11791 Homework 1 Report

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

1 IIS Design & Architecture

To finish the NER task, I load a pre-trained model and then use LingPipe to recognize the gene name entity. Thus, there isn't much fancy UML design in my homework. However, my system is carefully designed to give more flexibility, so that the developer and maintainer can easily change the codes and components. For example, all parameters like file path, machine learning model parameters are not hard-coded; developers can easily change the component combination in an aggregate analysis engine descriptor to find out a method with high performance, etc.

1.1 Collection Reader

The collection reader java file can be found in *src/main/java/edu/cmu/lti/fei/reader* directory and its descriptor can be found in *src/main/resources/descriptors*. The job of collection reader is to load the document text to JCAS. The input path parameter is not hard-wired and you can change this parameter very conveniently in collection reader descriptor or CPE descriptor.

1.2 Type System

The type system is designed in a inherited way and the descriptor can be found in *src/main/resources/descriptors* directory. Please refer to Fig. 1 for the design.

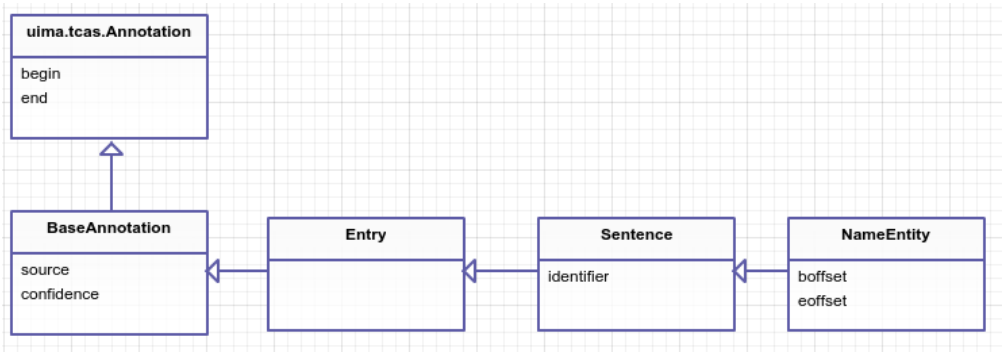

Figure 1: Type System

The BaseAnnotation inherits from the *uima.tcas.Annotation* and has additional *source* and *confidence*. These fields can be used later to store the original source and the confidence score of the annotation. Entry is defined as one line in the document file (i.e. an identifier followed by a sentence) and directly inherits from BaseAnnotation. As for the Sentence type, it annotates the sentence in an entry and memorize the sentence's identifier. Our final goal is the NameEntity type,

it gives the final annotation of the gene name entity and the non-whitespace offset with respect to a sentence.

This design gives us clear layers of the system. If we want to change something (for example, some parsing rules), we don't need to go to the Collection Reader. Instead, according to our type systems, we can go directly to the annotator part and make the change.

1.3 Annotator

The annotators can be found in *src/main/java/edu/cmu/lti/fei/annotator* directory and their descriptors can be found in *src/main/resources/descriptors* directory.

According to the type system, I have a bunch of annotators. EntryAnnotator is used to annotate an entry, i.e. the line in the file; SentenceAnnotator is used to annotate the sentence, which is a part of the entry; BSNameEntityAnnotator, LPNameEntityAnnotator, LPConfNameEntityAnnotator are all used to annotate name entities. We will talk more about them in **Algorithm Design** section. Besides, we have EvaluationAnnotator to do the evaluation.

1.4 Consumer

The consumer can be found in *src/main/java/edu/cmu/lti/fei/consumer* directory and its descriptor can be found in *src/main/resources/descriptors* directory. The consumer will get results from the last annotator and write results to file.

1.5 CPE Descriptor

The CPE descriptor is put in the *src/main/resources* directory. It is built using the collection reader descriptor, the aggregate analysis engine descriptor (called *AAE.xml* in my project) and the consumer descriptor.

2 Documentation, comments and coding style

This file is the report. For other related stuff, please refer to the submitted source code and Javadoc.

3 Algorithm Design & Performance

3.1 Baseline

Our baseline is the provided method based on Stanford NLP. In my pipeline, it is implemented in the BSNameEntityAnnotator.java file. The performance is shown in Table 1. We can easily see that the results are not good.

Table 1: Baseline Performance

Precision	Recall	F-1 Score	Time (s)
0.1025	0.5467	0.1727	15.112

3.2 Statistical Model Based

In this section, we mainly explore the statistical model based method. We take the advantage of the LingPipe tool and load a pre-trained model on another corpus – GENETAG[1], then do gene name entity annotation.

3.2.1 LingPipe Direct NER

The direct NER directly extracts gene names from given text according to the model, without giving confidence scores. This method is implemented in LPNameEntityAnnotator.java file. Please refer to Table 2 for the performance. It can be seen that the performance is significantly improved.

Table 2: LingPipe Direct NER Performance

Precision	Recall	F-1 Score	Time (s)
0.7685	0.8488	0.8066	3.351

3.2.2 LingPipe NER with Confidence Score

This not only gives the extracted gene names but also gives the confidence score of the annotation. We need to specify two parameters – the MaxBestChunks and the Threshold. The MaxBestChunks defines how many best annotations you want to get from this sentence, and the Threshold plays a role to filter out those false positive gene names in my codes. These parameters are not hard-coded, either.

Here comes another question: How do we set these two parameters? We are given a sample.in and sample.out file. We can use methods similar as cross validation to find the right parameters. Here I set MaxBestChunks=10 and Threshold=0.65. Please refer Table 3.

Table 3: LingPipe NER with Confidence Score

Precision	Recall	F-1 Score	Time (s)
0.8295	0.8010	0.8150	4.127

3.3 Analysis

Table 4: Performance Comparison

	Baseline	DirectNER	ConfidenceNER
Precision	0.1025	0.7685	0.8295
Recall	0.5467	0.8488	0.8010
F1-Score	0.1727	0.8066	0.8150
Time (s)	15.112	3.351	4.127

Let's first put all the results in Table 4. We get very good results in DirectNER and ConfidenceNER. Additionally, we get more flexibility in ConfidenceNER. We can use the Threshold

[1]http://www.biomedcentral.com/1471-2105/6/S1/S3

parameter to control the precision and recall. If we give a high value to Threshold, we get high precision and relatively low recall; if we give a low value to Threshold, we get relatively low precision but high recall.

I also checked some False positive and False Negative examples in my output file (generated by LingPipe method). It is quite interesting and in some cases, it is really really difficult to get it right. For example, One False Positive example is "P09261155A0833|84 109|insulin receptor substrates 1", the corresponding correct one should be "P09261155A0833|84 116|insulin receptor substrates 1, 2, and 3". Such things are very common in my results – We can almost get it right! I think that one future work could be to explore more in the linguistic rules to eliminate this kind of error.

4 Others

The program can run very well in my computer. If you cannot run my program, please contact me via feixia@cs.cmu.edu. Thank you.