11-791 Design and Engineering of Intelligent Information System Fall 2012 Homework 1

Yibin Lin (yibinl@andrew.cmu.edu)

• Hand out: October 2.
• Turn in: October 17.

1. **Overview**
   Most java files in this project reside in edu.cmu.lti.se.ner package. There are two Analysis engines in the system. They are GeneMentionAnnotator and POSBasedAnnotator (unused). There is one collection reader, InputFileCollectionReader. It reads a file and creates a CAS for each line in it. Since it does handle any specific details of the gene mention tagging task, the InputFileCollectionReader is in the package of edu.cmu,lti.se. NERWriterCasConsumer is the CAS consumer for this project.

   StringUtils is in edu.cmu.lti.se.ner.util package. It contains shared string operations.

   SimpleRunCPE is still the main class of this project, which resides in the default package.

   All cpe/annotator/collection reader/type system descriptor XML files are in the src/main/resource folder.

2. **Type System**
   The main output type system in this homework is NERAnnotation, which stands for the final annotation of the gene mentions. It contains geneName, for the string of the gene mention, and outputId, which is the sentence identifier.
   There is also a intermediate type system called POSAnnotation. It is used by POSBasedAnnoator to record noun phrases and the weights of the orthography features. (Please see 5 b).)

3. **General Data Flow of the system:**

   There is only one CollectionReader, one Annotator and one CasConsumer working in the final submission. Therefore the system has the simplest UIMA CPE data flow it can have.

4. **Machine Learning Techniques in this Homework**

   Hidden Markov Model is used to predict gene mentions in GeneMentionAnnotator. Specifically, it is done by using LingPipe library[1]. The model is also from the complete LingPipe Package. It is called "ne-en-bio-genetag.HmmChunker". Most of the work is done by calling the chunk method in the LingPipe Library. The model is located in resources/model folder.

5. **Other Techniques and Rules**

   a) Filtering of LingPipe results according to basic orthography.
      According to the orthographic features listed in [2] and some common sense, the single lower case letter that are matched in LingPipe was deleted.

b) POS tag + orthography features
The POSBasedAnnotator used 9 orthography features, given that the word/compound has already been classified as noun phrase (NN) by StanfordCoreNLP. The 9 features are also mentioned in [2]. It was used after the GeneMentionAnnotator.
1) whether the NN contains digits
2) whether the NN contains digits and capitalized letters
3) whether the NN contains digits and letters
4) whether the NN contains hyphens
5) whether the NN contains open squares ([)
6) whether the NN contains backslashs
7) whether the NN contains colons
8) whether the NN contains semicolons
9) whether the NN starts with capitalized letters
Regular expressions are used to detect these features. Different weights were given for each features. If the sum of all weights is greater than a threshold, then it will be added into our NERAnnotation.
Table 1 is the recall plot for different thresholds.

threshold weight	precision	recall	f1
2	0.5325	0.8492	0.6545
3	0.567	0.8491	0.6799
4	0.7031	0.8488	0.76
5	0.7704	0.8488	0.8077
999	0.7704	0.8488	0.8077

Table 1

From the table we can see that if we lower the threshold, the recall will go up as we add more NERAnnotation's into the Jcas object. On the other hand, precision goes down for exactly the same reason. Threshold 999 means that no NERAnnotation's will be added through POSBasedAnnotator. In other words, all the results in threshold 999 condition come from GeneMentionAnnotator, which uses LingPipe.

It may be very interesting if POSBasedAnnotator can serve as a reverse Annotator (a “prunner”) of the NERAnnotations after the GeneMentionAnnotator. Due to time constrain, this hasn't been implemented.

Because of the low F1 measure, POSBasedAnnotator is not included in the final version of CpeDescriptor.xml, so only GeneMentionAnnotaor is used in the final submission. This is the best performing system so far. (Table 2)

	precision	recall	f1
best performing system	0.7704	0.8488	0.8077

Table 2

6. Design Patterns Used

There are not much design patterns used in the project. The most obvious pattern that I can think of is edu.cmu.lti.se.ner.util.StringUtils class. It groups responsibilities for easier maintenance.

However, the project uses UIMA framework, which uses a lot of good design patterns. For example, There is a FSGenerator in the every type that is generated by the Type System. This is an example of the factory pattern.

Reference

[1]. Alias-i. 2008. LingPipe 4.1.0. http://alias-i.com/lingpipe (accessed October 17, 2012)
[2]. Mitsumori T, et al. Gene/protein name recognition based on support vector machine using dictionary as features. BMC Bioinformatics. 2005;6:S8.