CS388 Natural Language Processing
Homework 3: Active Learning for Statistical Parsing

Jianyu Huang

April 5, 2015

1 Introduction

We would like to get good performance using as few annotated examples as possible when learning PCFGs from data, since it is expensive to construct large training corpora like significant by human efforts. In this assignment we attempt to reduce annotation cost by adopting active learning for PCFG parsing, based on the experiment in (Hwa, 2000) [1].

Active learning, in particular, *sample selection*, requires that the learner itself select the examples for annotation from a large sample of initially unannotated data. 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. For statistical parsing, a training instance is a sentence and the annotation is a parse tree supplied by a linguistic expert.

The simplest approach to select training examples is *uncertainty sampling*. For statistical parsing, we use the following ways to measure the uncertainty in the automation of a sentence: sentence length, normalized probability of the top parse, and tree entropy using the top 10 PCFG parses.

Finally we compare the performance of active learning based on different uncertainty selection metrics with random selection of training examples and present corresponding learning curves.

2 Implementation

1. In the first step I create an initial training set, an "unlabeled" training pool for active learning, and a test set. To create the initial training set, extract the first 50 sentences from section 00. For the unlabeled training set, concatenate sections 01-03 of WSJ. So we get 4485 additional potential training sentences (approximately 100,000 words). For testing, use WSJ section 20. I write a *preprocessing.java* file to do the preprocessing by combine several files into one file, and count on the sentence number.

2. A simple command line interface to the `LexicalizedParser` is provided to support for active learning. I implement `ActiveLearn.java` in two steps.

   (a) **Initializion** I build the TreeBank objects for training set, "unlabeled" training pool and a test set from files with `maketreebank` function and transform TreeBank objects into memorybank objects because we can add/remove trees in memoryTreebank objects.

   (b) **Iterations** Since we can add/remove trees in memoryTreebank, we don't need to write out the next training bank and candidate pool to new files for the next generation, since we can simply add the selected trees into the training set and mark them as selected in the candidate pool so that they will not be selected in the following iterations. Though this way, we can iterate several times within one java execution. In each iteration, we will do the following parts.

   - Sort the trees in candidate pool by uncertainty selection metrics. If the selection metrics is random, we only need to shuffle the list. Otherwise, we calculate the string length, the normalized probability of the top parse, or the tree entropy using the top 10 PCFG parses as the `Comparator` for the `Collections.sort()` function.

- Add the random or most uncertain sentences by the sorted order whose total word count approximates 1500 from candidate pool into the training set.
- Train the parser with `trainFromTreebank` API based on the data in the training set.
- Evaluate the performance of PCFG parser with `EvaluateTreebank:testOnTreebank` API on the test set.
- Output the evaluation results for this iteration: iteration number, number of training words, PCFG F1 score.
- Generate the candidate pool for the next iteration by removing those selected trees in this iteration.

3. Collect enough data to plot a learning curve (F1 score versus number of training words) for each sample selection function.

3 Experiment

1. The results for random selection function and three uncertainty selection function (sentence length, normalized probability of the top parse, and tree entropy using the top 10 PCFG parses) are shown in Table 1. The corresponding learning curves are shown in Figure 1.

	random		strlen		prob		entropy	
	# training words	F1 score	# training words	F1 score	# training words	F1 score	# training words	F1 score
1	1514	47.78	1537	45.38	1514	45.67	1530	45.31
2	3017	52.23	3011	55.69	3012	49.92	3047	56.34
3	4511	55.19	4509	59.27	4518	59.23	4542	54.39
4	6022	57.32	6022	57.18	6008	63.92	6007	62.70
5	7513	57.47	7532	56.62	7511	64.93	7500	60.81
6	9021	57.74	9038	61.06	9008	66.28	9006	61.60
7	10524	64.93	10536	61.54	10515	67.46	10541	64.40
8	12020	65.10	12040	62.58	12027	68.64	12037	66.48
9	13507	66.72	13504	65.45	13511	70.05	13515	67.74
10	15017	69.47	15017	68.14	15003	70.57	15033	69.79
11	16545	68.05	16501	67.91	16511	71.11	16534	71.07
12	18004	70.29	18032	71.40	18014	71.78	18026	70.99
13	19518	71.27	19528	71.96	19530	72.46	19538	72.30
14	21020	71.77	21016	71.89	21016	72.83	21013	71.38
15	22523	71.97	22528	72.44	22512	73.18	22500	73.14
16	24003	72.68	24008	72.68	24010	73.62	24026	73.40
17	25509	72.66	25528	72.94	25509	73.89	25507	73.06
18	27021	72.99	27004	72.30	27032	73.40	27024	73.40
19	28512	74.90	28518	72.63	28509	74.49	28518	74.22
20	30023	75.04	30023	73.04	30035	73.85	30011	73.87

Table 1: Results of random selection function and three uncertainty selection function (sentence length, normalized probability of the top parse, and tree entropy using the top 10 PCFG parses)

2. In addition to the basic experiment, I also finish some extension experiments by changing the batch size of word count totals, changing the initial training set size, changing the Top K parses in Tree entropy, and changing the maximum iteration for each sampling function. The corresponding learning curves are shown in Figure 2.

   (a) Changing the batch size for word count totals.
   Set word count = 500.
   Observation: When the batch size is in finer granularity, active learning method has obvious better learning rate than random selection in the beginning of the learning curve.

   (b) Changing the initial training set size.
   There are 100 sentences from section 00 in the training set.
   Observation: When the initial training set is larger, active learning will have more confidence

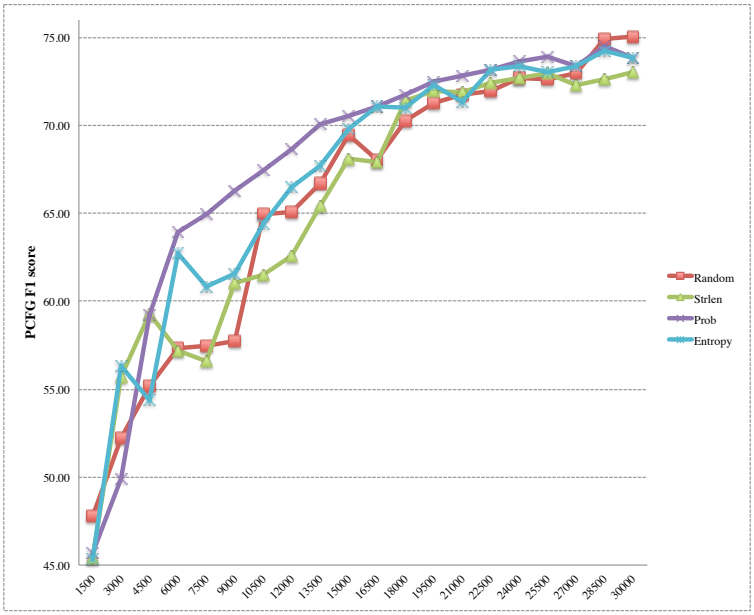


Figure 1: Learning curves for different selection functions

on some examples, so it can figure out the relatively uncertain examples more confidently and ask the expert to annotate it, thus it will learn faster.

(c) Changing the Top K parses in Tree entropy.
Compute the top 20 parses.
Observation: it will work slightly better to compute the top 20 parses for entropy than to compute 10 parses.

(d) Changing the maximum iteration for each sampling function.
Set iteration = 40.
Observation: In the end of the learning curve, both active learning and random selection converges to a constant, since the information it can learn from the candidate pool becomes saturated.

4 Discussion

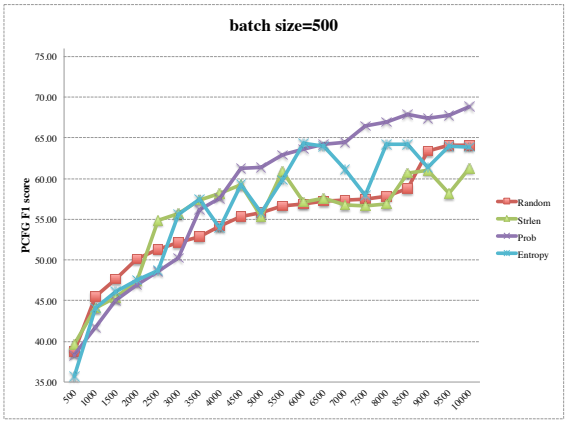
4.1 Comparison between Active Learning and Random Selection

1. Active learning methods perform better than random selection of training examples. There are several reasons:

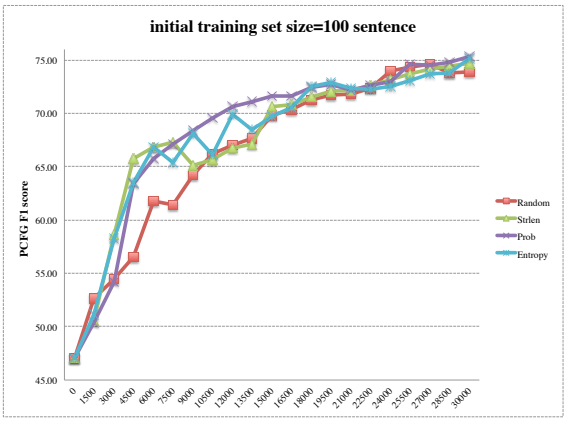
   - Active learning selects a small batch of the most useful examples for annotation and asks the expert to annotate them. By obtaining feedback on the cases in which it is most uncertain, it hopes to learn more than obtaining labels on random sentences in which its existing model is perhaps already quite confident in, and from which it would therefore not learn much.
   - Random selection just selects the random examples, so the "learning rate" is always the same. It cannot distinguish the examples which it can learn more and the examples it cannot learn much, since there are some examples it is quite confident in.

2. Active learning performs best in the beginning and middle part of the learning curve.

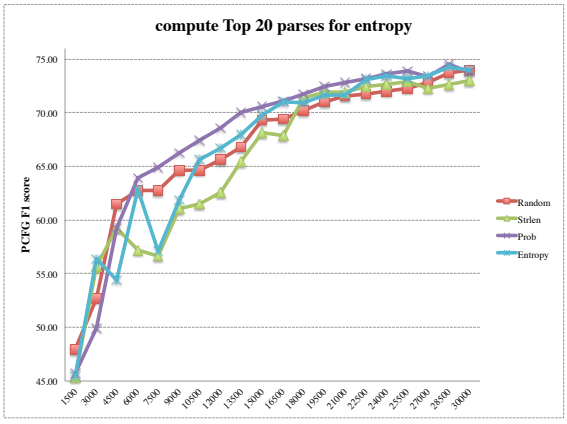
   - In the initial and the middle parts of the learning curve, active learning method selects the most useful examples for annotation by obtaining feedback on the cases in which it is most uncertain,



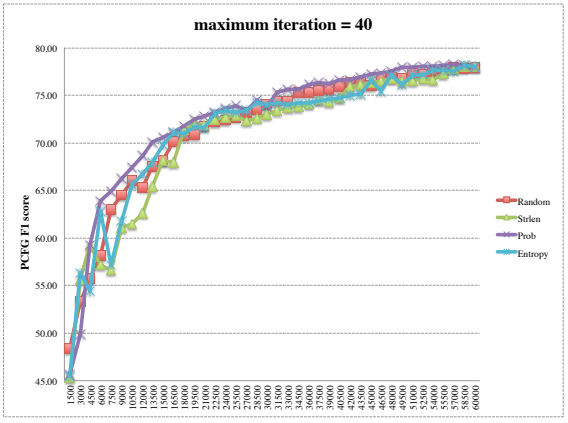
(a) Changing the batch size for word count totals



(b) Changing the initial training set size



(c) Changing the Top K parses in Tree entropy



(d) Changing the maximum iteration

Figure 2: Extension Experiments

so it will always learn faster than random selection. So the learning rate of active learning method is faster than random selection in the initialization part.

- However, in the final part of the learning curve, the learning curves of both active learning and random selection converges to a constant, since the information it can learn from the candidate pool becomes saturated. We notice that finally random selection may perform slightly better than active learning, because there is no discrimination in its selection of examples.

4.2 Comparison between Different Uncertainty Metrics

1. Measuring the uncertainty by normalized probability of the top parse works best.

   - We observe that all three uncertainty selection functions work well when the number of training words is small than 6000, but it may be unstable in some data points. Measuring the uncertainty by normalized probability of the top parse works better than by tree entropy using the top 10 PCFG parses, while tree entropy using the top 10 PCFG parses works better than sentence length.
   - Uncertainty measured by the normalized probability of the top parse improves the learning rate, since the normalized probability reflects the confidence of the model in the testing set most accurately.
   - Entropy reflects the disorder of the parses, so it will also reflects the uncertainty.
   - While the simplistic sentence length selection function is less helpful, its learning rate still improves slightly faster than the random selection baseline, since longer sentences are clearly more difficult for a human to annotate, thus more uncertain.

2. Compare results with [1].

   - In general, the performance of tree entropy using the top 10 PCFG parses and sentence length matches the result of [1], although my result is unstable in some data points. This is because the data distribution of unlabeled training set.
   - [1] doesn't measure the uncertainty with normalized probability of the top parse.

5 Conclusion

Through this assignment, We reduce annotation cost by adopting active learning for PCFG statistical parsing. It turns out that active learning performs better than random selection of training examples. Among the three uncertainty selection functions, normalized probability of the top parse works the best.

6 Apendix

To create the initial training set, extract the first 50 sentences from section 00. For the unlabeled training set, concatenate sections 01-03 of *WSJ*. So we get 4485 additional potential training sentences (approximately 100,000 words). For testing, use *WSJ* section 20. All the results are stored in "hw3.xlsx" file. Note that for single-word sentences, I don't normalize the probability by sentence minus 1, since it will lead to infinite.

7 Reference

[1] Rebecca Hwa. 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*, pages 45–52. Association for Computational Linguistics, 2000.