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

Yun-Nung (Vivian) Chen
yvchen@cs.cmu.edu

October 17, 2012

1. Preprocessing

Sentence Segmentation

The input is segmented into a lot of sentences, which is implemented by CollectionReader.

Tokenization

Because the original sentence may contain some noises, running a tokenizer can make us consider terms without noises. Here we use Stanford tokenizer to automatically tokenize the sentence, and then we can label these terms later. This is implemented by AnalysisEngine.

2. Approach of Name Entity Recognizer

For name entity tagging task, the better solution is to use supervised approaches. We implement machine learning and rule-based approaches, and integrate their results.

Machine Learning Method

With supervised approaches, we utilize advantages of external training data. Here we use two pre-trained models, one of which is trained on GENETAG corpus (LingPipe) and another is trained from PENNBIOIE corpus (JNET); they all have the name entity taggers in gene domain.

- Hidden Markov Model (HMM) - trained on GENETAG
  With a pre-trained model, we use HMM-based approach (Viterbi) to label name entities.
- Conditional Random Fields (CRFs) - trained on PENNBIOIE
  With a pre-trained model, we use CRFs classifier to label the tokens' types (gene types or others).

I use two released JAVA tools, supporting HMM and CRFs to label gene name entities, which are called LingPipe and JNET respectively.

Rule-Based Method

With some defined rules (ex. Acronym, continuous name entities), we re-label the name entity taggers.

- Acronym
  Because usually name entities are all capital, we detect if all characters are capital and to label the term as name entity. We also filter the lengths of the acronyms from 2 to 5. After evaluation, the performance seems not to be improved when acronyms are labeled.

- Merging
  If there are continuous terms labeled as gene name entities, we merge all terms into a longer name entity. It can make sure name entities with multiple words can be correctly outputed.

3. Data Flow of System

Our system includes three main parts as below.

1. Collection Reader (Input: File, Output: JCAS)
   First we read a file and do sentence segmentation; then we set each sentence as a JCAS object to transfer into the second part.

2. Analysis Engine (Input: JCAS, Output: CAS)
   After receiving the JCAS object, we change it into a sentence, then tokenize it. We call name entity recognizer to label, save each name entity as a JCAS object, and transfer them into the third part.

3. CAS Consumer (Input: CAS, Output: File)
   It receives a lot of JCAS then change it into string tokens then output them with gene tags (also precisely maintain the indexes of name entities).

Evaluation

Our system can support two different machine learning approaches, discriminative and non-discriminative training. It also combines the advantages of two different approaches and integrates the name entity results for output. Our final performance (span needs to fully match) is about 80% on F-measure, where 76% on Precision and 84% on Recall.