FINAL REPORT

Name: Ningna Wang
AndrewID: ningnaw
Date: 10/08/2014

1. Design Aspect

1.1 Architecture

For homework 1, I use LingPipe instead of Stanford NLP since LingPipe has better performance. However, only use one analysis engine to extract gene mention may not precise. So in homework 2, I decided to use an aggregated analysis engine (AAE), which combine LingPipe and Abner together. The reason that I did not use LingPipe and Stanford NLP is that Stanford NLP only extracts norms in articles, which will precision. Figure1 shows the class diagram of my project.

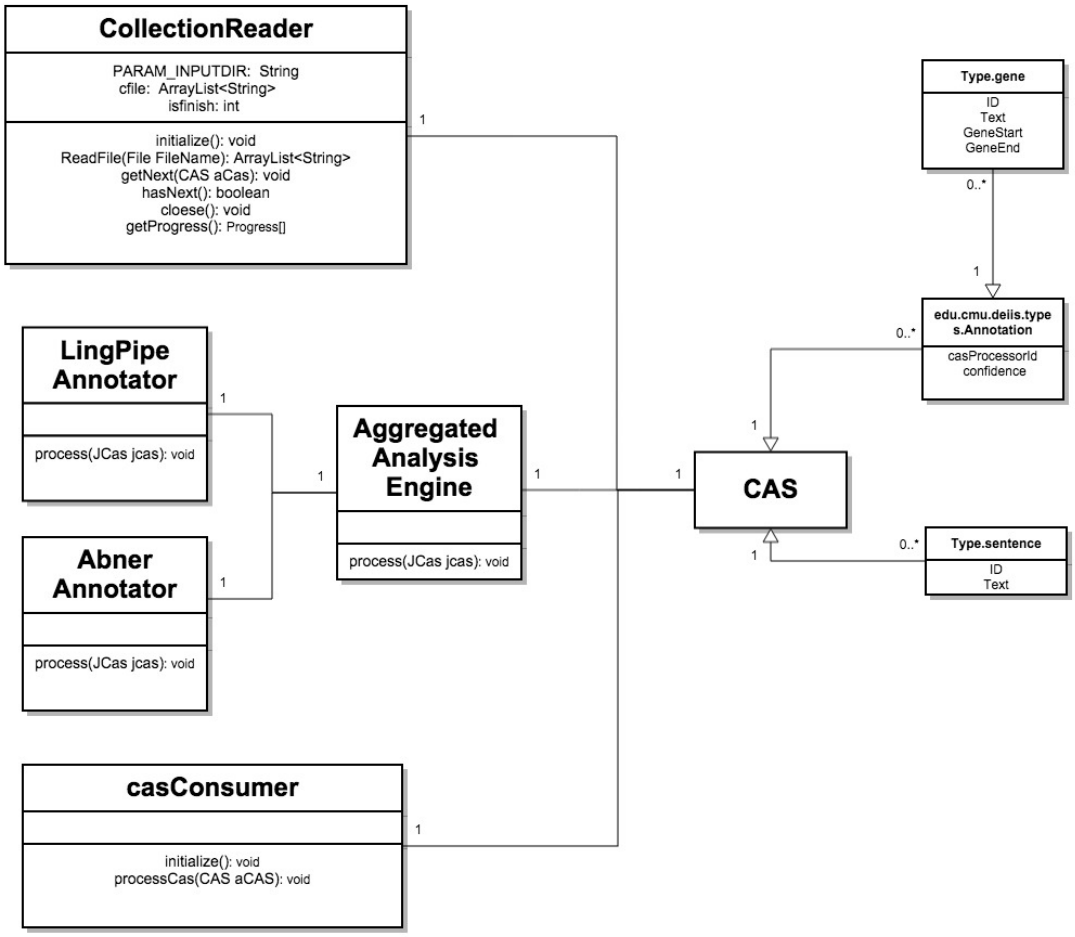

Figure1 Class Diagram

1.2 Annotation

Type System:

a. Type.sentence:
   a) Inherited from type "uima.tcas.Annotation"
   b) Features: ID—Store ID information of each sentence in document
      Text—Store text information of each sentence in document

b. Type.gene:
   a) Inherited from type "***edu.cmu.deiis.types.Annotation***"
   b) Features: ID—Store ID information of each gene extracted by AAE
      Text—Store text information of each gene extracted by AAE
      GeneStart—Store the number of non-whitespace character in the sentence preceding the first character of the mentioned gene
      GeneEnd—the number of non-whitespace characters in the sentence preceding the last character of the mentioned gene

1.3 Annotator:

1.3.1 Descriptor:

1. *CollectionReaderDescriptor*
   Implemented by ***CollectionReader*** java class file, and the construction parameter is ***InputDocument.***

2. *hw2-ningnaw-aae(aggregated analysis engine):*
   a) AbnerDescriptor:
      Implemented by ***AbnerAnnotator*** java class file.

   b) LingPipeDescriptor:
      Implemented by ***LingPipeAnnotator*** java class file, and the construction parameter is ***Library.***

3. *CasConsumerDescriptor:*
   Implemented by ***CasConsumer*** java class file, and the construction parameter is ***OutputDocument.***

1.3.2 Java file:

1. *CollectionReader*

It reads whole document named "hw2.in" sentence by sentence. Then, it splits id and text in each sentence and stores them to CAS respectively.

2. *Aggregated Analysis Engine(AAE)*

It is responsible for extracting gene name from each sentence. Since the accuracy of simply combining results generated by Abner and LingPipe is really low. So I set a threshold of confidence score for each annotator. For each annotator, if the confidence score of its result is larger than threshold, then write this result into new file.

   1) *AbnerAnnotator*
      Abner is a software tool for molecular biology text analysis. However, results generated by Abner always have strange symbols, such as '(' and '\', which are obviously not gene entities. So I use regular expression to calculate

confidence score of each result. This regular expression can cover most of the correct answer. Gene name will get lower confidence score if it not satisfies this regular expression.

2) *LingPipeAnnotator*

Most of method in this annotator is as same as homework 1. However, in order to avoid over-fitting, I use another trained model named as *NewTrainedLibrary* instead of original model offered by LingPipe.

3. *CasConsumer*

It receives id and text information for each gene extracted by Aggregate Analysis Engine(AAE). It also does some "selective work" like ignoring repeated gene mention. Then, it writes unrepeated result to a new file named "hw2-ningnaw.out".

The flowchart of my project is shown below, which shows the data flow of my project.

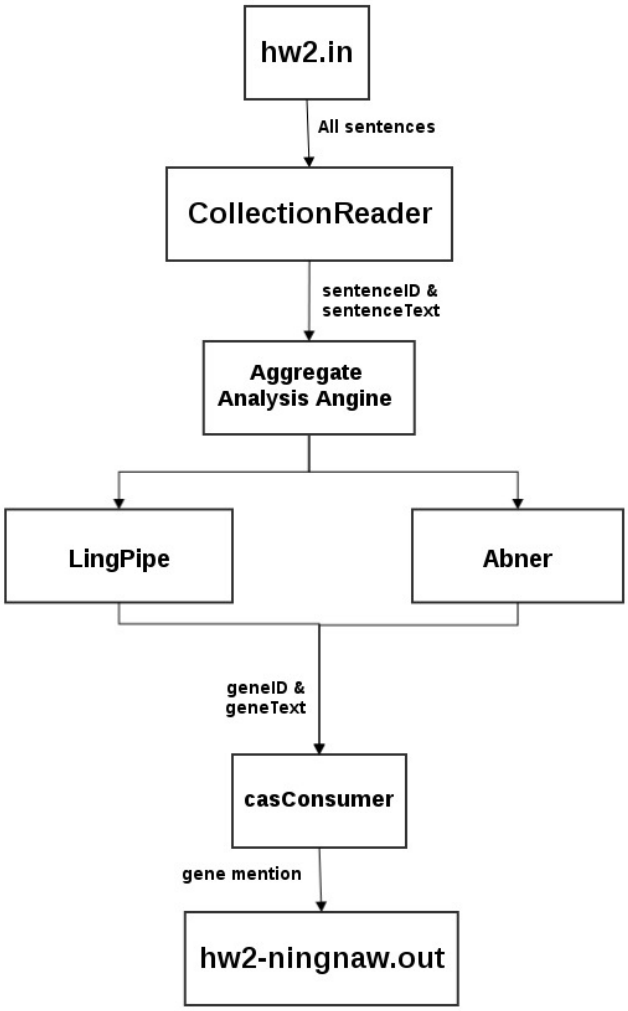

Figure2. Flowchart

2. Algorithmic Aspects

2.1 Solution Comparison:

To evaluate the performance of my AAE, I compared my AAE against Abner and LingPipe respectively. The performance report is shown in following table:

Performance Report for "sample.in" input file	Abner	LingPipe	AAE
Returned Correct Answer:	8987	12094	13723
Total Returned Answer:	19026	13898	24816
Supposed Answer:	18264	18264	18264
Precision:	0.4723536213602439	0.870197150669161	0.552990006447453
Recall:	0.4920608848007008	0.662176960140165	0.751368812965396
F-socre:	0.4820058997050147	0.752067657483987	0.637093779015784

From above comparison, we can see that my AAE has better precision and F-score than Abner. Also the recall of my AAE is better than both LingPipe and Abner.

2.2 Input-data Comparison:

To avoid over-fitting of my AAE, I also test another input file and evaluate result with gold standard.

I used test.in(./src/main/resources/data/test.in), which contains 5,000 sentences, as input file and compared my output with test.out (./src/main/resources/data/test.out), the performance report is shown in following table:

AAE Performance Report	"test.in"	"sample.in"
Returned Correct Answer:	4692	13723
Total Returned Answer:	8468	24816
Supposed Answer:	6330	18264
Precision:	0.5540859707132735	0.5529900064474532
Recall:	0.7412322274881517	0.7513688129653964
F-socre:	0.6341397486146777	0.6370937790157845