Report HW 2 11-791 David Klaper (dklaper)

1. Architecture

The pipeline consists of 3 steps. First, the collection reader reads the file as in the previous assignment.

Then, the aggregate analysis engine contains all other steps. It annotates each sentence with 3 unique annotators. The lingpipe NER and my custom trained CRF from the previous assignment (which is implemented as an AAE as well (for tokenization and PoS-tagging) so I have nested AAEs), as well as, Abner which I newly wrapped for this assignment. I also tried using the baseline as an additional annotator but it would not help performance. After the annotation of all three the new AnnotationMerger, filters out annotations based on a simple heuristic: It keeps all annotations that were reported at least by 2 annotators, as well as all annotations reported by the preferred annotator (which is a parameter by default set to the Lingpipe NER Annotator.) but only if it does not overlap with any annotations that were annotated by the two other annotators (implemented with an interval tree). During development the aggregate engine also contained the evaluationConsumer that printed performance values to stdout.

Finally,the CasConsumer writes the result as a file as in previous assignments.

To use the framework as efficiently as possible I loaded models as resources and even provided a SharedResource interface and implementation for the evaluationConsumer that reads the gold standard data. Also the Abner model can simply be modified by changing the parameter via reflection (works for internal models not external). While reflection is not a pattern per se it is very useful for allowing parameters to change complex behavior of libraries. By default it will use the Biocreative corpus, since the performance is better due to identical annotation guidelines.

2. Type system Details

My type system makes use of the provided Annotation and Token type in the deeis_types.xml by inheriting and extending from it. I use the CasProcessorIDeffectively by setting it to the qualified class name (so not just a manually generated string) so there will never be ambiguity even if the annotator is used in a different project with the same type system. I set the confidence values in all my annotators but the values from Lingpipe and my CRF are not nomralized to be in the range [0,1]. Nevertheless they could be transformed and used. I just did not need them for my simple aggregation that mainly aimed at not providing overlapping annotations of which one is certainly wrong.

3. Performance

I tested my annotator that is not trained on the sample data on the given sample data. Unfortunately it's slightly worse than the Lingpipe NER alone but the recall is quite a bit higher even if precision is a bit sacrificed:

Precision: 74.26
Recall: 85.15
F1: 79.33