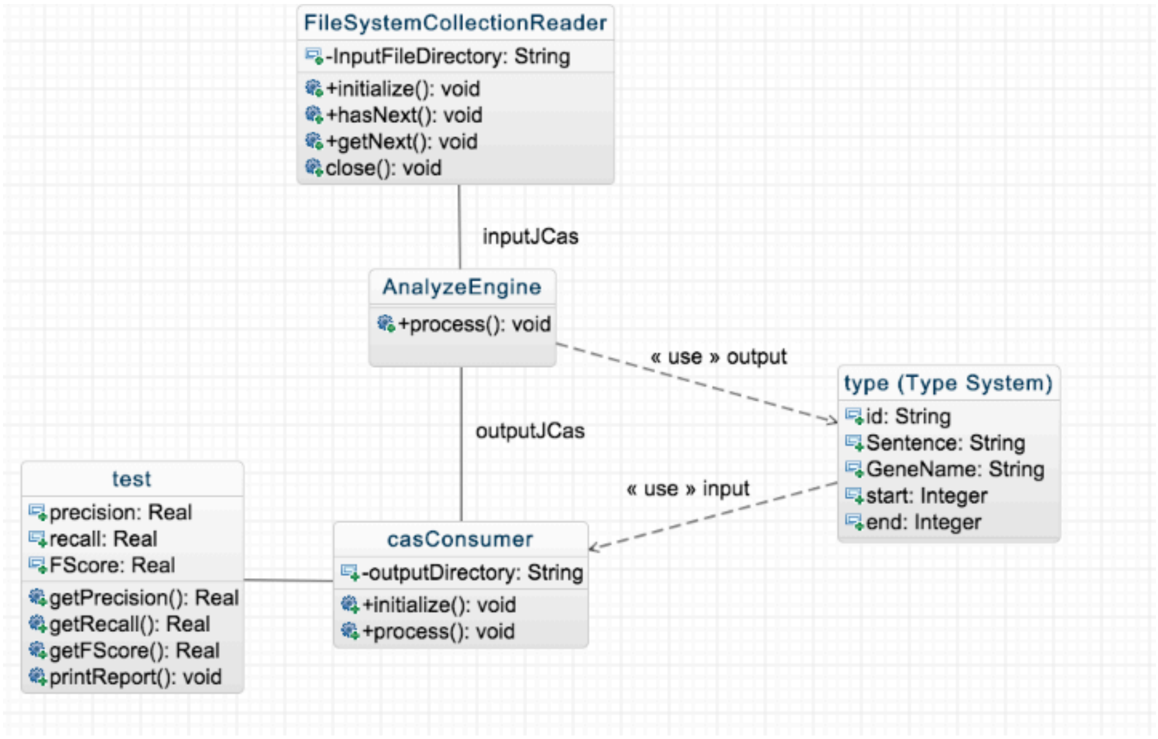
HW1-Report

1. System Architecture

I use a rather simple system architecture design pattern to realize this home work. There are three main components in my CPE system: Collection Reader that will read file, Analysis Engine that process the input, Cas Consumer that output the result. I will give detail description later. The architecture and general data flow can be seen below:



1.1 Collection reader (FileCollectionReader):

The collection reader use basic Java I/O to read file. By using the getNext() function, it read the input file line by line, then put the line document into JCas for Analysis Engine usage.

1.2 Analysis Engine (GeneAE):

1.2.1 Type System (type):

To keep my project simple, I only define one type system named type.

Type extends Annotator with start and end, and add three more new parameters:

id: store the id of the sentence/geneTag

Sentence: store the original text information

GeneName: store the name of the gene discovered by annotator

1.2.2 Annotator:

There is only one annotator in my project. It read a line of text from the JCas passed from collection reader. Separate the text into to part firstly, one is the id of the text and another is the Sentence. Then using the linpipe to find the gene tag and its start and end position in the Sentence. Write all this information into the System Type then pass it into the CAS consumer.

1.3 CAS consumer (GeneCASConsumer):

The CAS consumer read the type from the JCas passed from annotator, then calculate the right start and end position of the Gene Tag in the sentence. Finally write the id, gene name and start and end position in the assigned formation into the output file. In addition, I add a test class which will calculate the precision, recall and F score of my out put.

2. Algorithm Design and external Resources

Firstly, I tried using the PosTagNamedEntityRecognizer.java given by the instructor from the archetype. It can find most num. but the precision is very low. Average precision is around 10% and the recall is about 50%. Then I turn to use linpipe Name Entity Reorganization under the suggestion of my classmates. The resource linkage is http://alias-i.com/lingpipe/demos/tutorial/ne/read-me.html. All the methods I use are work together smoothly by using the same class for annotating the text com.aliasi.chunk. I use the model trained by expert called “ne-en-biogenetag.HmmChunker". It works very well, the Precision is enhanced to more than 0.75, the recall is also enhanced to more than 0.75. It runs much lower in speed than PosTagNamedEntityRecognizer but much higher in precicion and recall. After successfully implement the linpipe by using exist model, I tried to train a model by myself. Under the tuition of linpipe (http://alias-i.com/lingpipe/demos/tutorial/ne/read-me.html) I tried to train my model based on CoNLL 2002 NER Task Home. But finally, due to the time limitation, I give up it before the submission. You can see my TrainGeneTag.java class in my project.

3. Performance Evaluate:

3.1 Performance of PosTagNamedEntityRecognizer

precision: 0.102130436458

recall:0.51234232326

F: 0.17734334343

3.2 Performance of linpipe

precision: 0.7500701262272089

recall: 0.7827868852459017

F: 0.76470588295247

From the result we can see, the PosTagNamedEntityRecognizer only find the nums so it has a poor performance. The performance of Linpipe is relatively high, because I use the model trained from a biological dictionary.