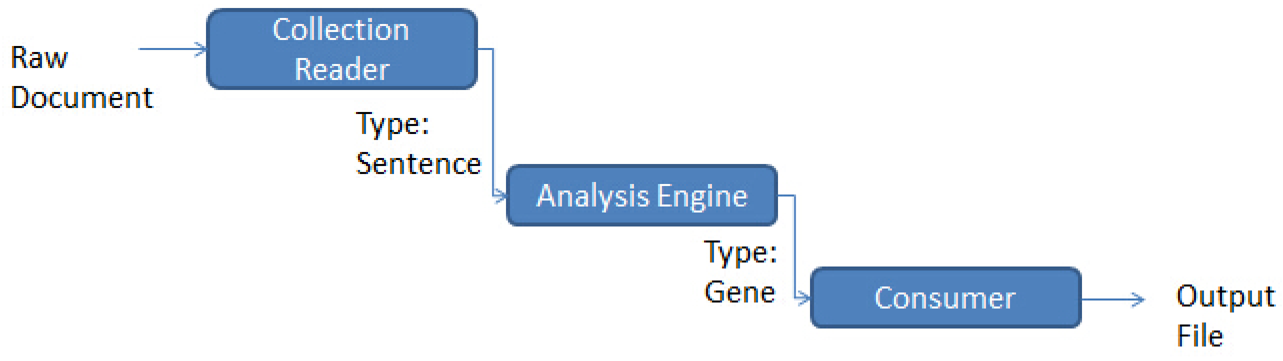
1: Architecture of Collection Processing Engine

The collection Processing Engine is described by cpeDescriptor.xml. It is composed of three main components, including Collection Reader, Primitive Analysis Engine, Consumer.



General Data Flow Collection Reader reads raw document data from input file by lines. Each line is considered as a sentence, which is composed of ID and text.Collection Reader stores the these information into the sentence type and passes them to Analysis Engine.

Analysis Engine receives sentence type and extracts the ID and text from them. ID is stored directly to the ID feature of gene type. Utilizing different machine learning algorithm or NLP tools, we can identify gene names and their positions in the sentence, store the information in gene type.

Then the gene type will be passed to Consumer. Consumer will extract the ID, begin and ending position in non-whitespace offset, and gene name from gene type. Then it outputs all the information in the required format.

2: Type System

sentence The type sentence has two attributes, String id and String text. Collection reader reads raw document and generate the sentence type to store the ID and text of each sentence.

gene The type gene has four attributes. Id stores the id of the sentence. GeneName stores the name of genes. Begin stores where the gene name starts in the sentence. End stores where the gene name ends in the sentence. Both Begin and End are corresponding to the non-whitespace offset in the sentence.

3: Design Pattern

Singleton Singleton pattern is applied to the Named Entity Recognizer. We only need one instance of the recognizer. If we initialize an instance every time we call it, it would waste much time.

4: NLP Tools Applied

I tried both Stanford NLP library and Lingpipe.

Stanford NLP Library The Named Entity Recognizer from Stanford NLP Library will process the text passed in, and returns a Map, which contains all potential gene names' starting and ending positions in the sentence. Then go through the Map and check each word's length. If it's larger than the threshold, add the id, starting and ending position, gene name to a gene type.

However, it can only recognize noun phrases instead of gene names, so the accuracy is not very good. To improve, I applied a simple lexical rule. If the length of the phrase recognized by Standford NLP is smaller than a given threshold, we will consider it not to be a gene name and ignore it. By adjusting the threshold, we can do trade of between the precision and recall.

Lingpipe The usage of Lingpipe is similar to Standford NLP. It also will return a Map containing information about gene names in the sentence. I used the statistical named entity recognizer with First-Best rule. That is, we only choose the way with highest confidence to split the sentence and extract the words. I used a trained dataset provided by Lingpipe official site.

External Training Data A data set trained with Hidden Markov Model will be utilized by Lingpipe NLP Tools to help identify gene names in the sentence.

Position of gene names Mind that we should store the non-whitespace offset of each word. So we need to keep track of the number of whitespaces before each word so that we can deduct the extra offset from the gene names' starting and ending positions.

5: Evaluation

I implemented an Evaluation class which is responsible for evaluating different algorithms' performance. First we read and store all the gene names in the given sample.out in a hash table. Then go through my own output and check which of them are in the hash table. Denote the number of matchings as P, total number of gene names in given sample.out as Q, total number of gene names in my output file as R. We can calculate $Precision = \frac{P}{R}$, $Recall = \frac{P}{Q}$. And the $F1Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$.

Stanford NLP For Stanford NLP library, I tried different thresholds to trade off the precision and recall, and find that when the threshold is set to around 5, the result is relatively better. Of course still much worse than Lingpipe with a trained data set. The statistics are as follows:

Hitting number: 5280

Total Number in my output: 32952

Total Number in sample: 12306

Precision = 0.16023306627822287

Recall = 0.42905899561189664

F1 Score = 0.23332891422510937

Lingpipe For Lingpipe, using the provided training set, the statistics is as follows:

Hitting number: 9452

Total Number in my output: 12224

Total Number in sample: 12306

Precision = 0.7732329842931938

Recall = 0.768080611084024

F1 Score = 0.7706481858948226

6: Conclusion

For this problem supervised learning algorithm with a good training data set will perform much better than a simple rule-based algorithm. Of course it requires more time and resources to do the training first. We need to ensure the data set is proper and be careful not to overfit on the training set.