Natural Language Processing – Assignment 3

Handed Out:	6th January 2020
Due Date	24th January 2020, midnight.
Expected deliverables	One compressed electronic file containing the reports and code specified below.
Method of Submission:	Online via Moodle (see below).

Assignment Description

Download the csv file of manually annotated Sinhala and 'Singlish' tweets distributed with this coursework. It consists of a set of 2,500 tweets tagged as either being *hate speech* or *not*.

The overall purpose of this assignment is to train a model that would be useful in monitoring social media content in order to detect possible use of hate speech so that stakeholders could be alerted about it.

You are required to produce Python 3 code to do the following.

(a) Read the csv data file given, into a Pandas *dataframe* and clean the data by (i) removing any remaining irrelevant content such as non-alphanumeric characters and possible extraneous html tags, (ii) tokenizing the text by separating into individual words and (iii) converting the case of the 'Singlish' tokens to obtain unique words. Finally, separate the data into two sets – one for Sinhala and the other for 'Singlish' entries and print the percentage of entries in the two sets. Write the data into two separate csv files for possible separate processing.

(b) Other ways of reducing 'noise' in the Sinhala data set are to (i) remove the so called *stopwords* and (ii) to stem the rest of the words. Print the number of total tokens and the number of unique tokens before this step and after each of the above two steps for the Sinhala dataset. Also print the maximum and minimum sentence lengths of the entries at each stage. Visualize the data using a histogram of the data for different sentence lengths (Hint: use matplotlib).

(c) Create bag-of-words representations of the Sinhala and Singlish datasets separately[1]. Fit logistic regression models[2] to the data in order to be able to predict whether a

[1] Using the *CountVectorizer* transformer in *scikit-learn*.
[2] One of the popular linear models for sparse data available in *scikit-learn*.

post constitutes hate speech or not using 20% of the data for testing. What is the accuracy, precision, recall and f1 measure of this model for the two datasets? Print also the confusion matrix in order to see what kind of errors it is making, and comment on it. Also check the most important words the model uses to discriminate between hate words and others for each of the two datasets[3].

(d) Repeat step (c) treating the full data set (Sinhala and 'Singlish') as one and compare the results with (c). Based on your results, decide whether to process these two data sets together or separately for the following tasks.

(e) One of the issues with the *features* we have used is that they only consider the *frequency* of tokens. A more effective way to deal with the problem of unimportant words that introduce 'noise', is to give tokens an 'importance score' taking into account their inverse document frequency. Implement a logistic regression model to the dataset after transforming it to their *tf-idf* values[4]. In order to test the stability of the predictive power of this model, this time compute the *cross-validation* scores for accuracy, precision, recall and f1 score for it. Also check the most important words the model uses to discriminate between relevant and irrelevant words as before.

(f) Other ways to improve our predictive model include (i) considering context words by building a bag-of-n-grams model[5] and (ii) using semantic embeddings[6] instead of bag-of-word vectors for word representation to model the data. Output the performance of the model you create from each of the above in the most meaningful way and comment on the predictive power of the model.

(g) Interpret the results you obtain for tasks (c), (d), (e) and (f) and comment on the issue of possible over fitting in the models. How could you more effectively use the whole dataset to train a better Sinhala hate speech detector?

Submission

You need to formulate solutions for each of parts (a) through (g) above, clearly explaining your Python code and specifying the outputs produced by the code for the given dataset, in a *Jupyter Notebook* named *IDNumber.ipynb* based on the template given[7]. For each such part, a descriptive summary with an interpretation should be given for the output obtained. Failing to overwrite the placeholder text given in the template will score minus points. You should assume that the original csv data file is in the same directory as your code.

[3] In a logistic regression model, this amounts to simply checking the terms with the highest coefficients.

[4] Scikit-learn's *TfidfVectorizer* class implements support for this. It may be more convenient to create a *sklearn.pipeline* to automate this task (optional). Warning: this training will be fairly time consuming.

[5] This can be done by using the above vectorizers with the additional parameter for n-grams (warning: these models will take time to build).

[6] *Word2vec* is a popular library for this, and can be accessed via the *gensim* python package.

[7] The IDNumber part of the filename should be replaced with your IIT ID number.