11791 Homework 2 Report

Fei Xia
Language Technology Institute
feixia@andrew.cmu.edu

1 IIS Design & Architecture

The system is carefully designed and has several improvements since Homework 1 – It is concise, has a clear structure, works well for the current task and has the ability to scale to multiple different NER annotators.

In general, all paths are followed the latest instructions and all parameters are not hard-coded. Developers can easily change the component combination in aggregate analysis engine and do test (in terms of efficiency and efficacy) to find out which component should be used.

1.1 Collection Reader

The job of collection reader is to load document text to JCAS. The collection reader read the text line by line. In addition, the id and the actual sentence are separated during reading collection – We have a Sentence type (see the Type System section) to store this information. Then later in the annotators, we can take the information from the Sentence type.

1.2 Type System

The type system is designed in an inherited way. Please refer to Fig. 1 for the design.

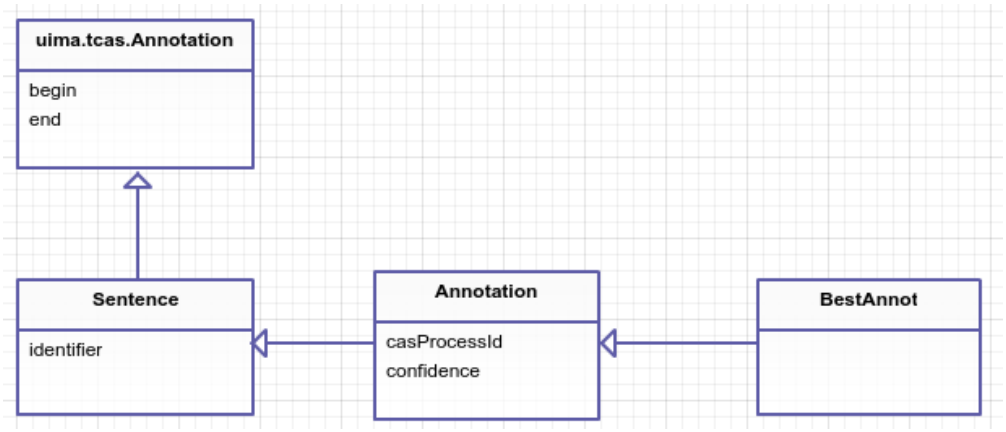

Figure 1: Type System

The Sentence type stores the identifier and the begin index as well as the end index of the actual sentence for each line. So later other NER annotators can directly extract relative information from the Sentence type. The Annotation type is used to store the results from different NER

annotations and has two additional fields – the confidence and casProcessorId. The confidence stores the confidence score of the current annotation and the casProcessorId stores the annotator id that makes this annotation. We further have a BestAnnot type. This type stores the annotation results that are combined from multiple annotators and can be used directly by the consumer to output the results later.

1.3 Annotator

I have designed four different NER annotators and one voting annotator. NER annotators stores their annotation results in Annotation type and voting annotator stores its annotation in BestAnnot type. The four different annotators are: LPConfNERAnnotator, LPDictExactNERAnnotator, LPDictAppoxNERAnnotator, AbnerNERAnnotator. Let me explain them one by one.

- LPConfNERAnnotator. It uses LingPipe statistical method to do the Gene Name Recognition and needs to load a pre-trained model. Other that that, I have some other parameters, see Algorithm Design & Performance section for more details.
- LPDictExactNERAnnotator. It uses LingPipe dictionary based method to do the NER task and do exact match. It needs to load a dictionary, so I built a dictionary based on the database from HGNC (HUGO Gene Nomenclature Committee)[1]. To get more gene names and combinations, I also added the gold results from sample.out.
- LPDictAppoxNERAnnotator. It uses LingPipe dictionary based method to do the NER task and do the approximate match. This basically used a trie tree data structure and limit the edit distance to a specific value.
- AbnerNERAnnotator. It uses ABNER and the model trained on BIOCREATIVE to do the annotation. The output of ABNER tagger is an annotated string in BIO principle, so we need to parse the results to find out the corresponding begin index and end index.

I have an aggregate analysis engine to combine those NER annotators and the final voting annotator. The order of NER annotators in my system doesn't matter, but the voting annotator should be put in the final position in AAE. Through testing, I found the LPDictAppoxNERAnnotator is too slow, so I will take it out in my submission.

1.4 Consumer

The job of consumer is to take the annotation from BestAnnot and output it to a file. I also implemented an evaluation component, if another optional parameter – the gold standard data file path – is provided, it will also do the evaluation.

1.5 CPE Descriptor

The CPE descriptor is put in the *src/main/resources* directory. Basically, it takes three XML file – FileSystemCollectionReader.xml, AAE.xml and AnnotationConsumer.xml. Once the system is constructed, we need to focus on the most important part, i.e. the analysis engines. For different combination of analysis engines, we only need to modify the AAE.xml, and then run the CPE descriptor directly. So it is really convenient.

[1] http://www.genenames.org/cgi-bin/download

2 Algorithm Design & Performance

In this part, I will mainly talk about the different annotators and some comparison. Note that all the results are based on the given *sample.in* and *sample.out* data. Let's look at them separately first, then go to their combinations.

2.1 LPConfNERAnnotator

As described above, I have other two parameters besides the pre-trained model – number of best matches in a sentence and the confidence threshold. The first one is how many annotations will be output for a given sentence. In my setting, this is set to 10. The latter one is used to filter out the poor annotation results. In my setting, this is set to 0.65. It will only keep the annotations whose confidence scores are more than 0.65. Please refer to Table 1 for results.

Table 1: LingPipe Confidence NER Annotator

Precision	Recall	F-1 Score	Time (s)
0.8295	0.8010	0.8150	4.58

2.2 LPDictExactAnnotator

As I have built the dictionary from HGNC and added all the gold results of *sample.out* to the dictionary, this annotator performs extremely well in recall on *sample.in*. But note, this doesn't mean this annotator is a good annotator in general! Presumably, it will get lower performance in other datasets, but should be better than the situation that we don't add the gold results from *sample.out* to the dictionary. Please refer to Table 2 for results.

Table 2: LingPipe Dictionary Exact Match NER Annotator

Precision	Recall	F-1 Score	Time (s)
0.8057	0.9901	0.8884	2.09

2.3 AbnerNERAnnotator

The ABNER annotator uses the model trained on BIOCREATIVE. The tagger of ABNER outputs a string that is in BIO principle so we have to parse it. My basic idea of parsing it is to searching the tagged Gene Name Entity from a specific index and the index is constantly updated. More details of the codes can be found in AbnerNERAnnotator.java. Please refer to Table 3 for results.

Table 3: ABNER NER Annotator

Precision	Recall	F-1 Score	Time (s)
0.7380	0.6791	0.7073	77.24

2.4 Annotator Combination and Voting

Since we have several components, we can think about how to combine them. Let's take an example – use LPConfNERAnnotator and ABNER. We have two ways: (1) LPConf AND

ABNER, i.e. only select the annotations that both agree (2) LPConf OR ABNER, i.e. select all the annotations from LPConf or ABNER. Let's look at the Table 4 for the results.

Table 4: Example of LPConf and ABNER

	LPConf only	ABNER only	LPConf AND ABNER	LPConf OR ABNER
Precision	0.8265	0.7380	0.9249	0.7085
Recall	0.8010	0.6791	0.6207	0.8907
F-1 Score	0.8150	0.7073	0.7429	0.7892
Time (s)	4.58	77.24	84.09	83.79

For "LPConf AND ABNER", we can easily notice that it gets higher precision but lower recall than LPConf only and ABNER only. This meets our intuition. Since both annotators have to agree with the annotation, the condition of generating an annotation is more rigorous, but we may miss some correct annotations, so we can get higher precision but lower recall.

For "LPConf OR ABNER", we can see an opposite results, i.e. it gets lower precision but higher recall compared with LPConf only and ABNER only. The reason is that we accept any annotation generated by LPConf and ABNER, thus we can get more correct annotations, but at the same time, we introduce many incorrect annotations. Thus the precision is lower and recall is higher.

Other observations are that: ABNER is much slower than LPConf. Two annotators are slower than one annotator. This obviously makes sense.

Finally, I think a better way to organize the three NER annotators is to let them voting. If an annotation gets two or more than two votes, it will be accepted. Please refer to Table 5 for the results.

Table 5: Voting

Precision	Recall	F-1 Score	Time (s)
0.9137	0.9149	0.9143	84.69

This gives us the best F-1 score all through this report. I have to say that the absolute value of the F-1 score may not be able to indicate the performance on test data (since I added the gold gene names to the dictionary). However, comparatively speaking, the voting can get the highest F-1 score, which means the voting strategy works pretty well.

So the three NER annotators (LPConfNERAnnotator, LPDictExactAnnotator, AbnerNERAnnotator) and a voting strategy will be my final submission. Let's see how it works in unseen test set!

3 Others

I can successfully run the script following the instructions here:
https://github.com/amaiberg/software-engineering-preliminary/tree/master/grading_hw1_2
If you cannot run my code successfully, please contact me via feixia@cs.cmu.edu. Thank you.