@TRUMPCULENT

Final Project

CSE 390 Natural Language Processing

Rahul Agasthya

Varun Goel

Stony Brook University

Stony Brook, NY.

CONTENTS

Introduction

The aim of the project was to study and analyse a language model of a famous personality to convert user provided sentences in the manner, the selected celebrity would express.

After careful assessment, Donald Trump was chosen as the subject of the project. The relative consistency and simplicity of his language in his tweets as compared to other United States 2017 Presidential Candidates like Bernie Sanders, Hillary Clinton, Ted Cruz, etc., made Donald Trump's language more straightforward to model.

Java was used to develop this application, primarily due to our familiarity with the language. We used an external library, `twitter4j` to connect to twitter through Java and extract the tweets of Donald Trump. These tweets were later parsed using our custom built parser to remove any retweets. Carnegie Mellon University has developed a parser namely, `Tweebo Parser`, to generate custom Part of Speech Tags for the tweets that handles non-standard words, as found in the language used Twitter. However, we had to develop another parser to modify the output of Tweebo Parser, to generate word-tag pairs, that would be suitable for the application.

Basically, the user enters a string, other than exit, which is converted to the manner in which it is spoken or expressed by Donald Trump (or, as we call it "Trumpify").

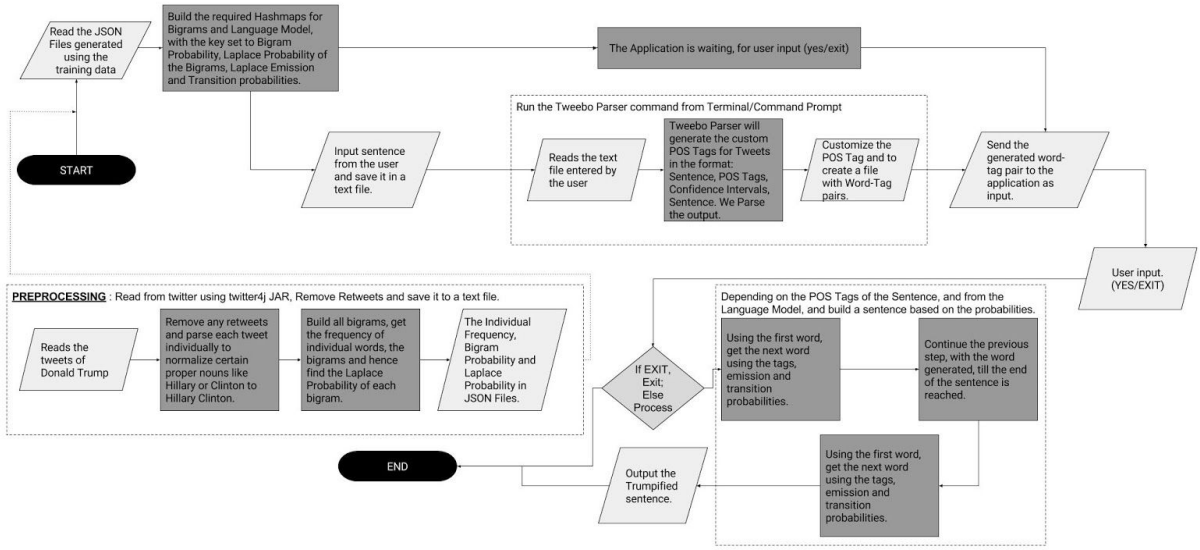
This task enabled us to study and build language models and gain a deeper understanding on Part of Speech tags, and how these tags can be used to analyse

Natural Language. We learnt how to play around with the techniques of NLP and gained an insight into some Machine Learning Techniques like Neural Networks, and how they can be extremely useful to build dependency parse structures in a sentence.

One disclaimer: The developers have nothing against a person or a group of persons or any country or religion, if criticized. It only the way Donald Trump expresses his views on Twitter. We apologise if any sentiments have been hurt.

System Description

The project flow diagram is given below:



The Last page comprises a magnified view of this flow diagram.

Project Details

First, the language model was built using the collected tweets. We decided to use a bigram language model where we gathered data about all the bigrams present and

their respective probabilities. This information was stored in a JSON file, which ensured that the data extraction for the application is efficient.

The information gathered after parsing the Tweebo Parser's output was used to generate Word-Tag pairs, emission probabilities and tag-transition probabilities. This was again stored in multiple JSON files.

In the main application, the user is prompted to give a sentence or type `exit` to exit the application. If the user enters a sentence, the application will write the user's sentence into a text file. The Tweebo Parser is called using command line, to generate the custom Part-of-Speech Tags, and write it to a file.

The user input is then processed using the techniques of Part-of-Speech tagging, where the tag of the first word generates the predicted tags of the rest of the sentence, using the Language Model previously built. There is a method to prepend certain adjectives related to certain nouns, in the sentence. The modified sentence is passed to this method, to ensure that the output is realistic.

Sample Input/Output

Run 1:

```
User Input:
My country is USA and I am Donald Trump

MODIFIED SENTENCE VERSION 1: My party , and the only one Donald Trump
COMMAND TO RUN: ./runTagger.sh examples/your_sentence.txt >
user_tagged_sentence.txt
Do you want to proceed?
yes
```

```
MODIFIED SENTENCE VERSION 2:  My people  .  Mighty  USA  .
@realDonaldTrump  :  Great  Donald  Trump  #Trump2016 will
#MakeAmericaGreatAgain
GIVE AN INPUT (or type exit to quit)
exit
```

Run2:

```
GIVE AN INPUT (or type exit to quit)
The lady is Hillary Clinton

MODIFIED SENTENCE VERSION 1: The Nation is against Hillary Clinton
COMMAND TO RUN: ./runTagger.sh examples/your_sentence.txt >
user_tagged_sentence.txt
Do you want to proceed?
yes

MODIFIED SENTENCE VERSION 2:  My people  ,  goofy  Hillary  Clinton
#CrookedClinton will #BringDownAmerica #CSE390_was_fun!
GIVE AN INPUT (or type exit to quit)
Exit
```

Run3:

```
GIVE AN INPUT (or type exit to quit)
Thank You for the support to vote for Donald Trump

MODIFIED SENTENCE VERSION 1: Thank you ! #Trump2016
#MakeAmericaGreatAgain #Trump2016 #MakeAmericaGreatAgain #Trump2016
@DonaldTrump
COMMAND TO RUN: ./runTagger.sh examples/your_sentence.txt >
user_tagged_sentence.txt
Do you want to proceed?
yes

MODIFIED SENTENCE VERSION 2:  Thank You  USA  is  FACE  CARING
Awesome  Donald  Trump  will #MakeAmericaGreatAgain
GIVE AN INPUT (or type exit to quit)
exit
```

Evaluation

The "*Trumpculent Application*" was a non-standard application, and there were no standard evaluation measures. So, to gain a better assessment of the performance of this application, two methods were built for the sentence conversion and then the output from both the models were compared.

The first method just used the language model built from Trump's tweets and the second model used POS tags as well. We expected the second model to outperform the first one in most of the cases. However, in certain cases, the output from the first model seemed much more realistic than the second one. We believe that this behavior might have been due to the non-standard POS tags that the Tweebo Parser uses and how they differ from the actual English language tags.

We also found that for longer sentence lengths, the first method produced more reasonable outputs.

Conclusion

The project enabled us to study and build language models and gain a deeper understanding on Part of Speech tags, and how these tags can be used to analyse Natural Language. We learnt how to play around with the techniques of NLP, mostly Part of Speech Tags, and gained an insight into some Machine Learning Techniques like Neural Networks, and how they can be extremely useful to build dependency parse structures in a sentence.

The use of Dependency Structures would have enabled better grammar and sentence structuring. We also studied @DeepDrumpf, a Donald Trump twitter bot, that made use of Neural Networks, to build well structured sentences and figured that Neural Networks can be extremely useful in structuring sentences.

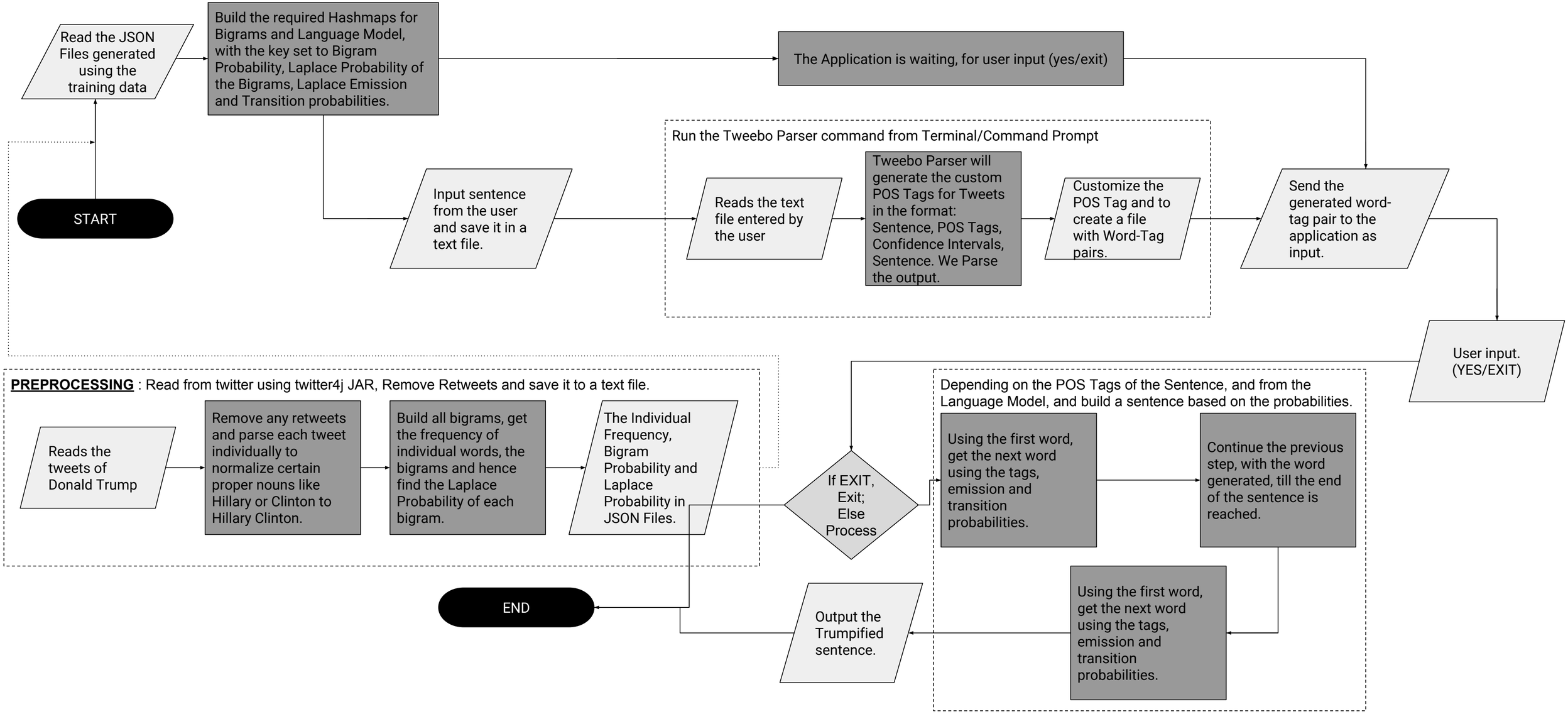
References

Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills,
Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and
Noah A. Smith

In Proceedings of the Annual Meeting of the Association
for Computational Linguistics, companion volume, Portland, OR, June 2011.

http://www.ark.cs.cmu.edu/TweetNLP/gimpel+etal.acl11.pdf


Read the JSON Files generated using the training data
Build the required Hashmaps for Bigrams and Language Model, with the key set to Bigram Probability, Laplace Probability of the Bigrams, Laplace Emission and Transition probabilities.
The Application is waiting, for user input (yes/exit)
START
Input sentence from the user and save it in a text file.
Run the Tweebo Parser command from Terminal/Command Prompt
Reads the text file entered by the user
Tweebo Parser will generate the custom POS Tags for Tweets in the format: Sentence, POS Tags, Confidence Intervals, Sentence. We Parse the output.
Customize the POS Tag and to create a file with Word-Tag pairs.
Send the generated word-tag pair to the application as input.
User input. (YES/EXIT)
PREPROCESSING : Read from twitter using twitter4j JAR, Remove Retweets and save it to a text file.
Reads the tweets of Donald Trump
Remove any retweets and parse each tweet individually to normalize certain proper nouns like Hillary or Clinton to Hillary Clinton.
Build all bigrams, get the frequency of individual words, the bigrams and hence find the Laplace Probability of each bigram.
The Individual Frequency, Bigram Probability and Laplace Probability in JSON Files.
If EXIT, Exit; Else Process
Depending on the POS Tags of the Sentence, and from the Language Model, and build a sentence based on the probabilities.
Using the first word, get the next word using the tags, emission and transition probabilities.
Continue the previous step, with the word generated, till the end of the sentence is reached.
Using the first word, get the next word using the tags, emission and transition probabilities.
Output the Trumpified sentence.
END