Malignant Comments Classifier

Submitted by:

Gourav kumar

INTRODUCTION

The goal is to create a classifier model that can predict if input text is inappropriate (toxic).

1. Explore the dataset to get a better picture of how the labels are distributed, how they correlate with each other, and what defines toxic or clean comments.

2. Create a baseline score with a simple logistic regression classifier.

3. Explore the effectiveness of multiple machine learning approaches and select the best for this problem.

4. Select the best model and tune the parameters to maximize performance.

5. Build a the final model with the best performing algorithm and parameters and test it on a holdout subset of the data.

Analytical Problem Framing

- Multilabel vs Multiclass classification ?

  As the task was to figure out whether the data belongs to zero, one, or more than one categories out of the six listed above, the first step before working on the problem was to distinguish between multi-label and multi-class classification.

  In multi-class classification, we have one basic assumption that our data can belong to only one label out of all the labels we have. For example, a given picture of a fruit may be an apple, orange or guava only and not a combination of these.

- Studying data & identifying hidden patterns
  I had a dataset of 95981 samples of comments along with their labels. I observed that every 1 in 10 samples was toxic, every 1 in 50 samples was obscene and insulting, but the occurrences of sample being severe-toxic, threat and identity hate was extremely rare. The first 5 rows appeared as shown above.

- Next I created some visualisations
  Keeping length of comments on the independent axis with buckets of size 200, I counted the number of comments which had number of characters in that range. For instance, from the graph we can see there are 10000 comments which have 400 to 600 characters.

- Partitioning into testing and training
  Based upon my experience till this point, I thought Scikitlearn's train_test_split would do the magic of dividing the data into testing and

training regardless of its size. Sigh, it didn't happen so and I ended up with an error instead. To solve the issue, I wrote a custom function which would shuffle the indices randomly. So that later we could split our data into training and testing sets without any bias.

Model/s Development and Evaluation

- The model has been trained, tested, and optimized using training and test subsets of the data. I will use an unseen holdout subset of the data to evaluate the model.

- The F1 Score on the holdout data is 0.98.

- Because it's performance is similar to the results obtained in the previous stage, I can confidently say that the model will generalize well to unseen data. Because it is a real world dataset with a huge variety of comments discussing a diverse range of topics and covering situations from informative posts to flame wars, this is probably one of the better scenarios for training a model on text

- he text vectorization strategy using Scikit-Learn's TfidfVectorizer() class makes the model immune to unseen features, as they will be ignored.

CONCLUSION

- This model has 96% accuracy. Now on the surface, that sounds great. But since this is a highly imbalanced dataset, that doesn't mean a lot. In fact, if I had just created a model that predicted "0" for every single item, it would get an accuracy of 90%.

- The real metric of how well the model performed at predicting a toxic comment is recall. This model achieved a recall score of 0.74, which means that it correctly 74% of the actual toxic comments as toxic. That may seem low, but optimizing for recall is a tough challenge. If we used recall as a training objective, it would classify every comment as toxic and quickly reach 100% recall and make every clean comment a false positive. It's necessary then to strike a balance between precision and recall. False positives waste time, while false negatives allow toxicity to fall through the cracks.