HW-2 REPORT

Name: Prajwal Yadapadithaya
AndrewID: pyadapad

Architecture and Design:

This Gene Named Entity Recognizer is implemented using the UIMA framework, with making use of the LingPipe toolkit and Abner toolkit for text processing. In this section, I will be describing the implementation of the type system and other descriptors used in the UIMA pipeline for Gene Named Entity Recognition, along with the implementation and design details.

Type System: *deiis_types.xml*
The type system describes the feature names used in the pipeline. The different types used in this system are as follows:
edu.cmu..deiis.types.Annotation - has *casProcessorId, confidence, geneData, startOffset and endOffset*
edu.cmu..deiis.types.InputData - Consists of *sentenceId* and *geneData*
edu.cmu..deiis.types.Results - Consists of *senetenceId*, *geneProduct*, *geneStartOffset* and *geneEndOffset*

Collection Reader: *GeneCollectionReader.xml (Implementation: GeneCollectionReader.java)*
The collection reader is responsible for reading input text line by line, and splitting the sentenceId and the string containing gene information *(geneData)* and passing the information to the analysis engine using the *InputData* feature in the Cas.

Analysis Engine: *GeneAnalysisDescriptor.xml*
Analysis engine comprises of a three annotators in this implementation. This is responsible for triggering the execution of *LingPipeGeneDataProcessor, AbnerGeneDataProcessor and GeneDataAggregator* in that sequence.

Annotator: *GeneDataProcessorL.xml (Implementation: LingPipeGeneDataProcessor.java)*
GeneDataProcessor is responsible for using the LingPipe named entity recognition library to extract the gene names based on a trained model (GENTAG). *LingPipeGeneDataProcessor* is also responsible for calculating the start and end offsets for the output, and passing the output to *GeneDataAggregator* for results aggregation.

Annotator: *GeneDataProcessorA.xml (Implementation: AbnerGeneDataProcessor.java)*
GeneDataProcessor is responsible for using the Abner biomedical named entity recognition library to extract the gene names based on a pre-existing model (BioCreative). *GeneDataProcessor* is also responsible for calculating the start and end offsets for the output, and passing the output to *GeneDataAggregator* for results aggregation.

Annotator: *GeneDataAggregator.xml (Implementation: GeneDataAggregator.java)*
GeneDataAggregator is responsible for reading the results from *LingPipeGeneDataProcessor* and *AbnerGeneDataProcessor* and filtering the results based on confidence of results obtained. *GeneDataAggregator* filters the results from *LingPipeGeneDataProcessor* based on the following rule: If the confidence of a named entity is greater than 0.575, the entry is directly added to the *Results* feature. If the confidence is between 0.4 and 0.575, then the entry is cross checked with results from the *AbnerGeneDataProcessor,* and if there exists an entry, it is added to the *Results* feature. If both these conditions fail, then the named entity is ignored and not included in the final results. *GeneDataProcessor* is also responsible for passing on the start and end offsets for the output Cas Consumer using the *Results* feature.

Cas Consumer: *GeneCasConsumer.xml (Implementation: GeneCasConsumer.java)*
GeneCasConsumer is responsible for writing the results to the specified output file. *GeneDataProcessor* sends an entry for each gene name obtained to the consumer, which processes them and writes them to the output file.

Algorithms:

The Analysis Engine in this implementation has two annotators called *LingPipeGeneDataProcessor* and *AbnerGeneDataProcessor* which uses LingPipe library and Abner biomedical named entity recognizer respectively for processing the input text and extract gene data. The LingPipe named entity recognizer expects a string to be given as input on which it does the required processing to extract gene names. The LingPipe Chunker class is trained using the model file (*gene_model*), and it returns slices of the input string which match to the gene names obtained from the training model, along with a confidence attribute associated with it. Internally, LingPipe provides an implementation for the Aho-Corasick string matching algorithm, which is used for named entity recognition. The Abner biomedical named entity recognizer uses Conditional Random Fields (CRFs) and Viterbi Algorithm for tagging named entities. I have included the model file used for this implementation in the project.

Data Flow in the System:

The data flow follows a typical UIMA architectural data flow. As described in the architecture, in this implementation, *GeneCollectionReader* (Collection Reader) reads from the input file line by line, splits the sentence ID and text in which we are interested, before passing the information to *LingPipeGeneDataProcessor, AbnerGeneDataProcessor and GeneDataAggregator* (Annotators) which uses LingPipe library and Abner biomedical named entity recognizer to check the data and extract gene information with the help of an existing training model. It then generates the required data for the *GeneCasConsumer* (Consumer), which finally writes the extracted gene information to the output file (hw2-pyadapad.out)

Experiments:

The pipeline was tested with the input file which was given in the archetype. The output generated was compared with the sample output file given in the archetype. Various values for the filter parameters were tested before finalizing the solution.

Results:

This gene named entity recognizer implemented using UIMA, LingPipe toolkit and Abner library is able to extract gene names out of the given input file in less than a minute.

References:

- http://burrsettles.com/pub/settles.bioinf05.pdf
- http://uima.apache.org/downloads/releaseDocs/2.1.0-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html
- http://alias-i.com/lingpipe/demos/tutorial/ne/read-me.html