1 Active Learning

A common scenario in natural language processing is having far more data than annotations, since data collection is easy and annotation is hard. Two common techniques for making the best of this imbalance are semi-supervised learning and active learning. This report focuses on active learning; in this particular scenario, 'active learning' simply means using prior annotations to select which unlabeled data would be most useful to annotate.

We find that active learning, particularly using the 'tree entropy' and 'length' metrics, boosts the usefulness of additional annotations. Active learning is most useful at the very beginning of the learning process. At this point, the difference between a random sample of the unlabeled data (potential annotations) and a heuristically selected sample is the greatest. However, we are using a limited pool of unlabeled data, and iterate towards using a higher proportion of it labeled.

This is unrealistic for most real parsing applications; we will rarely annotate all of the data we have on hand. As we use more and more of the unlabeled data, the difference decreases. When we reach the final batch of 60 sentences, the selection heuristic must choose the same sentences as the random function would, since that's all there is left.

2 Comparison between heuristics

We used three heuristics to select unlabeled sentences to 'annotate' (of course, we did not annotate any sentences, we merely allowed ourselves to see the pre-existing annotations for certain sentences). The goal is to minimize annotation costs; we approximate that by trying to minimize the number of sentences we train on.

It's debatable whether annotation costs are most directly related to token counts. Particularly if the sentences are preprocessed in some naive way (as the Penn treebank was), there might be other factors more important than token counts—aspects of verifying and completing a parse that make some sentence more costly than another.

1. **Tree entropy.** By measuring the distribution of trees and the confidence of the parser, we can select sentences that the parser is clueless about. If the parser thinks each of 20 trees are nearly equally probable, tree entropy will be very high; if it vastly prefers a single parse to all the others, tree entropy will be low. We select those sentences where the parser produces high entropy, because those will presumably be most instructive.

   While Hwa (2000) considers all possible parses, I consider only the 20 best. In most cases, particularly when seeking high entropy, there will be little difference. The intuition is that with low entropy trees, the majority of the probability space will be consumed by the top tree, maybe the top few. But in high entropy cases, the entropy of tens of trees equally sharing that space will be propotional to the entropy if we consider hundreds or thousands of trees sharing that space.

2. **Top parse.** The parser assigns a probability to the highest parse in a sentence, relative to other potential parses. This is not directly related to such probabilities measured for other sentences, but by normalizing for sentence length, we can facilitate a crude comparison.

3. **Length.** Longer sentences will have more complex parses, which are presumably more instructive.

We compare these to an incremental learner that randomly selects a number of new sentences to annotate from the unlabeled pool.

Fig. 2 below shows that the tree entropy and length metrics are most useful in selecting additional sentences to 'label' and train on, but that the top parse heuristic is no more useful than the random baseline.[1]

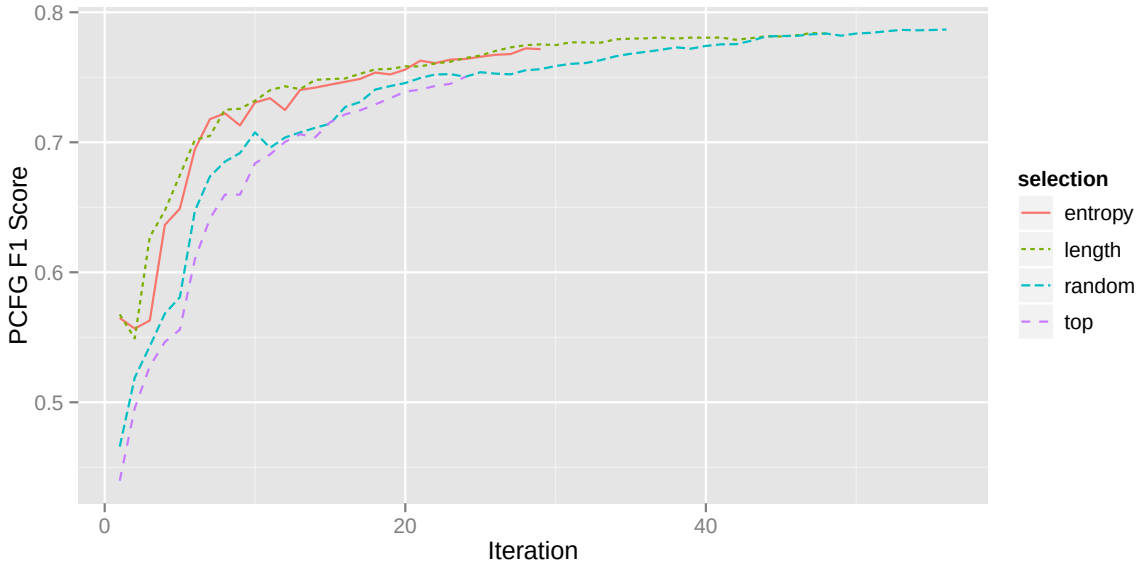


Figure 1: Results by iteration.

As expected, the random selection function is the weakest at the beginning. The tree entropy and sentence length measures are best, and produce similar results. The active learning advantage is clear even at the very beginning, but results converge quickly as we use up the unlabeled candidate pool.

Interestingly, if we reverse the selection ordering (we pick from the trees with the lowest entropy, instead) the difference is more pronounced. Of course, the entropy measure is disadvantageous in this case, while the random baseline is about the same as before.

[1]Some data points are missing for the slower heuristics at high numbers of iterations; this is due to my condor jobs stalling due to insufficient disk space. I remedied the issue by rewriting the Stanford source to reduce forced logging, see the **Instructions** section, but did not have enough time to rerun all the trials.

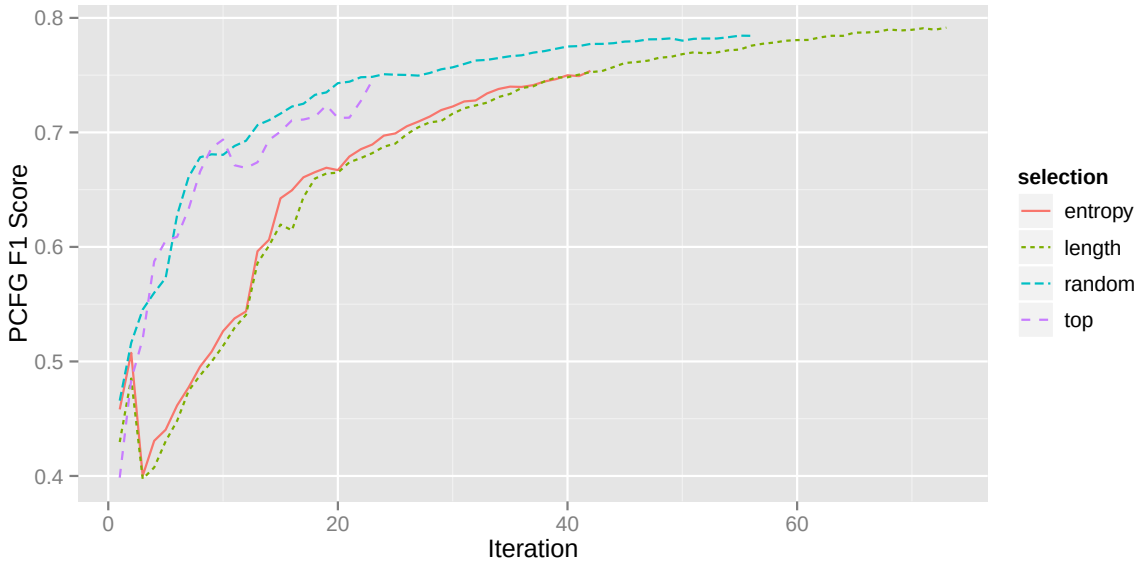


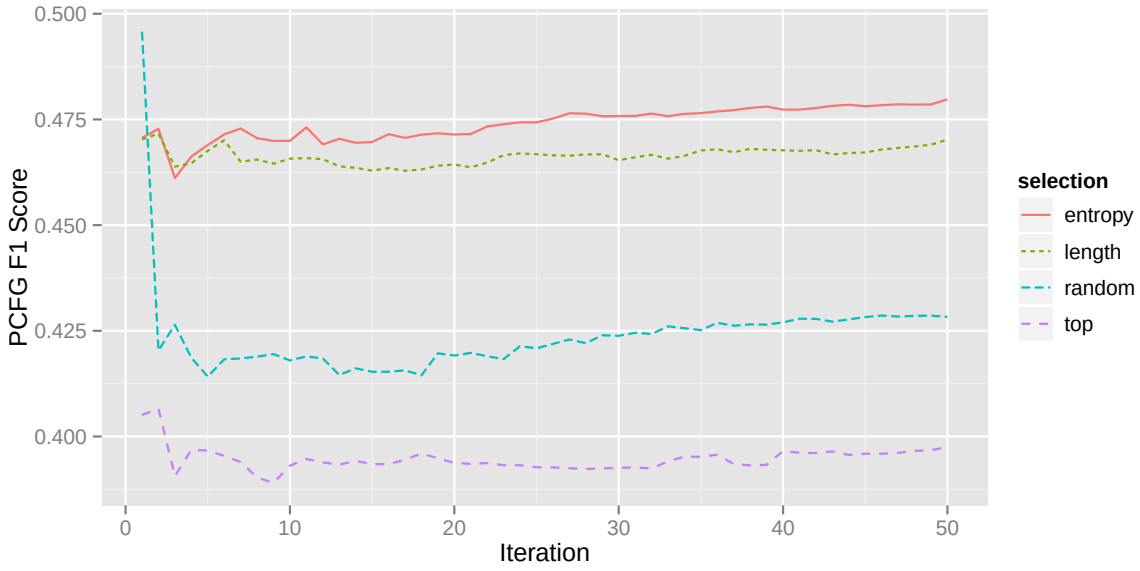
Figure 2: Results by iteration, reversed.

The parser often exaggerated the results of the very first iteration; I'm not sure what causes this idiosyncrasy, but we can consider that to be a burn-in period, whether it's due to the internals of the Stanford Parser, the JVM, or a bug in my code.

3 Semisupervised

I also tried training the parser on its own parses; I had the parser re-annotate the unlabeled sentences and then learn from those. This was not at all what the parser was built to do, but surprisingly, this was marginally helpful. I used the same selection functions, but sometimes in different order; for example, I had it learn from the sentences with the lowest tree entropy, i.e., those sentences for which it was most confident in the best parse.

Active learning did not play a significant role here; the random selection function was just as helpful as the tree entropy function. Nevertheless, I thought that the parser's ability to improve purely by self-training was interesting.



4 Conclusion

Hwa (2000) differs in that for her, the advantage of active learning is not as visible at the very beginning, but mostly in between the first few iterations / data points and the convergence point. This could be due to a variety of factors: starting out with a different initial training set (or larger set), using a different corpus as the candidate / unlabeled set (smaller?), or a less lucky test set. Hwa averages her results over multiple runs, which I do not perform in this duplication of her experiment.

Ideas for improvements

- In a truly active learning situation, we could construct the parser to gauge the most useful-to-annotate subtrees; our parser could determine what part of a tree it is highly confident about, and then present those parts of the tree to the annotator cemented together, leaving only the unconfident structure for the annotator to fill in.
- I expect that if we used WSJ sections 1-19 instead of just 1-3, the active learning mechanism would maintain an advantage over the baseline longer. But because every unlabeled / candidate sentence must be reparsed after each iteration, this would take much more time.
- It would be interesting to see how much help an unsupervised parser is in selecting the very first set of sentences to train on. If the overall goal is to minimize annotation labor, unsupervised parsing (or chuning) seems like it serve as both a useful starting point as well as when we are still in the first iterations, though I imagine its effect would diminish quickly as our training corpus grows.

5 Instructions

- All code and the full SBT project is available at `https://github.com/chbrown/nlp`.
- Commands used to run the code can be found in `hw03.README`.

- I made considerable changes to the Stanford Parser source code because it was emitting a huge number of logging messages and does not allow setting the log level. I was able to silence some of the messages via a custom TreebankLangParserParams class with pw() method overrides, but there were still far too many System.err's hard-coded into the Stanford Parser source. With all the jars that the library requires, I'm left with about 80 MB of free disk space on the CS cluster (which isn't too surprising considering I only have 256 MB to begin with). The Stanford parser quickly burned through that legroom with all the forcibly logged parse trees and parameters displays. So I replaced many of the forced standard error calls with log4j rootLogger.trace calls, which brought the library's output down to a much more reasonable level.
- These changes are available at `https://github.com/chbrown/stanford-parser`, and they are part of the `https://github.com/chbrown/nlp` repository, too.

References

[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*, 2000.

[2] Dan Klein and Christopher D. Manning. "Accurate Unlexicalized Parsing." *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 2003.

[3] Dan Klein and Christopher D. Manning. "Fast Exact Inference with a Factored Model for Natural Language Parsing." In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, Cambridge, MA: MIT Press, 2003.