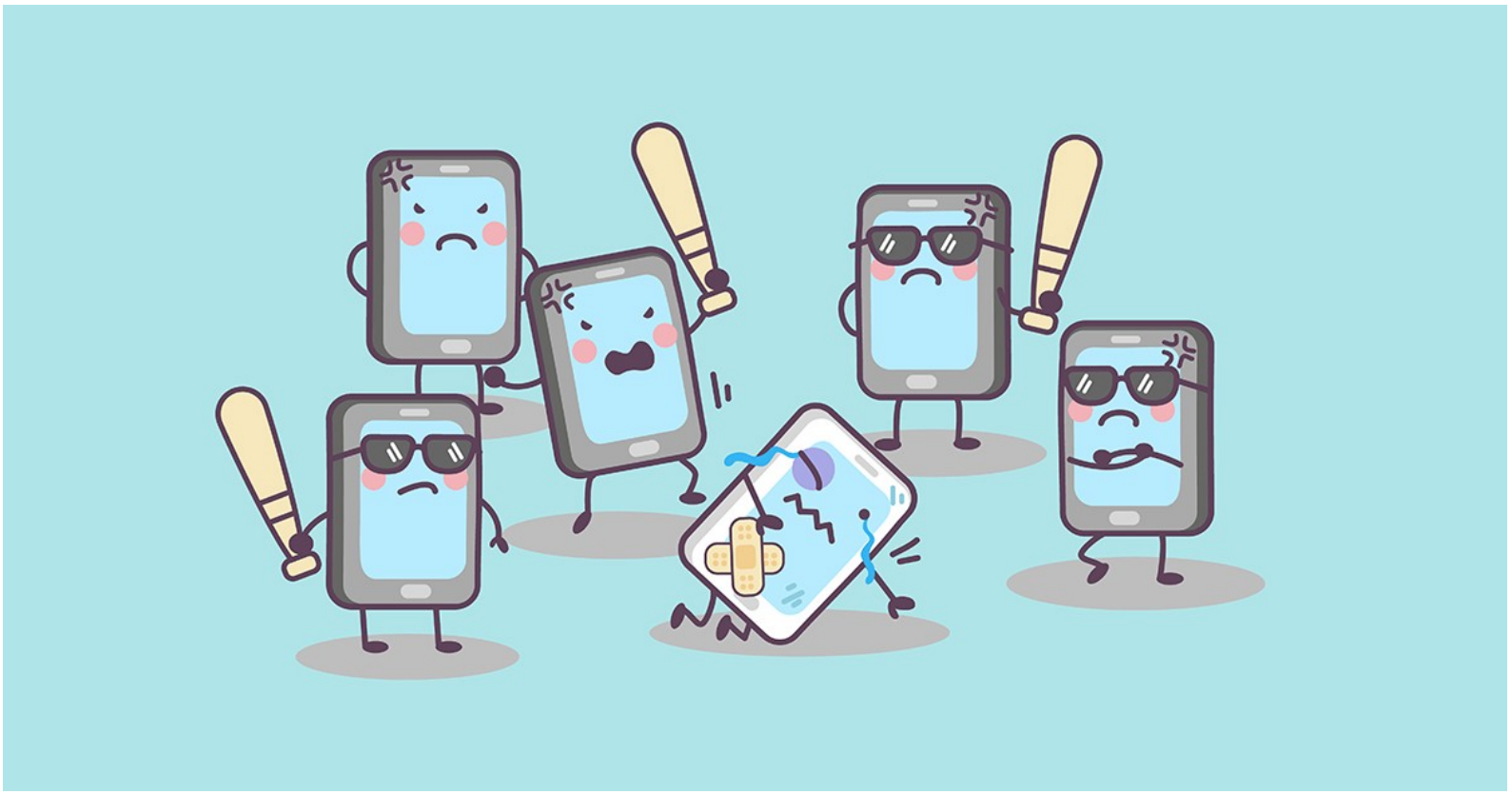
NLP PROJECT

TOXIC COMMENT CLASSIFICATION



Group 6
Sebastián Piscoya
Amishee Choksi
Jimena Germana
Xavier Noval

Introduction

This report will explain our natural language processing tool. The tool is an NLP model that will help detect malicious wikipedia comments and classify them into different categories. Wikipedia is a user-edited page and this makes it vulnerable to negative comments, which has become a problem for them. This tool aims to help solve that problem by quickly identifying these types of comments and restricting this type of behavior.

Business understanding

With the recent growth of people on the internet, we can see that civil conversations are declining. Many online platforms which once encouraged intellectual discussions are forced to close the comment section to avoid problems between users. Toxic comments can be harmful and offensive to individuals or groups, and can lead to negative effects such as online harassment, cyberbullying, and mental distress. For businesses that operate online platforms and communities, the presence of toxic comments can also have negative consequences, such as reduced user engagement, loss of reputation, and legal liabilities.

The benefits of a toxic comment classifier for businesses are significant. By filtering out toxic comments, businesses can create a safer and more welcoming environment for their users, reduce the risk of legal and reputational damage, and improve user engagement and retention. Additionally, the model can help businesses to prioritize and allocate resources more effectively by identifying the most toxic and concerning comments that require immediate attention. Using a NLP tool also reduces the need for manual checking of these comments, freeing up human resources to focus on other, more critical activities.

Description

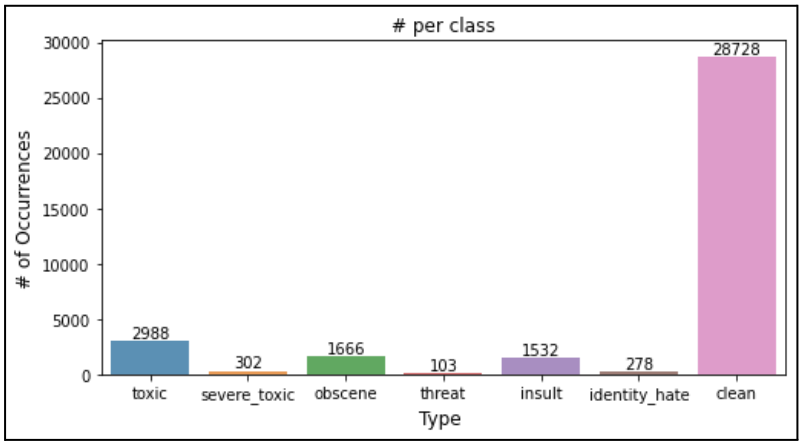
To address this issue, a toxic comment classifier is an NLP model that can automatically detect and flag comments that are likely to be toxic or harmful. By implementing machine learning techniques, the model can learn to identify patterns and features in text that are associated with toxic language. For this, we divided the different ways of toxic comments into 6 categories: toxic, severe toxic, obscene, threat, insult and hate speech. The model will identify if there is any of these categories present in the comment and will input a 1. This means that when a comment only has 0 for all the categories, it is identified as not toxic.

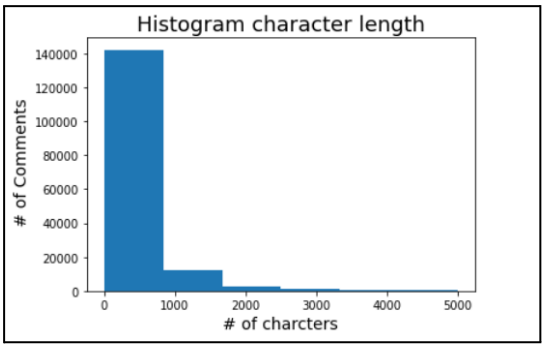
Exploratory Data Analysis

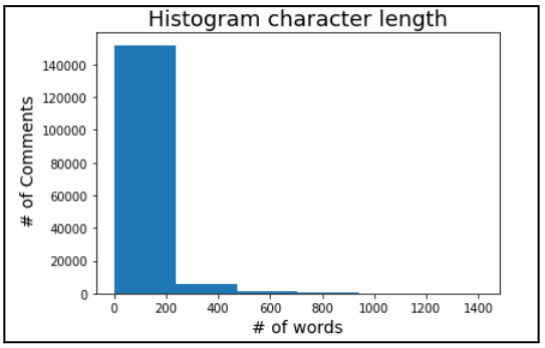
Data exploration

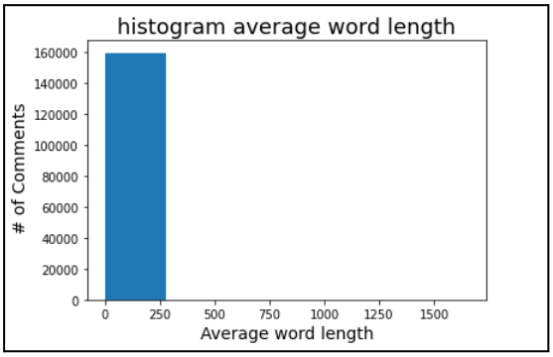
First, it is very important to have a clear understanding of the data we will be working with so we started by reviewing the dataset and its meaning. The dataset comes from Wikipedia and has a total of 31,914 comments.

We analyzed the number of comments per category, and noticed that toxic, obscene and insults had the highest quantity of comments. Looking at average word length and character length, we found that the average number of characters per comment was less than 1000, and most of the comments were less than 200 words in length.







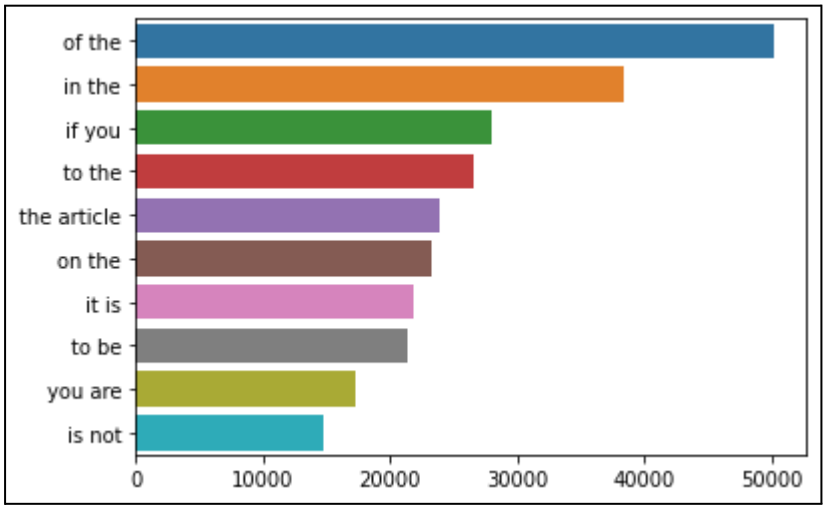


The next step was identifying the most common words in the comments. When visualising the most common words, we noticed that these included words like the, to, of, and, a and is.

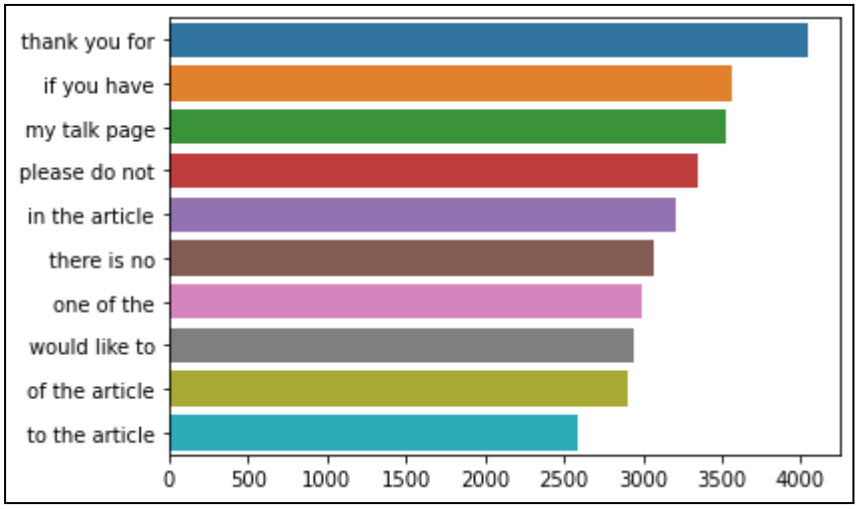
These words by themselves have no meaning, and pose no relationship to the category of comments. To remove the influence of these words, we used the nltk library for stopwords (english) and removed these from the text. This step yielded marginally better results with words like article, page and Wikipedia being more common.

We also looked at ngrams (continuous sequences of words) of length 2 and 3 words, and the most common phrases are as below:

Ngrams, length 2



Ngrams, length 3



The data did not contain any missing values nor outliers.

Data preparation

For the data preparation steps, we performed the following steps:

1. Remove all extra characters and retain only the letters - we removed all the special characters, punctuation marks and other non-alphabetical characters
2. Convert the text to lowercase - to standardize the text, we converted the text to lowercase. This ensured that all comments had the same format and could be used for training and comparison purposes
3. Use the NLTK stop words library to eliminate stop words - stop words are removed from the text as they do not add much valuable information. They contain low-level information and are not relevant to the analysis. By removing the stop words, we can focus more on the important information contained in the text
4. Conduct stemming to reduce data redundancy - the English language has several variants of a single term. The presence of these variances in the text results in repetitive words and leads to data redundancy when developing the model. By reducing the text down to their root form, the model can identify similarities between similar words (like vs likes vs liked) and create a smaller vector set of information

All these steps help to extract the core sentiment from each comment, without needing computationally heavy resources and time.

Modeling and evaluation

All the models that we used could work with a multi-label dataset and give a multi-label output. The original comments were often classified into multiple categories such as toxic, severe toxic and obscene. Given the nature of the data, we needed to use models that could handle this complexity. Similarly, the outputs needed to be able to reflect that a certain comment could be classified into one or more categories. Identifying models that could adapt to this multi-label output was a key factor in choosing the following models.

Model 0: Decision trees

We used a Decision Tree Classifier model to classify comments into multiple categories based on their content based on the given categories. The TF-IDF vectorizer is used to convert the text data into numerical representations that can be used by the model. The maximum depth of the decision tree is set to 2, which limits the complexity of the model.

We used this to create a baseline model as it is a starting point from which we should try to improve later on. This will allow us to get a sense of the difficulty of the problem and the level of performance that can be achieved with a basic approach. Also, by identifying the key challenges and limitations of the problem, it guides us into selecting the appropriate modeling techniques and features.

This model gave very bad results, classifying three of the categories with 0% accuracy, and hence we have opted to discard this model.

Model 1: Random Forest

The second model trains a Random Forest Classifier using TF-IDF vectors for a natural language processing task of multi-label classification. The trained model is then used to make predictions on a test set, and the evaluation report shows the precision, recall, f1-score, and support for each label. The model achieved an accuracy of 0.928 and high precision across all classes, but low recall on severe_toxic, threat comments, and identity_hate. The model successfully identified all labels, unlike the previous Decision Tree model.

Model 2: K-Neighbours

Model 2 is a K-Nearest Neighbors (KNN) classifier that uses TF-IDF to vectorize the text data. The KNN algorithm works by assigning a class label to a new sample based on the majority class label among its K-nearest neighbors. In this case, K is set to 3. The model achieved an accuracy of 0.906, which is lower than the accuracy of Model 1 (a random forest classifier). The precision and recall scores for each label indicate that the KNN classifier performed poorly compared to the random forest classifier. It had low precision and recall across all labels, indicating that the model had difficulty classifying the different types of toxic comments. This model is not performing well and needs improvement.

Model 3: LSTM

Model 3 is a neural network model using LSTM (Long Short-Term Memory) architecture for natural language processing. The model was trained to predict toxic comments using a tokenizer to convert text into numerical sequences and pad the sequences to a fixed length. The model was then split into training and validation sets, and the architecture was defined with an embedding layer, LSTM layer, dropout, flatten, and dense output layer with a softmax activation function. The model was compiled with the Adam optimizer and categorical cross-entropy loss function. The model was trained for one epoch with a batch size of 128. The trained model was saved as a pickled file and loaded into the program using Python's pickle module. No results were provided in the code snippet.

Model 4: Bert Zero Shot Learning

We decided to use BERT as it is a deep learning algorithm that is trained with a large amount of text to understand the context and meaning of words in a sentence. BERT model already has a good performance on many NLP tasks, including sentiment analysis and text classification. For this purpose, the zero shot learning approach was taken as the model already has a high performance on this type of problems.

We gave 6 categories to the model in order to classify the text, where by using transformers the model will be able to process the data in forwards and backwards directions (making it bidirectional). This allows the model to have a better understanding of the meaning and context of words more accurately.

The model outputs probabilities for each of the labels we want to predict, taking the highest ones as the boolean ‘1’. In the train or test sets, for some comments, we have more than one label with a value of 1. For instance, if a comment is labeled ‘severe_toxic’, it would also be labeled as ‘toxic.’ One of the weaknesses for doing zero-shot learning using this model was that we were not able to identify comments that had true values for more than one label as

there were no clear thresholds for which value would be considered as a 1 or 0. Finally, in spite of the model weaknesses, it was interesting to see a high accuracy of 85% for a model that had not even been trained with our data.

Conclusion

Overall, we found that the LSTM model performed the best. This was consistent with what we observed during our testing and validation stages. We have found that the best model for this specific task was LSTM with a 99.2% accuracy on test.

However, there is still room for improvement, particularly with respect to fine-tuning this model for a wide variety of applications. Increasing the complexity and diversity of our dataset will be key to enhancing the models' performance in diverse real-world scenarios.

Next steps

Upon completion of the project, we applied the 4 models to evaluate how each one performed in the same task. The dataset used contained many comments from Wikipedia, which allowed us to compare and contrast the models' performance with a wide variety of comments as this platform has many users and it contains intellectual discussion as well as bad behavior. However, it could be useful to extract data from several platforms such as Twitter or Reddit, which are known for containing aggressive language. In other words, exposing our model to a wider range of comments will improve its performance to ensure it can be applied in many more environments.

Also, it is important to have in mind that this model is only working for the English language. With further work, it is possible to expand it to other languages to make sure any type of harassment online is monitored.

Moreover, the code we have worked on for this project is the backend for the solution, however, we would need to work on the frontend to provide a user friendly interface that can be applied to any case. This tool has a lot of potential growth, as it is known how language can be very difficult to interpret for a machine. Expressions that include sarcasm or slang are very complex to identify, so more in depth studies and a bigger data sample will keep improving the accuracy of the model. Another aspect to consider is the actual application scenario of this model. In cases like social media, language trends change very frequently and hence the model may need to be re-trained more frequently. In other situations, if the adoption of slang is slower, it may be possible to train the model less frequently and still generate fairly accurate results. Finally, it is important to consider the ethical implications of building this type of model because there could be false positive and false negative predictions, so there should be a plan to handle this kind of situation. Flagging / blocking users due to an incorrect prediction may cause problems for companies like Wikipedia that are so dependent on their user base. Thus, they will need to monitor, train and fine-tune these models further to reduce the possibility of false predictions.

References

https://medium.com/@venkateshkandibanda471/toxic-comment-classification-9c748364883b
https://www.kaggle.com/code/jagangupta/stop-the-s-toxic-comments-eda#Wordclouds---Frequent-words: