Report for 11-791 Design and Engineering of Intelligent Information System Fall 2012, Homework 1

Yuchen Zhang (yuchenz)

System Architecture

The intelligent information system does the job of recognizing gene names from a document. The document consists of a number of lines (sentences), in which each line consists of a sentence id (fist word) and its content (rest of the words). The intelligent information system takes a document, and outputs the detected gene's sentence id, start and end location, as well as the gene's name to the output file.

(1) Type System

The type system consists of tree types:

- **BaseAnnotation**: base type for the other two types, inherit from uima.tcas.Annotation, contains one attribute called "source".
- **GeneInputSentence**: stores the input sentence in the form of (sent id, sent content), inherit from **BaseAnnotation**.
- **GeneAnnotation**: the type that stores the actual annotation in the form of (sent id, start, end, text), inherit from **BaseAnnotation**.

(2) Collection Reader

The HuskieGeneFileSystemReader class defines the collection reader for this IIS. The object of this class is responsible for getting the input filename from the configuration parameter "inputFile", and load the entire file into one CAS.

(3) Cas Processors

- **GeneInputSentenceAnnotator**
  This class extends the **JcasAnnotator_ImplBase** class. It's responsible for retrieving each raw sentence from the CAS returned by **HuskieGeneFileSystemReader**, and annotate each sentence with the type **GeneInputSentence**. This step is basically strip out the sentence ids from the sentence contents.

- **HuskieGeneAnnotator**
  This class also extends the **JcasAnnotator_ImplBase** class. It's responsible for actually annotating the genes from the input sentences. It initializes a **RegexModel** object (described later), and iterate through each sentence from the CAS returned by **GeneInputSentenceAnnotator**, and calls the RegexModel.annotateLine() function to annotate the sentence with gene names. It then set the sentence ids of the new annotations and add them to index.

(4) Cas Consumers

- **HuskieGeneFileSystemCasConsumer**
  This class extends the **CasConsumer_ImplBase** class. It's responsible for writing the final CAS returned by **HuskieGeneAnnotator** to a file on disk using the specified format. It retrieves the filename for output from the configuration parameter "outputFile", and writes each gene annotation to this file using the format of "sent id|

start end|gene".

- **HuskieGeneRecognitionEvaluator**
  This class also extends the **CasConsumer_ImplBase** class. It's responsible for calculating the precision, recall and F1 score for the annotation result. It retrieves the reference filename and the result filename from configuration parameters "referencePath" and "resultPath". Then it calculates the above statistics from the number of relevant annotations and number of retrieved annotations.

  The statistics are written to standard output.

Annotation Algorithm

The **HuskieGeneAnnotator** object uses a regular expression trained on the sample output data. The basic idea is that we assume the genes look somewhat alike. For example, if we know there's a gene called "factor IX", we may assume names such as "factor V", or "factor IX" are also genes. So the training procedure takes the sample output data (or part of it if divided into training and testing), substitute the following patterns with the corresponding regular expression:

(1) Numbers → [0-9]+
(2) Roman numerals → [IVX]+
(3) Greek alphabets → \b(alpha|beta|gamma … psi|omega)\b
(4) Single capitalized letters → [A-Z]+
(5) White spaces → [ ]+

The regular expressions are stored in the model file in src/main/resources/data/regex_1.model. Some of the regular expressions devised from this pattern are too general, such as "\b[A-Z]\b", and "\b(alpha|beta|gamma … psi|omega)\b". The model eliminate them automatically.

During the annotation process, the **HuskieGeneAnnotator** object loads the pre-trained model, and annotate each sentence with regular expression matching.

Performance

Among the 7676 sentences that has at least one gene, 6294 are randomly chosen to be the training set, and the rest 1382 are testing set. The automatically trained model contains 11184 unique regular expressions.

The performance is measured by precision, recall and F1 score. The model training on 6294-sentence training set got precision of 0.416060, recall of 0.422666 on the 1382-sentence testing set. The F1 score is 0.419337.

Discussion

A HMM based named entity recognizer was also attempted. The idea was to train a two-state HMM where each state represents either the gene or non-gene tag. Observations will be the

entire vocabulary. The attempt was abandoned because the HMM module was written in Python, therefore a lot of glue code needs to be written to let it run inside the Java application through Jython.