**Offensive query detection (on Reddit/Twitter dataset) and generalization to multi-lingual setting**

Akash Das 20CS10006

Gaurav Malakar 20CS10029

Atishay Jain 20CS30008

Roopak Priydarshi 20CS30042

1. **Task Description**

The problem is about detecting offensive content in social media posts, and classifying them into different categories such as hateful/ offensive, and clean. The task requires developing an approach to detect and classify the content with high precision and recall, and comparing various models for hate-speech detection. Additionally, the approach should be able to generalize to different languages without the use of large scale Universal Language Models. The overall objective is to develop a method that can effectively filter out offensive content from social media platforms, to ensure a safe and positive environment for users.

1. **Literature Review**

**2.1 Introduction**

Offensive language detection models have been developed to address this issue by automatically classifying text into various categories such as hateful, offensive, and clean. This literature review analyzes the top-performing models in the 2019 and 2020 OffensEval competitions to identify insights for improving more accurate and effective offensive query detection models. Our goal is to identify insights that can help improve the development of more accurate and effective offensive query detection models.

**2.2 OffensEval Competition**

The OffensEval competition was held in 2019 and 2020, with the objective of identifying and classifying offensive language in social media platforms such as Twitter and Reddit. The competition involved three sub-tasks: (1) identifying whether a post contains offensive content, (2) identifying the type of offense, and (3) identifying the target of the offense. The organizers provided a labeled dataset for training and testing models, and the evaluation was performed based on the F1 score, which is a harmonic mean of precision and recall.

**2.3 Methods:**

Several methods have been developed for detecting offensive language in text data. In this section, we provide an overview of some of the commonly used methods.

* CNNs: Convolutional neural networks that can capture local features in text data, and have shown good performance in detecting offensive content in social media datasets (Severyn and Moschitti, 2015; Davidson et al., 2017).
* RNNs: Recurrent neural networks that can model temporal dependencies in a sentence, and have shown good performance in detecting offensive content in social media (Waseem and Hovy, 2016; Park and Fung, 2017).
* SVMs: Support vector machines that can handle large feature sets and sparse data, and have been used for feature selection and classification in detecting offensive language (Wiegand et al., 2018).
* BERT: Bidirectional Encoder Representations from Transformers is a pre-training method for generating contextualized word embeddings that has achieved state-of-the-art performance in various natural language processing tasks, including detecting offensive language in social media datasets (Wulczyn et al., 2017; Huang et al., 2020).
* Ensemble models: Models that combine multiple classifiers to improve overall performance in detecting offensive language, and have been used in various natural language processing tasks, including offensive language detection (Zhang et al., 2018).

1. **Individual Contributions**
2. **Roopak Priydarshi**: Analysed existing models, mainly Offenseval submissions to gauge existing methods and their effectiveness in the task. Collected English-based datasets and Integrated them into a common format to be used for baseline models.
3. **Gaurav Malakar**: Proposed the use of Naive Bayes as the baseline model for the project. Integrated sentiment analysis to improve its accuracy. Collected relevant datasets to train and test the models.
4. **Atishay Jain**:
   1. Read research papers and understood the Naive Baye's model.
   2. Implemented variation 3 and variation 4
5. **Akash Das**:
   1. Implemented variation 1 and variation 2
   2. Diagram for Naive Baye's model used.
6. **Atishay and Akash both**:
   1. Preprocessing using nltk libraries to remove stop words and perform Porter stemming and lemmatization
   2. Train the Naive Bayes classifier, testing and calculating metrics.

**4. Next Plans**

Our current model for detecting offensive messages in social media uses a combination of Naive Bayes and Sentiment Analysis. While this approach has shown promising results in identifying offensive language in English queries, it has a relatively low precision and is limited to only handling English queries.

To address these issues, we plan to incorporate additional NLP models and techniques. Specifically, we aim to explore the use of more advanced techniques such as neural networks and deep learning to improve the precision of the model. These models can be trained on larger and more diverse datasets, allowing us to better handle nuances in language and improve our ability to accurately identify offensive messages.

Additionally, we plan to investigate methods for handling multilingual queries, potentially including techniques such as machine translation or cross-lingual embeddings. While our initial approach may involve a bag-of-words model, we will continue to research and evaluate other methods in order to achieve the best possible results.

**5. Dataset, Metric Used**

The dataset that we have used for this contains more than 10,000 tweets and they are classified into three categories. It contains the columns ‘tweet’ containing the tweet and ‘class’ which marked as 0, 1 and 2. 0 and 1 signify that the tweet is offensive and hateful, 2 signify that the tweet is neutral. The dataset currently contain only English language tweets.

The first step in the analysis was to explore the data to understand the distribution of each variable and any possible correlations between them. Descriptive statistics and visualization techniques, such as histograms and scatter plots, were used to understand the data. We found out that the this dataset contains hate tweets majorly. So, we can say that the data is biased.

Next, the dataset was split into training and testing sets, with a 80:20 ratio. The training set was used to build predictive models, and the testing set was used to evaluate the models' performance. We calculated the accuracy, precision, recall and F1 score to evaluate our model. We are using the naïve bayes algorithm to build the model.

**6. Baselines + Initial Analysis**

1. **Naive Baye's Model:**
   1. Naive Bayes is a probabilistic classifier that uses Bayes' theorem to predict the likelihood of a particular outcome based on input variables. In this case, the input variables are the preprocessed tweets, and the outcome is whether or not they contain hate speech.
   2. The Naive Bayes algorithm assumes that each feature (i.e., word) in the dataset is independent of all other features, which is why it's called "naive." This assumption simplifies the calculations and makes the algorithm fast and efficient.
   3. To train the Naive Bayes model, we first split the preprocessed dataset into training and testing sets. We used the training set to train the model by calculating the probabilities of each feature (i.e., word) occurring in hate speech and non-hate speech tweets.
   4. Once the model was trained, we passed the preprocessed test data through the detection model. The model calculates the likelihood of each test tweet containing hate speech or not based on the probabilities learned during the training phase.
   5. To evaluate the accuracy of the model, we compared the predicted outcomes to the actual outcomes for the test set. We calculated metrics such as precision, recall, and F1-score to evaluate the model's performance.

**7. Experiment:**

1. **1 + log(tf):**
   1. This is a variant of the classic term frequency (tf) scoring method used in information retrieval and text mining. It's designed to give more weight to terms that occur frequently in a document.
   2. The 1 + log(tf) formula calculates the score for each term in the document based on its frequency (tf) in that document.
   3. This method gives more w`eight to terms that occur frequently in a document, but not so frequently that they dominate the score.
2. **(1 + log(tf)) N/log(df):**
   1. This is a variant of the tf-idf (term frequency-inverse document frequency) scoring method. It's designed to give more weight to terms that are rare in the corpus but occur frequently in a particular document.
   2. The (1 + log(tf)) N/log(df) formula calculates the score for each term in the document based on its frequency (tf) and inverse document frequency (idf).
3. **log((1+tf)/((total\_count + len(vocab))):**
   1. This is another variant of the classic term frequency (tf) scoring method used in information retrieval and text mining. It's designed to give more weight to terms that occur frequently in a document.
   2. The score is calculated by taking the logarithm of the ratio of the term frequency (tf) to a smoothing factor that includes the total count of all terms in the document and the vocabulary size (len(vocal)).
   3. The smoothing factor is used to avoid division by zero and to give equal weight to terms that occur infrequently or only once in the document.
   4. Overall, this score calculation method is a variation of the classic tf scoring method that aims to give more weight to important terms in the document, while downweighting terms that are less important. The choice of scoring method may depend on the specific goals of the project and the characteristics of the corpus being analyzed.
4. **log((1 + df) / (total\_doc\_count + len(vocab)):**
   1. The score is calculated for each class (c) in the classification task, based on thetweet/sentence frequency of each term in the training documents of that class.
   2. The score is calculated by taking the logarithm of the probability of the term occurring in that class, given the tweet/sentence frequency of the term in the training documents of that class.
   3. By taking the logarithm of the probability, the score calculation reduces the impact of outliers and balances the contributions of different terms to the overall score.
   4. The log\_prob score for each class is accumulated over all terms in the document.

**Result and Discussion:**

1. Results after 10 fold cross validation for the variation 1
   1. Accuracy: 0.8361124121779862
   2. Precision: 0.8353276670777516
   3. Recall: 0.9993801432611029
   4. F1 Score: 0.9100194362174151
   5. We can see that the recall is very high, which means all hate speech will be detected.
2. Results after 10 fold cross validation for the variation 2
   1. Accuracy: 0.8407962529274003
   2. Precision: 0.8394179367010726
   3. Recall: 0.9991546563375167
   4. F1 Score: 0.9123472668348473
   5. Due to the high recall, we can say that the non-hate classification will be accurate.
3. Results after 10 fold cross validation for the variation 3
   1. Accuracy: 0.9127400468384076
   2. Precision: 0.9318956140700264
   3. Recall: 0.9653144108441424
   4. F1 Score: 0.9483106813173678
   5. The accuracy is very high using the basic naive bayes model.
4. Results after 10 fold cross validation for the variation 4
   1. Accuracy: 0.9071662763466042
   2. Precision: 0.9173098774983341
   3. Recall: 0.9760415199006582
   4. F1 Score: 0.9457647727551592

**References**

1. Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). [*Automated Hate Speech Detection and the Problem of Offensive Language*](https://github.com/t-davidson/hate-speech-and-offensive-language). In Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM)
2. Fortuna, P., Nunes, S., & Bizarro, P. (2012). [*Automated detection of offensive language behavior on social networking sites*](https://libstore.ugent.be/fulltxt/RUG01/001/887/239/RUG01-001887239_2012_0001_AC.pdf). In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12).
3. Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R., ... & Mubarak, H. (2020). SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media ([*OffensEval 2020 Shared Task*](https://sites.google.com/site/offensevalsharedtask/home))
4. Park, S., & Fung, P. (2017). Detecting aggressive tweets with recurrent neural networks. [Social Media in State Politics: Mining Policy Agendas Topics](https://dl.acm.org/doi/10.1145/3110025.3110097)
5. Wiegand, M., Ruppenhofer, J., & Klein, E. (2018). [AllenNLP: A Deep Semantic Natural Language Processing Platform](https://www.aclweb.org/anthology/W18-2501.pdf)
6. Wulczyn, E., Thain, N., & Dixon, L. (2017). [Ex Machina: Personal Attacks Seen at Scale](https://dl.acm.org/doi/10.1145/3038912.3052591). In Proceedings of the 26th International Conference on World Wide Web