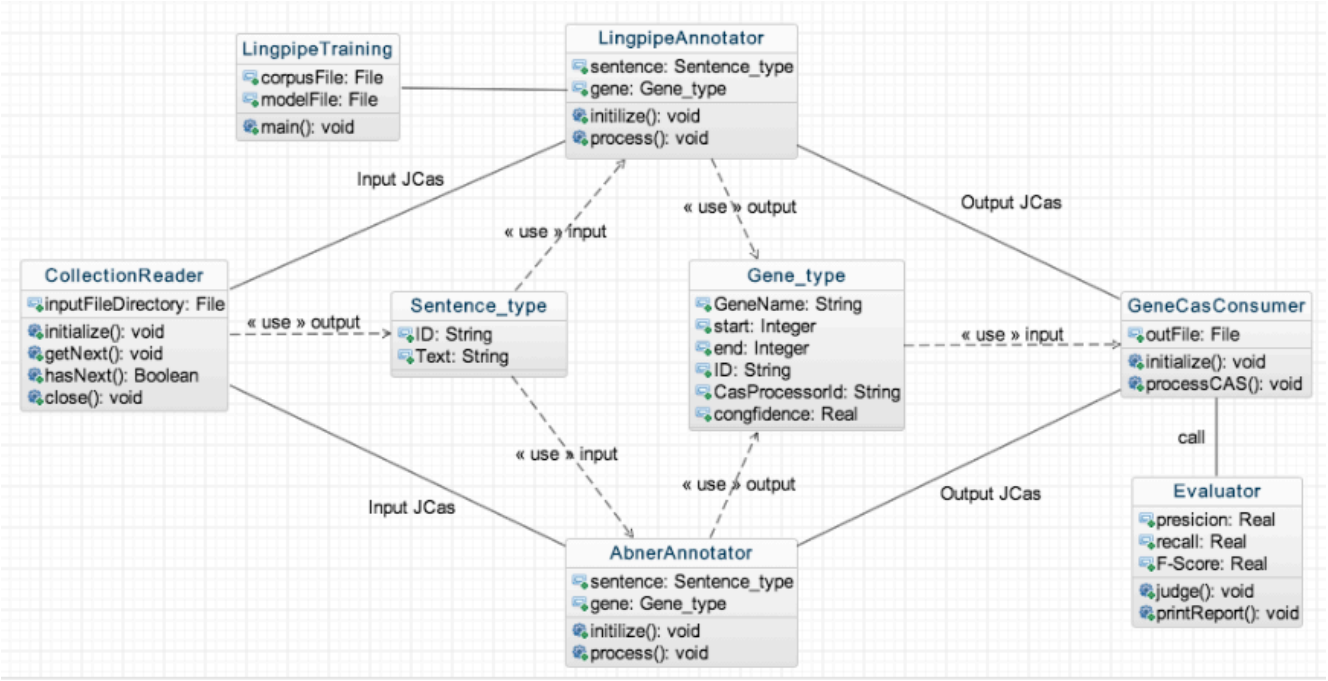
11693 - Software Method for Biotechnology

HW2-Report

1. System Architecture

Firstly, I will give a brief view of my design pattern of the system architecture. There are three main components in my CPE system: Collection Reader that reads file from the input and separate the sentence id and sentence text; Aggregated Analysis Engine that constituted by two paralleled annotator, which will find the gene tags and their positions; Cas Consumer that will filter the gene name and output the result. I will give detail description later. The architecture and general data flow can be seen below:



Graph 1 - architecture and general data flow of hw2-xiaomins

1.1 Type System (type):

I define another type named hw2_type which inherits the deiis_type.

The first type in hw2_type is called Sentence. It acts as a bridge between CollectionReader and Annotators. More specifically, it stores the sentence id and the rest text of the input sentence.

The second type in hw2 is called Gene. It connects the annotators to the Consumer. This type stores the gene name, start & end position of the gene(before delete white space) and sentence ID. It also inherit the CasProcessID and confidence from the super type.

1.2 Collection Reader:

The collection reader use basic Java I/O to read file(FileReader and BufferedReader). By using the getNext() function, it reads the input file line by line, then mark the ids of the sentences and the texts of the sentences. By using the type called Sentence, the Collection Reader pass those information to Analysis Engine through JCas.

1.3 Aggregated Analysis Engine (AggregatedAnnotator):

I tried three annotator separately, that is linpipe, abner and stanfordCoreNLP. Finally I decide to use both linpipe and abner.

1.3.1 Lingpipe Annotator

Comparing with hw1, I change from “Approximate Dictionary-Based Chunking model” to “N-Best Named Entity Chunking” to improve the performance. In detail, The initialize() function will firstly initialize the trainedModel that I trained myself. Then read documents from the JCas. By using the chunk() function provided by lingpipe, it can find the gene tag with the position (including white space) of this tag in the text. Next, filter those gene tag whose confidence is less than 0.6 because it is very less likely to be the right answer. Finally, it uses the Gene_type to pass the information to the Consumer through JCas.

1.3.2 Abner Annotator

The AbnerAnnotator will firstly initialize the Tagger class provided by Abner using the NLPBA model. Then it will use the getEntities() method of the Tagger class to find the gene tag in the text. Abner does not provide the position of the gene tag so I should find it by myself. After that, I add an Regular Expression to filter those gene tags that are less possible to be the answer. Finally, it uses the Gene_type to pass the information to the Consumer through JCas the same as what Lingpipe Annotator does.

1.4 CAS consumer (GeneCASConsumer):

The CAS consumer read the Gene_type from the JCas that passed from the two annotators. Then using the algorithm I describe below to filter the gene names tagged by the two annotators. It can also avoid duplicated name. Then the consumer will calculate the right start and end position without white space. Finally write the id, gene name and start and end position in the required formation into the output file. In addition, the consumer will call the Evaluator class which will calculate the precision, recall and F score of my output. The evaluation report will be print at end (the evaluation process has been commented/hide to avoid run time error during the assignment evaluation).

2. Algorithm Design and external Resources

Firstly, I tried the StanfordCoreNLP annotator. This is based on the class PosTagNamedEntityRecognizer.java given by the instructor. It can find most nouns. but the precision is very low. The average precision is around 10% and the recall is about 50%. This is because it is not designed for gene annotation.

Secondly, I use LingpipeAnnotator. It is based on Lingpipe Name Entity Reorganization (external source from http://alias-i.com/lingpipe). At the beginning, I use the "Approximate Dictionary-Based Chunking model" trained by others which called "ne-en-bio- genetag.HmmChunker". It works much better than StanfordCoreNPL Annotator. The Precision reachs more than 0.75, the recall is also more than 0.75. To gain better performance, I switch to "N-Best Named Entity Chunking" and at the same time, train my own model. Under the instruction in the webpage of linpipe, (http://alias-i.com/lingpipe/demos/tutorial), I trained my model based on the data from CoNLL 2002 NER Task Home. I named the new model trainedModel and use it for N-Best-Chunker. As a result, the performance improved a lot (precision 0.92 and recall 0.83).

Thirdly, I tried Abner, another Name Entity Reorganization performs well among my classmates. The source of this NER is http://pages.cs.wisc.edu/~bsettles/abner
Version 1.5 of Abner includes two models trained on the NLPBA and BioCreative corpora. I tried both of them and the performance is relatively the same. Therefore, I decide to use the first one because the second one only distinguish the protein names. At last, the precision of Abner Annotators about 0.84 and recall is about 0.73.

Finally, I decide to use both Linpipe and Abner. My algorithm to filter the results from the annotators is firstly, ignore the gene names whose length is less than 3, which is an experience from the golden standard. Secondly, select the gene name tagged by lingpipe whose confidence is higher than 0.6 and select the gene name tagged by abner that only contain characters from a-z, A-Z, 0-9 and '-' (I use a regular expression to implement that). Thirdly, compare the gene name tagged by both annotator. If the one tagged by lingpipe is the same as or the substring of the gene name tagged by abner, only output the shorter one and vice versa. It is because the longer one is very likely to be the wrong one and this operation can avoid duplication.

3. Performance Evaluate:

3.1 Performance of StanfordCoreNLPAnnotator

precision: 0.102130436458121

recall:0.512342323261234

F-score: 0.177343343431234

3.2.1 Performance of LinpipeAnnotator

precision: 0.7500701262272089

recall: 0.7827868852459017

F-score: 0.76470588295247

3.2.1 Performance of LinpipeAnnotator with N-Best-Chunker

precision: 0.923743414301245123

recall: 0.832346458478456763

F-score: 0.87262364545326254

3.3 Performance of AbnerAnnotator

precision: 0.843463235632432623

recall: 0.731346923692369234

F-score: 0.79135634734734574

3.4 Performance of AggregatedAnnotator

precision: 0.92145125123412564

recall: 0.78125014519281258

F-score: 0.8443125125143629 5