**Toxic Comment Classification using Naïve Bayes**

*HW4*

**Harsha Vardhan, Khurdula.**

**CS59000 Natural Language Processing**

**Professor: Jon, Rusert.**

**Task**

Perform and explore ***Naïve Bayes classification***, for a dataset which has comments made by user, to predict if the comment is toxic in nature. The goal is to programmatically implement two methods, *a method to train and return the classifier* and another *method for testing and generating output within a csv file with exact format as a input file*, but with an additional column called ‘*is\_toxic*’ that has a boolean integer flag value i.e. 0 or 1 within it.

**Approach**

First things first, the given data contains a lot of irrelevant characters, which are not very useful to analyze a comment, and would rather act as a medium which leads to improper fitting of the classifier. So I decided to perform some Pre-Processing with Normalization!

**Normalization**

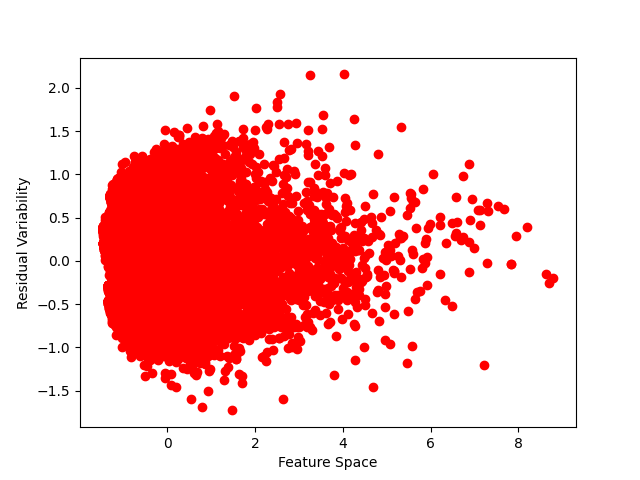
Programmatically, normalization in this task is done for the entire dataset by the class called ‘Normalization’ within the script submitted. The following steps were carried to filter and normalize data:

* Irrelevant columns which are not very helpful have been dropped.
* The features which were dropped are: "*id*", "*obscene*", "*severe\_toxic*", "*threat*", "*insult*", "*identity\_hate*".
* Now one might argue that the above features are very useful in order to classify a comment. But after looking at the data a bit longer, I observed that there is a global Boolean feature ‘*toxic*’ that is 1 or true, for any of the dropped columns where their value is 1. Meaning, *toxic* label is always 1, if any of the above dropped columns had 1 instead of 0. And many comments were simply toxic, and not severely toxic, as well as the fact to be remembered is every *severe\_toxic* comment is *toxic*, but not all toxic comments are *severe\_toxic*.
* All the text has been converted into lower case for case consistency this plays a very important role, while creating probabilities using frequencies by a *CountVectorizer*.
* Only text, has been captured in-order to eliminate special characters.
* And lastly, comments which have no text remaining after performing the above steps have been dropped.
* The ‘***Normalization***’ class also creates the corpus, which is used for training, and testing the classifier.

**Understanding the Data**

After normalization, and before fitting the models was for me to see how far apart the comments are in terms of their toxicity. In-order to achieve this I try to plot a scatter plot for the corpus, instead I get smacked in the face with an error stating, “Failed to allocation 209 gb for plot.” I don’t have such computational resources, the best I can do is run Minecraft with shaders at 80 FPS.

So instead of plotting for the entire *Vectorized sparse matrix*, I reduced the matrix generated by CountVectorizer into 2 component vectors (Two dimensions) i.e. X-axis: “Feature Space” and Y-axis: “Residual Variability”. By plotting this scatter for a sample with just the toxic comments we get the following plot:

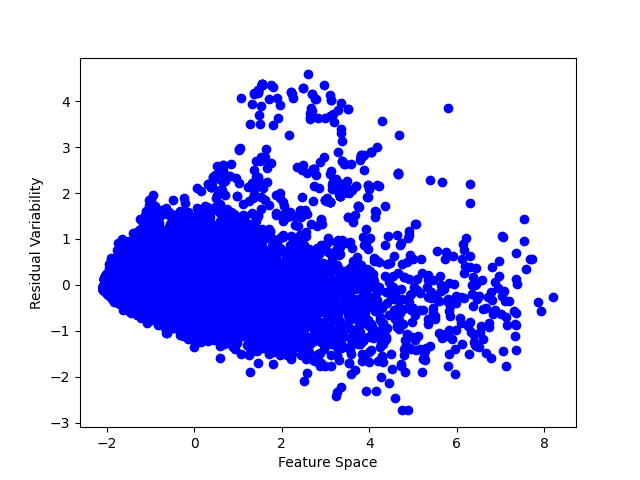


***Scatter plot by taking a sample of toxic comments n = 10000.***

**Note:**

The feature space determines the comment’s feature count, i.e total features created from CountVectorizer, and Residual Variability shows how different these features are for each toxic comment.

Now that we have a plot for toxic comments, only let’s plot a graph for 10000 non toxic comments, we get:



***Scatter plot by taking a sample of non-toxic comments n = 10000***