FINAL REPORT - TEXTJOINER IMPROVEMENTS

Tejaswinee Sohoni
Net ID: tss995
Email ID: tejaswineesohoni2015@u.northwestern.edu
EECS 349 - Northwestern University

I. Introduction And Overview:

Automatically extracting information from the web is called Web Information Extraction. On demand Web Information Extraction systems allow users to search the web for textual queries, for instance, “Nobel Laureates from Austria”. This is done by specifying the query as a relation. However, such systems have to make a trade-off between precision and recall. Therefore, a new approach has been proposed by researchers in Prof. Downey’s group, in which queries are considered as conjunctions and disjunctions of multiple contexts instead of a single context. This offers high precision as well as high recall. This approach has been implemented in a system called TextJoiner.

However, the existing TextJoiner system does not take into account every mention of a particular entity in a piece of text. It misses out on the mentions of an entity where it is referred to with pronouns or equivalent words. This makes the task of coreference resolution interesting to the TextJoiner system. Coreference resolution means finding all the expressions that refer to the same entity in a text. For instance, in the sentence,

“I would like to spend some time with Mary because only she can answer my questions.”, John said.

I, my and John refer to one entity, and she and Mary refer to another. If the TextJoiner system can understand this difference, it can discover more sentences about the entity being queried, and hence can extract more information out of the text.

As a solution to this task, I performed coreference resolution on text with the help of Stanford's NLP library. I then processed the output to form a dataset which will be explained in detail in the following section. I ran 5 different algorithms, decision tree, random forest, naive bayes, k nearest neighbor, and multilayer perceptron, with 10-fold cross validation, to predict how accurately the Stanford library finds coreferences.

I found from the algorithms that the Stanford library actually works quite well in finding coreferences. Most of the coreferences it finds are correct. It is possible that the library is ineffective in finding all of the coreferences, but of the ones it does find, most are correct. This indicates that precision is quite high, even though I can’t make any conclusions about the recall.

II. Coreference Resolution:

I used the Stanford NLP library for coreference resolution. This library has a specific module for coref resolution. While it is not very easy to figure out its usage, it turns out that this module works quite well once one can figure it out. I used some sentences from a parsed Wikipedia page that is used by the actual TextJoiner system. The article I used was about Chicago. The text I used was:

"Chicago is the third most populous city in the United States . Its metropolitan area is sometimes called Chicagoland. Chicago is the seat of Cook County , a small part of the city extends into DuPage County. It was incorporated as a city in 1837. Chicago is listed as an alpha+ global city. The best-known nicknames for the city include Windy City and Second City. The City of Chicago was the fastest growing city in the world for several decades."

I used the coref module to clean the text, tokenize it, and annotate it. Then I used this to build a parse tree of the current sentence and its corresponding dependency graph. Using this, I built a coreferenced graph of chains, where each chain indicates an entity, and includes all mentions of that entity.

III. Dataset:

I processed the coref output to obtain each of the phrases, i.e. mentions, in the chains. I used these to form a dataset. After further processing the chains, I got the individual words or phrases from each chain, its corresponding sentence number and chain number. From this data, I arranged the phrases into every possible pair they could form, their corresponding sentence and chain numbers, whether they belonged to the same sentence or chain, and if they belonged to the same chain, the size of the chain they belonged to. So, essentially, the features of the dataset I formed were:

Phrase A
Phrase B
Sentence Number of Phrase A
Sentence Number of Phrase B
Chain Number of Phrase A
Chain Number of Phrase B
Whether A and B belong to the same chain
If A and B belong to the same chain, size of the chain
Result (Whether the Stanford library coreferences the pairs correctly or not)

I hand labeled the Result field. Initially for a paragraph of about 6 sentences, I got about 20000 pairs, which was too big a dataset for me to hand label, hence I cut down the text, and shortened some sentences. With this text I used I got about 300 pairs, for which I hand labeled the data with Y if the coreferencing is correct and N if it is not.

IV. Training/Testing:

I used the dataset against 5 different algorithms, all with 10-fold cross validation. They gave me different levels of accuracy. They were:
J48 (Decision Tree)- 90.4762% accuracy
Random Forest - 89.1156% accuracy
IBk (k Nearest Neighbors) - 85.034% accuracy
Naive Bayes - 79.932% accuracy
Multilayer Perceptron - 95.2531% accuracy

I compared the accuracy of these and found that Multilayer Perceptron works best in predicting whether the output obtained using the Stanford coref library is accurate it not.

V. Conclusion:

Overall, I think that the Stanford coref library works quite well for finding coreferences for a text, and I believe the module I have created is ready to be used with TextJoiner so that it can coreference text to improve its performance. The model created using Multilayer Perceptron is most apt for predicting whether the output of the coref is correct or not.

VI. Instructions To Run Code:

1. Include all the jars in the lib folder.

2. Run src/coreferencing/Coreferencing.java to perform coreferencing for the input file “input.json” which gives “coref_dataset.csv”.

3. I have hand labeled this dataset to get “coref_dataset_final.csv”.

4. Run src/coreferencing/Convert.java on “coref_dataset_final.csv” to convert the CSV file to arff format.

5. Run the algorithms mentioned in the report in Weka on the file “coref_dataset_final.arff” using 10-fold cross validation.

VI. References:

TextJoiner System: http://websail-fe.cs.northwestern.edu/textjoiner/
Paper Referred To For TextJoiner: http://www.akbc.ws/2014/submissions/akbc2014_submission_24.pdf
Code: https://github.com/tejaswineesohoni/EECS-349-Project---Coreferencing