two transitions at each step of parsing normalized by the sentence length.

   $$MarginScore = \sqrt[2n]{\prod_{i=1}^{n} p'_{shift}(i)p'_{reduce}(i)} \tag{2}$$

   where n is the total number of words in the sentence and $p'_{shift}(i)$ and $p'_{reduce}(i)$ are the differences in the top two transition probabilities of i^{th} shift and reduce operations in parsing the sentence.

2.1 Implementation

The *DependencyParserAPI* class implements the active learning simulation. *DependencyParserAPIUsage* class is a wrapper for the former class and contains a main() method that takes the input arguments as the paths to seed (initial), training and test data, embeddings, number of sentences to choose from initial and training set, the method of learning and the model path. In each epoch, the dependency parser is trained on the seed (labelled) set for *maxIter* iterations. The model is then used to sample unannotated sentences from the unlabeled set based on the method of learning. In the case of **Random**, we choose K sentences at random such that the total number of words just exceeds 1500 words. For **SentenceLength**, we reverse sort the sentences with respect to length and choose top K sentences at every epoch. For **RawScore** and **MarginalScore**, we compute the raw score and marginal score of the sentences in the unlabeled set after every epoch, sort based the scores, and choose the first K sentences. The candidates from the unlabeled set are added to the labeled set after every epoch and retrained in the next epoch with these appended candidate sentences.

3 Experiments

The model is trained and evaluated on WSJ data with annotations in the Penn Tree-bank format. Initially, a seed set of 50 sentences is used to train the model for the first time. An unlabeled data set of 4500 sentences approx. is used for active selection. After every epoch, candidates from the unlabeled set chosen based on the policy is removed and added to the seed (labeled) set. The experiments were run with $maxIter = 500$ and $number\_of\_epochs = 20$.

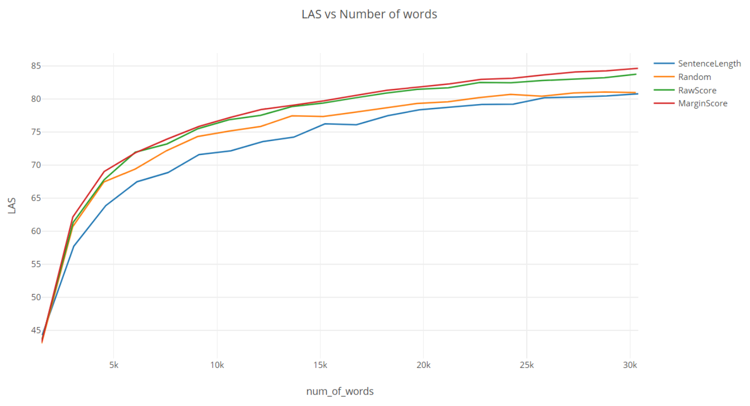

Figure 1: Performance of the 4 methods/policies as a function of the number of words in the seed/labeled set

3.1 Discussion

3.1.1 Uncertainity vs Random Sampling

From Figure 1 we can see that uncertainity sampling methods perform better than random sampling. This was obvious and expected because the parser might choose a sentence which its already confident (certain) about. So, the model doesn't learn much from these candidate sentences in the unlabeled set. On the other hand, **RawScore** and **MarginScore** methods help the model choose sentences from the unlabeled set that its least confident or most uncertain about. The performance of the **SentenceLength** method was found to be inferior to **Random** sampling. This might be attributed to the presence of *garden path* sentences in the training set. Random sampling helps in avoiding these sentences to be chosen as candidates and probably that's why it performs better than Sentence Length.

3.1.2 Performance of active methods across the learning curve

Initially, all methods (except SentenceLength) perform equally well. This is because the learner selects the active samples that have been trained on very less sentences, leading to bad sampling quality (almost random). As the number of training sentences increase, the Uncertainity methods sample better than random causing them to perform better than Random sampling.

3.1.3 Comparison between active learning methods

Over the course of 20 epochs, both RawScore and MarginalScore methods perform almost equally well and better than Random sampling. And, Random sampling performs better than SentenceLength as mentioned in the 3.1.1. Uncertainity based sampling performs the best because it enables the parser to choose sentences from which it can benefit the most. I expected MarginScore to outperform RawScore by a slightly greater margin than shown in Figure 1. The marginal probability between the top two parses reveals more information than the absolute probability of the top parse usually. The larger the marginal probability the less uncertain the parser is about the sentence. From figure 1, we can see that the marginal probabilities were comparable to raw probability for most of the training instances. This might be the reason why the LAS of both the methods don't drift away much from each other.

3.1.4 Comparison to Hwa's paper

(Hwa, 2000) obtain 36% reduction in the number of brackets. In our experiment, random sampling takes approx. 10500 words to achieve an LAS of 75. Whereas, the uncertainity based methods take approx. 8k words to achieve the same score (23.8% less than random sampling). This might be because we are using the state-of-the-art dependency parser from Stanford and hence the random baseline is pretty hard to surpass. The evaluation metric used by (Hwa, 2000) was a bracketing metric whereas we used LAS. This may also be the reason for the difficulty with beating the random baseline.

4 Conclusion

In this assignment, we have used Active Learning to reduce the number of data points required to achieve the same performance in Dependency Parsing. We used three policies to actively sample data and compared it with the random baseline. Uncertainity sampling based on Margin probabilities seems to just outperform other methods.