11-693 Software Methods for Biotechnology

Name: Carol Cheng
Andrew ID: carolche

Homework 1

Task 3.4 Final report

1 External training data used

The trained model used by the name entity recognizer, "ne-en-bio-genetag.HmmChunker" is from GeneTag named entity data, provided by the Unites States National Center for Biotechnology Information (NCBI), a part of the U.S. Library of Medicine, and described in this paper:

- Tanabe, L., N. Xie L. H. Thom, W. Matten, And W. J. Wilbur. 2004. GENETAG: a tagged corpus for gene/protein named entity recognition. *BMC Bioinformatics 2005*, **6**(Suppl 1):S3.

The training details and implemented code was documented in LingPipe's tutorial.
http://alias-i.com/lingpipe/demos/tutorial/ne/read-me.html

2 NLP techniques/components used

This gene name tagger used **LingPipe 4.1.0** as a dependency for doing name entity recognition. Specifically, it imports Chunk package to run a statistical named entity recognizer. Two kinds of chunking were used, including first-best named entity chunking and confidence named entity chunking. These two were used in LingPipeRunChucker class, whose object will be created when analysis engine processing sentences. By providing model file path and specifying which chunking will be used, the LingPipeRunChuncker object can call methods inside Chunk package to finish the task of name entity recognition. First-best named entity chunking will only record the best set of detected genes. Confidence named entity chunking can provide possible candidates in order of confidence. In this framework, the selected gene names were those whose confidence were larger than 0.65 and whose length were larger than 1. Since in my previous experience, if I use confidence named entity chunking to detect genes, there will be some errors happen in single letter like N. In order to exclude these wrong entities, it only returns those entities whose length are longer than 1.

3 Name Entity Recognizer with UIMA framework

3.1 Type system

The type system described by the *typeSystemDescriptor.xml* is used by the UIMA framework. There are types for input sentences and output gene annotations as listed by follow.

- **`Type.input.Sentence`**

  This type contains two features, ***id*** and ***text***, which are both String objects and the type is mainly designed for input data. The type is used to store every line of separated identifiers and texts. It is used by collection reader and GeneCASConsumer of this system. GeneCollectionReader will use this type to save id and text of each sentence. GeneCASConsumer will use this type to get the original text in order to calculate the span of the given gene mention without counting white spaces.

- **`Type.output.Gene`**

  This type is designed for store the identified gene mentions and it contains two features, ***id*** and ***name***. Id is the identifier of the sentence in which the gene mention is found and name is the name of the gene mention. This type is mainly used by GeneAnnotator of this system for saving the identified gene mentions returned by LingPipeRunChuncker object and will be stored in CAS to pass to GeneCASConsumer. GeneCASConsumer will print gene mentions identified from each given sentence.

3.2 Collection processing engine

This collection processing engine is described by *CPEdescriptor.xml*. It starts from GeneCollectionReader, which takes input file, reads the sentences line by line and passes the processed Sentence type in CAS to GeneAnnotator. GeneAnnotator then creates LingPipeRunChunker object to run the pre-trained model identifying the gene mentions in each sentence and store them to Gene type in CAS. Finally, GeneCasConsumer reads original sentences and annotaions of genes from Sentence and Gene type of objects and print out the identified gene mentions in a specified format.

- **`GeneCollectionReader`**

  This collection reader is described by *collectionReaderDescriptor.xml.* GeneCollectionReader reads the contents of input file(hw1.in) line by line and splits each line by white space. It puts the separated identifier and text into an instance of Sentence type. These instances of Sentence type will be stored in CAS and then pass to GeneAnnotator.

- **`GeneAnnotator`**

  This annotator is described by aeDescriptor.xml. GeneAnnotator is the analysis engine of this system. It gets the path of gene tag model file, ***ne-en-bio-genetag.HmmChunker***, from Resource Manager component, then uses the path to create a specified type of LingPipeRunChucker. Here it can either create a LingPipeRunChucker run by Chunker or ConfidenceChunker, given the second argument of LingPipeRunChucker's constructor. If the argument is 1, it creates Chunker. If the argument is 2, it creates ConfidenceChunker. The Chunker will be used to identified the gene mentions in the given sentence. The results will be store in a map and be returned by LingPipeRunChucker. GeneAnnotator then iterates the map to create gene annotations for each sentence. The span of the gene mention, the name of the gene and the id of the sentence in which the gene was found will be store in Gene type and be put into CAS to pass to GeneCASConsumer.

- **`GeneCASConsumer`**

  This CAS consumer is described by casConsumerDescriptor.xml. GeneCASConsumer will first create a file(hw1-carolche.out). Then it iterates through every instance of Sentence type to get the original text in order to re-calculate the correct span without considering white spaces. Here it gets the span identified by LingPipeRunChucker from a instance of Gene type and stores it in a Span object. Span class has a method called ignoreSpace in which a new span will be calculated for the the given gene mention without counting white spaces. The final step is to print out the gene tag results in a format of “sentence identifier|start end|text” to the specified file.