CORE NLP Pre Requisites, Implementation & Analysis

Team Members

Shilpa Singh
Ayush Srivastava
Sumeet Mishra
Nikhil B Mankame

A. Pre Requisites:

1. Corpus:

- Development Corpus: Frankenstein or the Modern Prometheus
- Test Corpus: Little Man

2. Java Virtual Machine Memory Requirements

- 64 bit Java Virtual Machine
- Java 1.8
- JVM Arguments -Xmx16g

B. Implementation:

The following steps were taken in order to implement & run the Core NLP

1. Ran the Stanford Core NLP API from the PowerShell using the following java run command:

   java -cp "*" -Xmx16g edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tokenize,ssplit,pos,lemma,ner,parse,dcoref -file "C:\Users\Ayush\Desktop\sem 3\anlp\Assignment 2\frank_part7.txt" -outputFormat json

2. The json file obtained from the above-mentioned command comprised of the following information:

   - Token
   - Lemma
   - Named Entity Recognition
   - Parts of Speech Tag

- Coreferences
- Dependencies between tokens

3. The above-mentioned information was extracted from JSON file via a Java Implementation (FreqAnalysis.java)

C. Analysis

As part of the entire implementation & execution the following observations were made:

- Both Development & the Test Corpus had similar writing and conversation styles as both of them belonged to the early 19th Century.
- Both the above-mentioned corpus had a plentiful amount of dialogs and conversations involving multiple speakers.
- In Order to execute and test the CoreNLP pipeline, both the Development and Test Corpus were divided into multiple parts in order to facilitate the smooth and timely execution confirming to the strategy of Divide & Conquer.
- Inorder to identify the frequency with which one character is talking to another, we have taken the output of the corenlp which is in the form of multiple json files and has sentences in the form of jsonarray.We looped through these sentences and got the tokens belonging to each sentence.Each token has the speaker information in it.We found that there could be 1,2 or more speakers for tokens belonging to a single sentence which denotes a conversation.After that we created a sorted set of all of the speakers for a single sentence sorted by their name and concatenated them to use as a hash key for a global data structure for the entire corpus which consists of 7 different json files.As we started scanning the sentences in the corpus one by one, we updated this global hashmap and incremented the count if we found the same pair of speakers in a sentence as encountered before or created a new hash key if an unseen pair of speakers were found.Finally, we got a hashmap where the keys denotes a unique pair of speakers and value denotes the number of times they were found in a sentence.
- We ran the above frequency analysis on both the development and test corpus and reported the frequency of conversation between a set of speakers.There were some anomaly in the data with respect to the speakers where some of them came out as numbers in the output of core nlp.There were some instances where there was only a single speaker for a sentence.

D. Future Developments

- There were some instances were we couldn't find both the speakers. We intend to rectify it and find both the speakers.
- Using Multithreading we can spawn multiple threads in order to facilitate the faster reading and processing of the input file. This can significantly reduce the execution time required to run the CoreNLP Pipeline.

E. What is provided

The following documents have been submitted as a part of deliverables:

A. FrequencyAnalysis.java
B. Test_corpus_frequency_report.txt & Dev_corpus_frequency_report.txt
C. Spacy Implementation of 'Little Men'.ipynb (The location of the file needs to be changed)
D. Finite State Automata.ipynb
E. Markov Chains.ipynb
F. Part of Speech Tagging.ipynb
G. Working of POS Tagger & the Entropy Model.pdf