11-791: Design and Engineering of Intelligent Information System

Homework Assignment #1

Yu-Hsin Kuo
Andrew ID: yuhsink

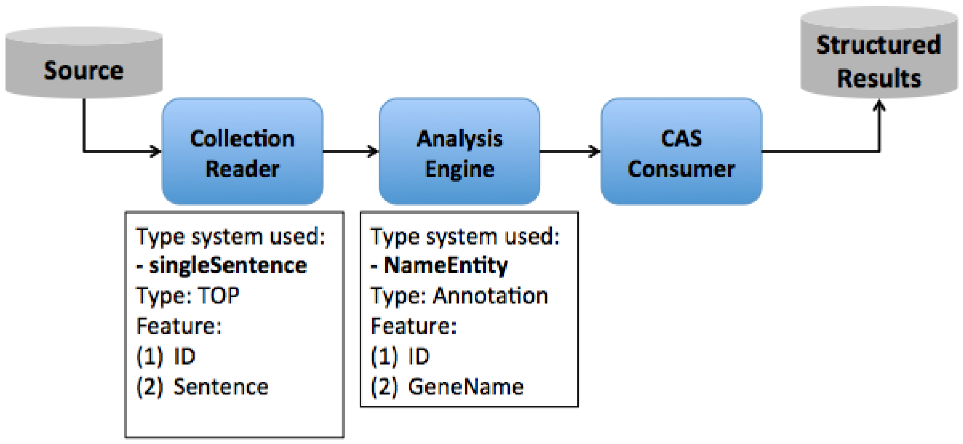
Due: September 24, 2014

Report

Type System
Type name: NameEntity
Supertype: uima.tcas.Annotation
Feature:
- GeneName (uima.cas.String)
- ID (uima.cas.String)

Type name: singleSentence
Supertype: uima.tcas.TOP
Feature:
- ID (uima.cas.String)
- sentence (uima.cas.String)

Design
There are three main components in my implementation: (1) collection reader, (2) analysis engine (which is our name-entity recognizer) and (3) CAS consumer. The brief overview of my design is shown below followed by the explanations:



(1) Collection reader: basically, its function is to open a file and read sentences line by line. Here, we *separate* the ID and the sentence and store in the "singleSentence" type. The "singleSentence" is a TOP type with ID (String) and sentence (String) as two features. To open a file, we didn't hard-wired the file, instead, we set a parameter called "fileName" which you can specify its path in collectionReaderDescriptor.xml.

(2) Analysis Engine: the main function performed here is to do name entity recognition. Here, we used the name-entity recognizer from LingPipe which we will explain its principles later. Since we have to load a model for our name-entity recognizer, we set a parameter called "model" and we can access it through getConfigParameterValue(). The results our recognizer returns are the start and end index of the name entity. However, the indexes should be calculated without whitespace, so we calculate them here and store in the "NameEntity" type. The "NameEntity" is an Annotation type with ID (String) and GeneName (String) as two features.

∗ LingPipe Name Entity Recognizer:
In LingPipe, name-entity recognition involves the supervised training of a statistical model or more direct methods like dictionary matching or regular expression matching. In this homework, we are running a **statistical name-entity recognizer**. The model we used is a **HMM model** trained on **GENETAG corpus**, which contains 20K sentences tagged with gene/protein names. The machine learning technique used here is **hidden Markov model (HMM)**. Basically, given the HMM model we have, it does sequence labeling for each sentence.

A sequence labeling problem is to map a sentence x_1, x_2,...,x_n to a tag sequence y_1, y_2,...,y_n. Essentially, we would like to find the most likely tag sequence for an input sentence. This is the problem of finding:

$$\underset{y_1, y_2, ..., y_n}{\operatorname{argmax}} p(x_1, x_2, ..., x_n, y_1, y_2, ..., y_n)$$

To find the best sequence label, we can use Viterbi algorithm.

(3) CAS Consumer: it writes the final output to a file where each line contains sentence ID, text span, and the gene mention. Again, here we need to write the output to a file, we set a parameter called "outputFile" and again we can access its value through getConfigParameterValue().

Design Pattern Used:

- Singleton pattern

I created another Utility class that has static method to calculate the start index in a sentence given a gene mention because this method should be used by different classes.

Evaluation:

The metrics I am using here are F1, precision and recall and I also wrote a script to evaluate the performance. The definition of those metrics are:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where precision and recall are:

$$\text{Recall} = \frac{\#\text{Gene Mention Correctly detected}}{\#\text{Gene Mention in Gold Answer}} \times 100\%$$

and

$$\text{Precision} = \frac{\#\text{Gene Mention correctly detected}}{\#\text{Name Entity reported}} \times 100\%$$

The numbers are below:

Baseline system (TA provided):

Precision	0.10
Recall	0.54
F1	0.17

My implementation using LingPipe:

Precision	0.75
Recall	0.82
F1	0.78