Final Report for Homework 2

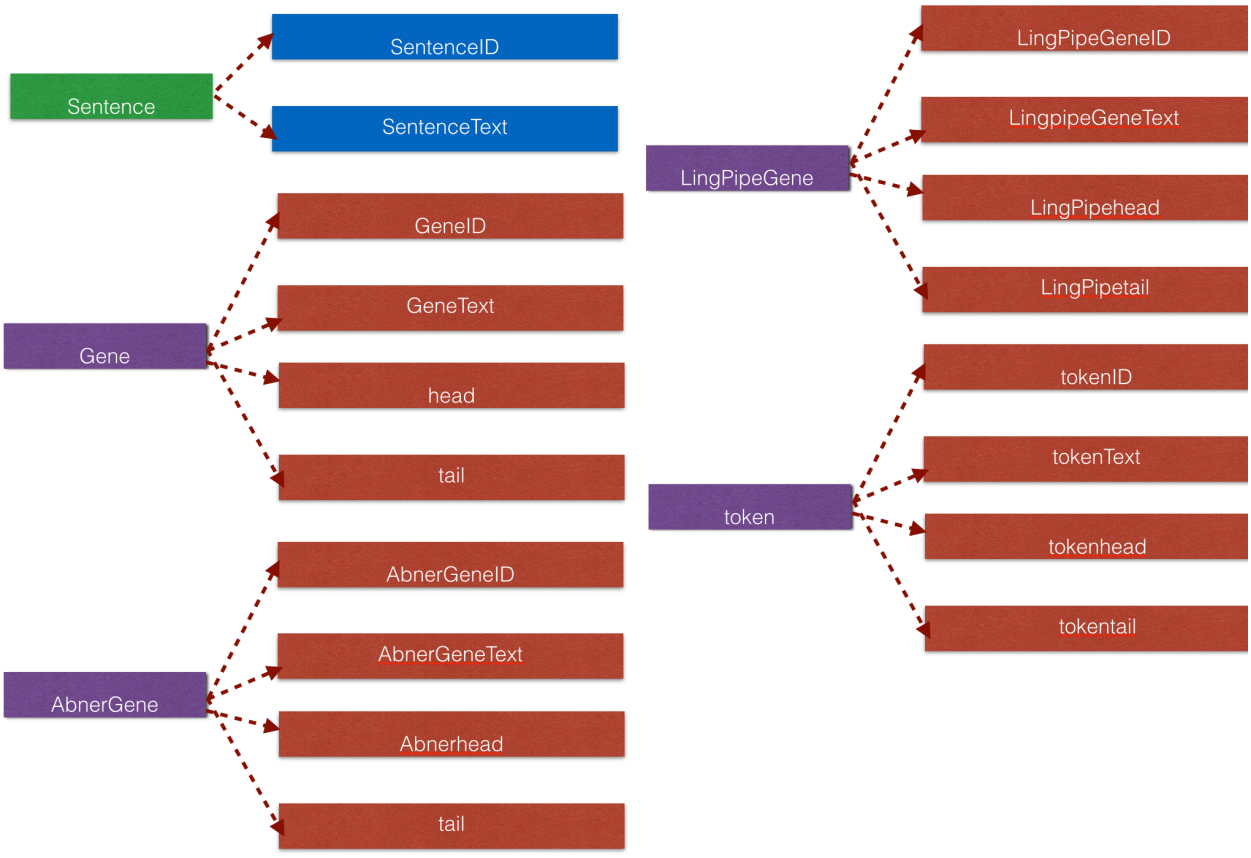
Andrew ID: leixiao

Name: Lei Xiao

1. Architecture Design

(1) Type System

In this part, I designed five types in it, which include Sentence, Gene, Abnergene, Lingpipegene, token.



Sentence has two functions, namely SentenceID and SentenceText. This type is used to deal with the mission that when reading the input file, the CollectionReader can automatically turned the input file into several sentences. And then separate the sentence into SentenceID, which could be used later for identifying the GeneID, and SentenceText, which part would be used to analyze the gene tags. The

partition method is to identify the location of the space(“ ”), and the part from the beginning of the sentence to the space can be recognized as SentenceID, and the rest part is SentenceText.

Gene has four functions, namely GeneID, GeneTag, head, and tail. This type is used in the annotator, which we got from the stanford method. Gene is the analyzing result of the Sentence type, and we can say that the annotator changes the Sentence type into gene type, and then sends it to the casConsumer, which could put the content of CAS in the output document. GeneID is the same as the SentenceID, which is used to find which sentence the gene tag belongs to. GeneTag is the analysis result from the annotator, which we get from the name entity recognizer. Head and tail means the begin position and end position of each gene tag of a specific sentence, which satisfies the output format. And head and tail are the return results (as a format of map)of the entity recognizer.

Abnergene is just like the gene type, except that this type is come from the abner method.

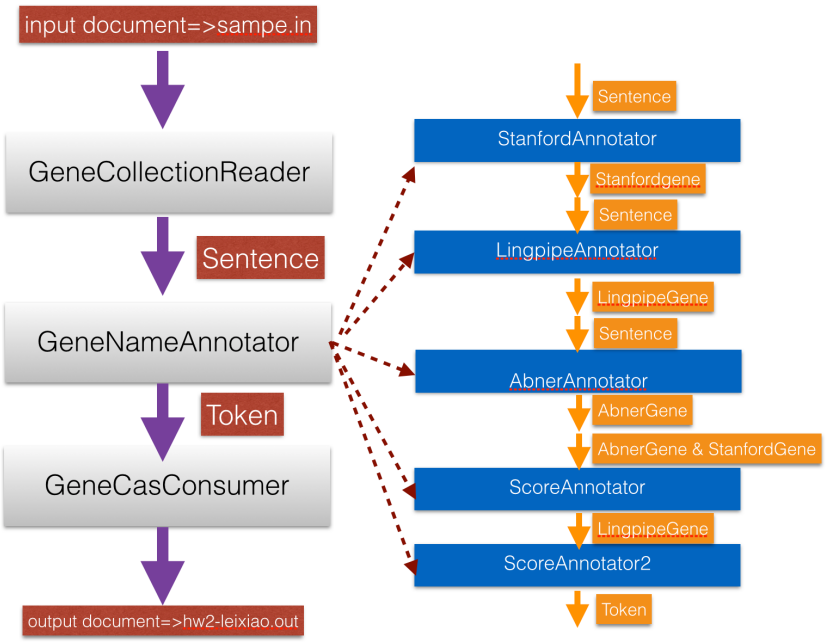
Lingpipegene is just like the gene type, except that this type is come from the lingpipe method.

Token type is the final type, which we would used in casConsumer part, which could put the content of CAS in the output document. Also, we should use the result of token to compare with the sample.out to get the precision, recall and f1-score of the system.

(2) Engineering Process

In this part, I will introduce the whole architecture of the process.

The following picture shows the process of dealing with data and specific types in each component of the system.

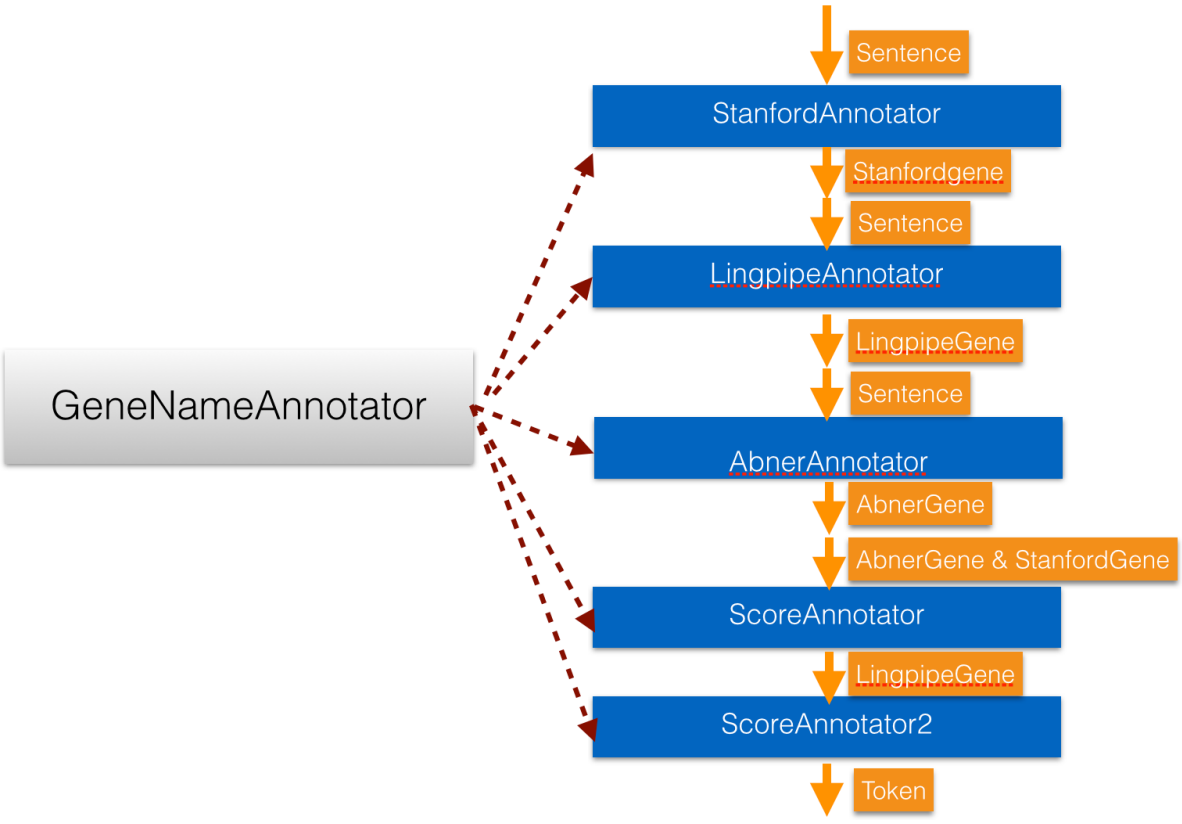


(3) Design Pattern

As can be seen from the above picture, the whole design of the system can be divided into three parts, namely GeneCollectionReader, GeneNameAnnotator and GeneCasConsumer. Each component has its own functions and roles in this system.

1) GeneCollectionReader, whose function is to read from the input document, for example, sample.in, and then separate the document into several sentences, and then give the value to SentenceID and SentenceText. The main idea of this part is to get the initial format (Sentence) from the input file and transfer it to the annotator to process it later.

2)GeneNameAnnotator, which is the most important part of this homework, and I design five annotators to implement the whole system, which include stanford_annotator, Lingpipe_annotator, Abner_Annotator, Score_Annotatore, and Score_Annotator2. And the data dealing process is as follows. And to deal with the recognition process, I use three methods to find the gene entity, and use the vote method to choose the gene-entity which have more possibility to be the gene entity. The checking method is that the one who won two votes would be stored in token.type and will be finally sent into the casConsumer, where they would be written in the output file.



The following are the details about each annotator.

- stanford_annotator: This is a casAnnotator of the system which use Stanford Method to recognize the gene Entity. The main idea of this part is to deal with the SentenceText, and recognize the location of the gene tag. Put the SentenceText into this method, it will automatically return a Hashmap. Through traversing the hash map, we could get the begin and end of each gene tag of the sentence.
- Lingpipe_annotator: This is a casAnnotator of the system which use LingPipe Method to recognize the gene Entity. The main idea of this part is to deal with the SentenceTextand recognize the location of the gene tag. Put the SentenceText into this method, it will automatically return a set.Through traversing the set, we could get gene entity, which we store in Lingpipegene Type.
- Abner_annotator: This is a casAnnotator of the system which use Abner Method to recognize the gene Entity. The main idea of this part is to deal with the SentenceText and recognize the location of the gene tag. Put the SentenceText into this method, it will automatically return a String[][].Through traversing the string[][], we could get gene entity, which we store in Abnergene Type.
- Score_annotator: reading from the stanford and write the gene which is recognized by stanford into the stanford.in
- Score_annotator: to search if the gene recognized by linpipe is also recognized by abner or stanford. if one of them has recognized it, we should add it into final token.

3)GeneCasConsumer, whose function is to get the gene type and then output them in the document, in this case, we put it into the hw2-leixiao.out. The main idea of this part is to obtain the gene type, transfer it into the format we want to store in the document, and then put it into the outputfile. Also, this part includes the evaluation method, which could calculate the precision, recall and f1-score.

(4) Additional Design

The system need several descriptors to help design the functions of each component. For example, we should build ColletionReaderDiscriptor.xml to describe the parameter, type system, capabilities, and indexes of the GeneCollectionReader.

In addition, we need to build a CPEDescriptor to implement all the components together to deal with the whole data process, which could be recognized the controller of the whole process. The method of building CPEDescriptor is from the UIMA CPE GUI, which could automatically build the CPEDescriptor.xml.

Finally, we need to add an aarDescriptor.xml to aggregate all the annotators, and the order of the annotators, which is the most important part of this homework.

2. Algorithm Design

1)Divide the sentence into id and text

To identify the location of the space(“ ”), and the part from the beginning of the sentence to the space can be recognized as SentenceID, and the rest part is SentenceText.

2)Use Name Entity Recognizer from Stanford

Put the SentenceText into this method, it will automatically return a Hashmap. Through traversing the hash map, we could get the begin and end of each gene tag of the sentence .

And the codes are as follows:

```
Stanford recognizer = null;
recognizer = new Stanford();
Map<Integer, Integer> mymap = recognizer.getGeneSpans(docText);
```

3)Use Name Entity Recognizer from LingPipe

Use the ne-en-bio-genetag-4.HmmChunker from LingPipe, we could easily get the result set. Through traversing the set, we could get the begin and end of each gene tag of the sentence .

And the codes are as follows:

```
Chunking chunking = chunker.chunk(docText);
Set<Chunk> set = chunking.chunkSet();
Iterator<Chunk> it = set.iterator();
```

3)Use Name Entity Recognizer from Abner

Put the SentenceText into this method, it will automatically return a String[][].Through traversing the string[][], we could get gene entity, which we store in Abnergene Type.

And the codes are as follows:

```
Tagger tag = new Tagger();
String[][] m = tag.getEntities(docText);
```

4)Use the vote method to get the final token

First I write all the tags, which is recognized by the stanford method, into the file stanford.in, and then I write the tags, which is recognized by Abner method into the file abner.in. Then I traverse all the gene

tags, which is recognized by lingpipe, and find if it is also recognized by stanford or Abner. If it is true, I will store it in token.type, which will be later sent into the CasConsumer.

5)f1-score Evaluation

I use f1-score method to evaluate the final result of this system.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

f1-score=2*(precision*recall)/(precision+recall)

3.Evaluation of this system

When using three methods together, I could get the result as follows:

precision:0.8858678301585613

recall:0.9115247741582261

f1_score:0.8985131816832619

4. Additional information

I read the source code of the DEIIS system and learn the whole process of dealing UIMA related framework. And then I read related materials of UIMA tutorial and run some examples of the uimaj-examples to understand the details of the three main components of the system, namely CollectionReader, NameAnnotator and CasConsumer.