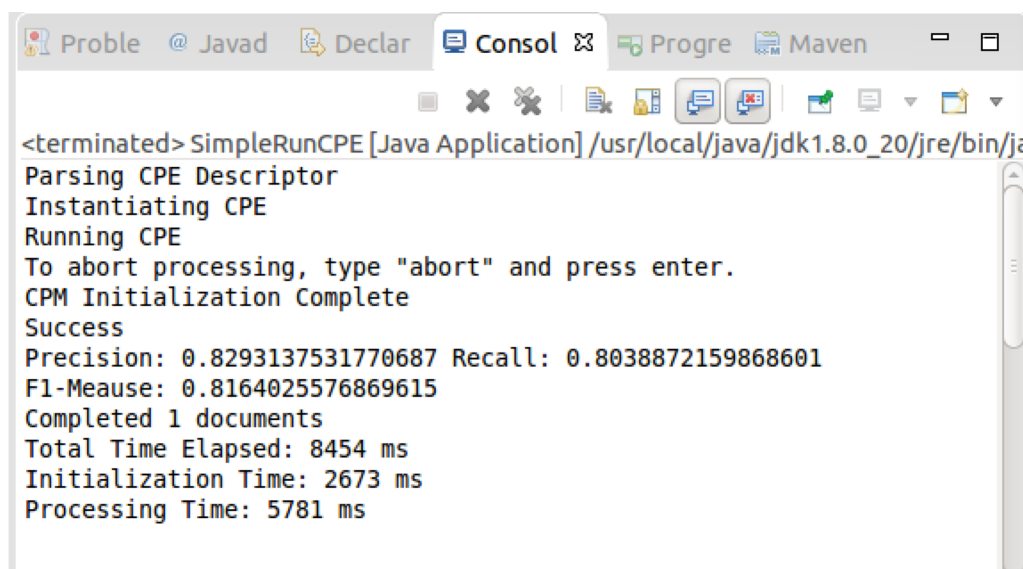


NER Report

This project implements a named entity recognizer based on UIMA SDK filtering out gene-related words plus its position in text.

Final running result:



```
<terminated> SimpleRunCPE [Java Application] /usr/local/java/jdk1.8.0_20/jre/bin/java
Parsing CPE Descriptor
Instantiating CPE
Running CPE
To abort processing, type "abort" and press enter.
CPM Initialization Complete
Success
Precision: 0.8293137531770687 Recall: 0.8038872159868601
F1-Measure: 0.8164025576869615
Completed 1 documents
Total Time Elapsed: 8454 ms
Initialization Time: 2673 ms
Processing Time: 5781 ms
```

The whole projects can be divided into two main parts: Architecture and Algorithm. The architecture is based on UIMA pipeline. The Figure 1 demonstrates the basic flow chart.

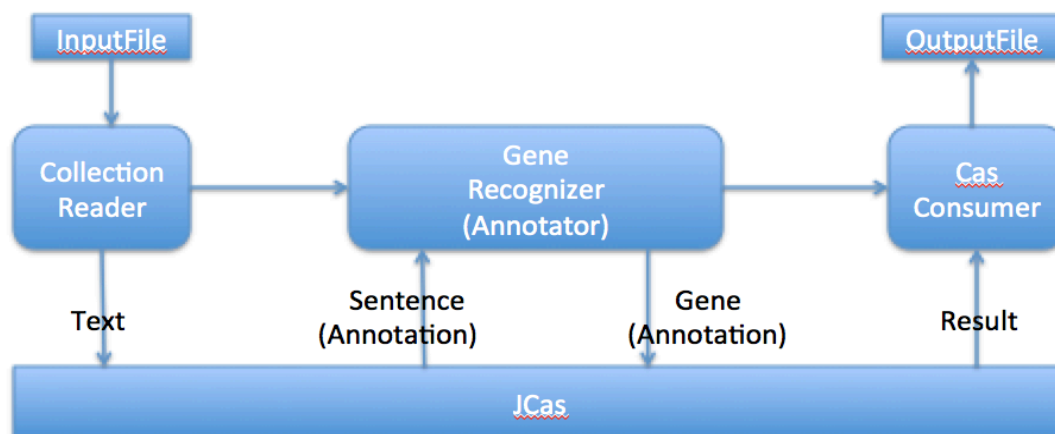


Figure 1

In this flow chart, there are several types constitutes type system. One type is called Sentence, which has two attributes: “ID” and “Words”. The ID stores the ID of each sentence and Words stores every word in sentence. The other type is called Gene, which stores words filtered by model. It has seven attributes: “ID”, “Words”, “Begin”, “End”, “Confidence”, “outershift”, “innershift”. ID and Words do the same kind of work as Sentence’s; Begin and end record the first character position and last character position of words in original sentence respectively. Moreover, we need to know how confident we give such result, so we store it in confidence attribute. Because of the model output position index including the blanks, to pinpoint the final position, we record the number of blanks before the first character of the word saving in attribute “outershift”, and the number of blanks inside the words saving in “innershift”. These two variables prepare for the next consumer phase.

At very start, I use the POS Recognizer in standard-core-NLP plus some regular match to filter out some phrase are apparently not the gene-related phrases, but the accuracy is very low about 50%. Then I browse the piazza to find another gene-Tag model. It was incorporated by Gene based on LingPipe4.10.jar. The model file names ne-en-bio-genetag.HmmChunker. Based on this model, we used method Confidence Named Entity Chunking. Chunk means word or phrase whose length cannot exceed MAX_N_BEST_CHUNKS set by programs. Finally procedure runs chunking calling the model to return chunks in order of confidence, which we take to be the probability of the chunk given the input text, $P(\text{chunk}|\text{text})$.

To fully use the pipeline characteristic of UIMA, in the phase of reading files, the Collection Reader read content of file line by line and store string line and ID separately into Sentence annotation. So when file is being read, the annotator has already extract gene-related phrase which save more operation time. In practice, it will save almost 75% time.

Reference:

1. Ling Pipe, Confidence Named Entity Chunking, website:

<http://aliasi.com/lingpipe/demos/tutorial/ne/read-me.html>, 2014, 9.24