11-791
Homework 1 Report
Nikolas Wolfe
Andrew: nwolfe

Overview

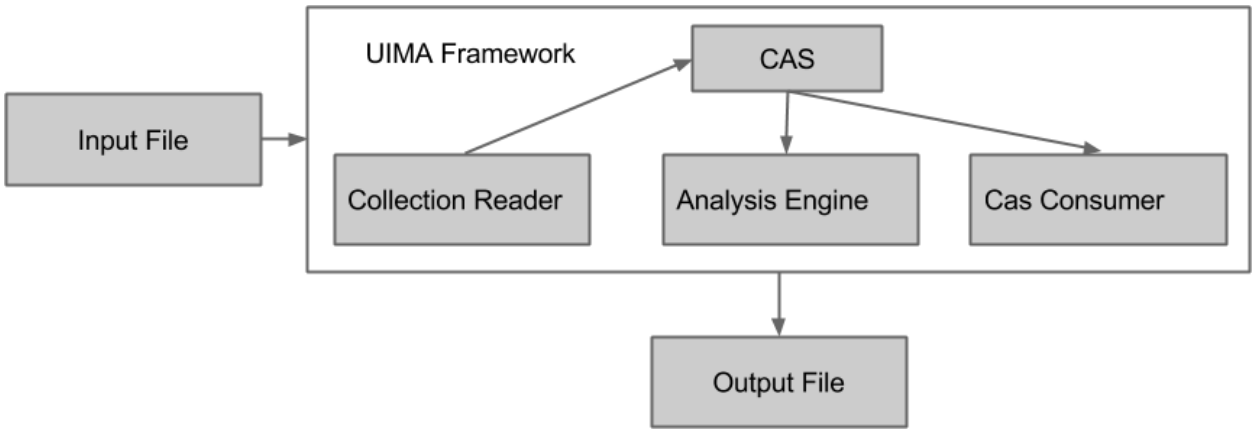
Using an Analysis Engine is an important first step to leveraging the power of the UIMA framework to do data analysis on an unstructured text corpus. In this homework, we learned how to set up an Analysis Engine to do gene-based named entity recognition on medical text data.

Type System

I use a fairly straightforward type system. I have a **Gene** Annotation type, which has a contentString attribute which takes the text of the gene mention. I have a **Sentence** type which stores the text of a sentence and its ID. These are pruned from the text in my Collection Reader. Finally there is a **Token** type which stores the Part-of-Speech of a given word and the word itself.

Engineering and Design Patterns

The overall design concept is as follows:



An input file is given over to the UIMA framework. An output is produced. In between, the file is read by a Collection Reader, which reads the Input File and passes the data to the CAS (a data collection object). The CAS is then passed to an Analysis Engine which does some processing of the data in the CAS, and then a CasConsumer is used to take the data in the CAS and output it to a file.

As the general UIMA framework takes advantage of many patterns already, engineering a pipeline to do NER is a fairly simple process of extending several base class implementations, e.g. CollectionReader_ImplBase, JCasAnnotator_ImplBase, and CasConsumer_ImplBase. In general these classes are run as templates, and our job is to implement the appropriate hook methods e.g. processCas(), initialize(), and destroy(), which are called by the superclass at runtime. This allows for the superclass to defer certain aspects of its job to subclasses while maintaining the basic structure of the overall template

pattern. In the case of my Analysis Engine, I compose the engine with the NBestChunker HMM Chunker model from LingPipe, and use it to create Gene Annotations which I then index in a CAS object. This allows us to decouple the processing of the data from the storage. Furthermore we are not required to know anything about how the data is persisted, we just know that when our template hook methods are called that we are given a jCas object which is able to provide us with the data we need to do the work we have to do.

Understanding the design patterns used by the UIMA framework helps us to better understand how to implement our code, and not redo work. Recognizing the intent of the various classes is key, for instance, the Collection Reader object can in some ways be thought of as a partial wrapper or adapter of a Java Scanner object, which reads a file and contains more or less the same functionality. We are extending this functionality via subclassing, but the macro-level behavior is the same.

Results

When I run this pipeline on the sample.in data I produce an output file, which when compared to the gold standard sample.out file produces the following results:

```
precision: 0.769
recall: 0.849
f1: 0.807
```

Q/A

1. Please identify/describe any machine learning techniques used

I used the LingPipe HMM-based Tagger, which of course uses a Hidden Markov Model trained on an English text corpus with Gene tags in order to do Named Entity Recognition of genes.

2. Please identify/describe any NLP techniques/components used

To start, I used the Stanford CoreNLP Part-of-Speech Recognizer. The assumption here is that Genes are nouns and so being able to distinguish nouns reduces the amount of searching required to mark gene mentions in text correctly. I threw this out after realizing that it was much easier to use UIMA components with LingPipe.

3. Please identify/describe any external (marked up text) training data used

I only used the provided sample.in file and the HMM Chunker model from Lingpipe which is availble here: http://alias-i.com/lingpipe/demos/models/ne-en-bio-genetag.HmmChunker

4. Please identify/describe any external lexical resources (terminology lists)used

I did not use any external lexical resources.

5. Please describe any rule sets used

I did not use any rule sets.

6. If your system interacts with or uses data from any biological database(s), please describe.

I do not interact with any biological databases, I simply use the Lingpipe HMM Chunker to find the gene mentions in the text.

7. Please identify/describe any other relevant resources used to train/develop your system

One of the benefits of using a framework like UIMA is that I can use models created by third parties and not have to train anything of my own. I simply used the Lingpipe HMM Chunker.

8. Please describe the general data flow in your system

This pipeline consists of three major steps. The first, predictably, is the use of a Collection Reader to read through an input file and load the data into a CAS object which can be passed to an Analysis Engine. The second step involves the Analysis Engine which leverages the Lingpipe HMM Chunker to break apart sentences into chunks which are believed to contain mentions of genes. This implementation is largely straightforward and did not involve anything beyond creating the basic descriptor files and extracting the appropriate annotations using the NER Analysis Engine. The final step is the combination step, which takes places entirely inside a CAS Consumer, where I read the data in the CAS and output it to a file, specified in the Cas Consumer XML descriptor.