Indian Institute of Technology Kanpur
CS671A: Introduction to Natural Language Processing, (Sem II, 2017-18)

ASSIGNMENT 1

Student Name: Prerit Garg
Roll Number: 17111032
Date: February 1, 2018

Problem A (1) : At several places in the text there is conversational text enclosed in single quotes. Single quotes are also used at other places in the text. Your program should identify conver- stational text and replace enclosing single quotes by double quotes.

Approach : As the text contains the instances where sentences are present inside single quotes, but are not the part of conversational text, a simple regular expression could not be applied to do the job.

We need to identify the quotes of conversation text, and also the single quotes that have been used as apostrophe in the text.

By observing the data we can see that quotes that present as the conversational text have a pattern. They are preceded by either a new line character or a space and they are followed by any letter immediately. At the end of conversation text as well they are usually followed by space or newline.

Single quotes are also easy to identify as they lie between 2 letters.

Just like we match the parenthesis, we can match the nested quotes and avoid treating the nested quotes as conversational text.

Problem A (2) : Build a sentence terminator recognizer and tag each sentence in the file test.txt with a beginning of sentence and end of sentence tag.

Approach : We need to ignore all the sentence terminators that lie in a conversational text and identify the sentence terminator at the end of conversational text which marks the end of sentence. There are several places where "." is used as abbreviation. We need to mark them negative as well.

We also need to identify question mark as the end of sentence. We can oberve that question marks that are outside the conversational text are usually sentence terminators. Also if the question mark is followed by a capital letter it is a sentence terminator. "." used in acronyms follow a patters such that it contains sequence of a capital letter and a dot following it. These are not sentence terminators."." used in abbreviations such as 'Dr.','Mr.','Sr.' etc are also not sentence terminators.

We can replace the conversational texts using the unique ids to avoid the "." inside them. Similarly "." used for abbreviations and acronyms can be ignored using a unique identifier for each acronyms and abbreviations. As these 2 cases are taken care of, we are left with the . and ? when they have been used as sentence terminators.

Now we safely split the text on . and ? and ! and place the tags, and then replace back the unique identifiers with the respective strings.

Problem B : In this question you must build a learning based sentence terminator classifier using the context around the terminator character (a punctuation character). You can use any binary classification algorithm available in the Python scikit-learn library (scikit-learn.org). You should be able to do this even if you have not done an ML course. Consult with your TA if you have difficulty. Invent a way to create a feature vector from the context of the punctuation character. Use the output of the sentence tagger in question 1(a)2 as the labelled set. You can use a small part of the tagged data for testing your model. If you want a larger tagged data set run your sentence tagger on file fullTest.txt which is the full text of Chestertons book. Report the accuracy of your model.

Approach : To classify the sentence terminators, I have used a simple Support vector machine model of scikit-learn library. Firstly I tokenized the text, and took the average of the neighboring vectors of "." or "?" or "!". To do this, I used the pretrained glove vectors of words of twitter data (https://nlp.stanford.edu/projects/glove/) with 2 billion tweets, 27 Billion tokens, 1.2 Million vocabulary, 25d, 50d, 100d, 200d vectors.

I created a word-vector dictionary of the vocabulary used in our dataset using the twitter dataset. Identified the ".", "?" and "!" and their respective labels in the text and calculated the average vector of the neighboring words (context vectors). Using these average vectors and the corresponding true label of the sentence terminator, fitted the **SVM** model. There was a huge class imbalance in the data, as the number of "." and "?" and "!" other than being a sentence terminators were quite less in the text. Due to this class imbalance, accuracy was coming close to 97%. So I added a few random sentences with a lot of abbreviations in them so as to obtain somewhat balanced dataset. I tried different window sizes with varying dimension of vectors. Here are a few observations : In the table we have different window sizes across the rows, and different dimensions of word vectors across the columns.

Window \ Dimensions	25	50	100	200
1	0.851	0.851	0.839	0.802
2	0.783	0.796	0.783	0.777
3	0.802	0.790	0.783	0.814
5	0.802	0.790	0.783	0.753