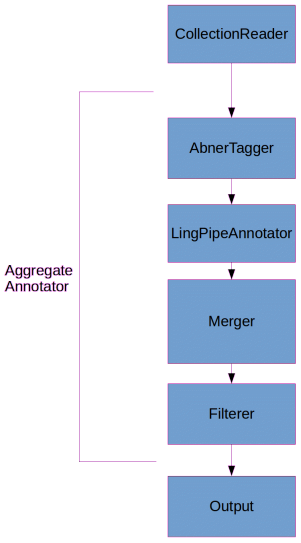
11-791: Homework 2

Maya Tydykov

Monday, October 6, 2014

1 System Pipeline

My system first reads an input file and an optional gold standard file. It then consults the aggregate analysis engine to determine which annotators (and which order) it should use to process the input. The system can currently use two annotators - LingPipeAnnotator and AbnerTagger, which are based on the LingPipe and ABNER tools, respectively. The system can then use the extraction results of the tools to train a logistic regression classifier (also from LingPipe) for each annotation collected from the previous annotators based on a few simple features. When the logistic regression classifier has been trained, the system can use the following pipeline to tag an input document:



The current features used to train the logistic regression classifier are:

- For each annotator used in the system, whether or not the annotator voted for the annotation.
- Whether or not all annotations that overlapped with the given annotation had perfect overlap.
- Whether or not any "undesirable" terms matched any of the tokens in the mention. Most of the undesirable terms came from a list of non-peptide hormones from Wikipedia and were inspired by the Appendix in Homework 1.

After the merging step I added another filter that uses several rules (inspired by the Appendix section of Homework 1) to eliminate mentions that were very unlikely to be true positives. These rules include eliminating mentions that consisted only of an undesirable term and also eliminating mentions that were encapsulated by parentheses or were only one character in size.

The final mentions are produced in this step and sent to the CAS Consumer to be written to the output file.

I tried to design my system based on ideas borrowed from several design patterns in order to keep it as flexible and maintainable as possible. The design follows the ideas of GERP in that annotations are first generated by two different annotators (LingPipe and ABNER); some evidence (in the form of features) is presented to the merger, and although there is no ranking step, there is a pruning step at the end which filters out bad left-over annotations. I also tried to design my system such that it kept to a high cohesion pattern, where each class has a very specific responsibility in the overall pipeline. For example, there is a specific class that gathers annotations from each basic type of annotator (LingPipeAnnotator and AbnerTagger); a class (FeatureExtractor) responsible for extracting features and setting each GeneMention's feature array; a class responsible for training (Trainer); a class responsible for the classification/merging (Merger), a class for filtering (Filterer), and a scoring class (Scorer). Thus, although there are currently not many features, rules or basic annotators in the system, it should be easily extensible so that one can conceivably experiment with adding many different kinds of rules, features, and annotators without making any significant changes to the code.

2 Discussion and experimental results

I first wanted to incorporate annotator confidence from each annotator into the classification and merging process. I tried using the ConfidenceChunker from LingPipe, but was not able to get good results with this method - final recall was low and precision was low as well. I finally reverted back to using the Chunker which gives only the best chunk, although this method gave me a confidence score of 0 for each chunk (using the ConfidenceChunker and setting it to return only 1 item did not do as well as using the Chunker). Additionally, ABNER did not seem to have any built-in way of getting a confidence value. Thus, I was not able to use confidence as I had originally wanted; instead, I decided to have the logistic regression classifier serve as an intermediate voter of sorts based on the simple, above-mentioned features.

The best-performing results on the same training data as was used in homework 1 in terms of F-Score were achieved with the current pipeline and the set of features described above. The results were as follows: Recall was at 83.64%, precision was at 75.73%, and F1-score was at 79.49%.

Interestingly, these are somewhat lower than the results I was able to achieve when using only LingPipe to annotate the data. The recall may be lower because my extra filtering (both from the logistic regression classifier and the rule-based filter) may be overly strict, and the precision may be lower because of the addition of ABNER into the pipeline. Nevertheless, it is possible that this pipeline may perform well on other, previously unseen data, particularly if it were extended to include more annotators and/or more features.