**Assignment 2 documentation**

During data preprocessing, the following are done:

1. Mentions(@USER) are removed
2. The text is converted to lowercase
3. Stop words. Punctuations, numbers and emojis are removed

**Language model observations**

Used the vocab.lookup method to see if the vocabulary actually contains the words in the training set and the model returned the correct values

The approach used to calculate the MLE scores is as follows: Each test instance was preprocessed and then MLE score was calculated for each word in the test instance. Once this was done, I tried to find the average score of all the words in the test instance to calculate the final MLE for that test instance, but the value was extremely small. I then tried multiplying the MLE scores of these words to form the MLE score of the test instance, but that value would either turn out to be extremely small or 0 (even if one word’s MLE score is 0, the end result would be zero). Hence, I found the sum of the MLE scores of the individual words that make up the test instance and stored that value in the output file.

The average MLE scores for the three models are as below:

Lm\_full: 0.019

Lm\_off: 0.017

Lm\_not: 0.020

The average MLE score of the non-offensive language model is the highest. This could mean that the number of non-offensive tweets in the test set is high and hence the MLE score is high as well.

Results:

Text

Description automatically generated

**Logistic regression observations**

**Hyperparameter modifications**

*C value:*

C helps achieve regularization. C basically tells us to what extent we’re supposed to trust the training data. A high value of C means we give high weight to the training data.

Created a logistic regression model with C=2 and max\_iter=1000. The initial accuracy was 80.58%.

* Decreased the value of C to 1 which decreased the accuracy to 80.11%
* Increased the value of C to 10 which increased the accuracy to 81.16%. So fixed the value of C to be 2
* After 10 irrespective of how much I increased the value of C, the accuracy remained constant

The training data probably did not consist of lot of variations that we see in the real world. Hence increasing the value more than 10 did not make a difference.

*max\_iter:*

Initially set max\_iter to 1000 and C to 10.

Reduced max\_iter value to 1 which decreased the accuracy to 72.09%

Increased max\_iter value to 100 which increased the accuracy to 81.04%

Between 100 and 150 the value of accuracy remained 81.04%

Increased the max\_iter value to 150 which increased the accuracy to 81.16%

Any value after 150, kept the accuracy constant. So fixed the value of max\_iter to 150. Max\_iter is the number of maximum iterations taken for the solvers to converge. This means that it needs minimum 150 as the value of max\_iter to converge.

**Data preprocessing modifications**

Case 1: Preprocessed training and testing data. Accuracy: 81.16%

Case 2: Preprocessed training data and non-preprocessed test data. Accuracy: 80.34%

Case 3: Non-preprocessed training data and preprocessed test data. Accuracy: 80.69%

Case 4: Non-preprocessed training and test data. Accuracy: 80.46%

When C was set to 2 and max\_iter was set to 1000 initially, the accuracy increased from 80.58% to 81.16% when only the test set was preprocessed. This increase could probably be because a few features that contributed to the prediction were discarded during training.

I also tried including the stop words to see if they contribute to improving the accuracy but including stop words actually decreased the accuracy which shows they don’t contribute to predicting better.

**Naïve Bayes observations**

**Data preprocessing modifications**

Case 1: Preprocessed training and testing data. Accuracy: 75.69%

Case 2: Preprocessed training data and non-preprocessed test data. Accuracy: 73.25%

Case 3: Non-preprocessed training data and preprocessed test data. Accuracy: 76.97%

Case 4: Non-preprocessed training and test data. Accuracy: 75.11%

In the case of the Naïve Bayes, the accuracy increased when the training data was not preprocessed. This is mostly because the features contributing to classifying the tweets correctly were removed during preprocessing.

I also tried including the stop words to see if they contribute to improving the accuracy but including stop words actually decreased the accuracy which shows they don’t contribute to predicting better.

The accuracy of Naïve Bayes is less when compared to the Logistic Regression model mostly because the features are not independent, and the dataset is huge. Logistic Regression with regularization performs better than just Logistic Regression.

**Final Comments**

I also tried to add the tweets length as a new column to the dataframe to see if it would help me contribute to better predictions but always ended up getting **ValueError: Found input variables with inconsistent numbers of samples [2, 13240]**. I was not able to figure out how to resolve this. Hence could not test how that affects the accuracy. If you have any idea on how to resolve this error, please let me know. Thank you for the help.

If the code does not run or if you face any other issues the perfectly running code is available in the GitHub repo: <https://github.com/aishwaryaanaidu/offensive-tweets-classification>