CS 410 - Final Report

Team Information

Team Name: Waiting for the Next MP
Captain: Charlene Zhang (yuqianz6)
Team Members: Gregory Znoyko(gznoyko2), Joel Feddes (jfeddes2), Itay Gozalzani(itayg2)

Free Topics: Hate Toxic Comment Detector
Presentation video: https://youtu.be/p_VHhClhnvQ

Overview

Social media platforms offer a myriad of benefits, yet they simultaneously serve as breeding grounds for various online issues, notably the spread of toxic content. The unrestricted liberty for individuals to post and express opinions often escalates into the proliferation of harmful narratives, generally without significant consequences. Recognizing this challenge, recent years have witnessed a surge in research exploring the application of Natural Language Processing (NLP) to identify and mitigate such toxicity in digital communications. Despite these advancements, the practical adoption of these NLP techniques remains limited. Our project addresses this gap by developing a program that utilizes NLP models to flag toxic comments on social media. Leveraging a comprehensive dataset sourced from Kaggle, we experimented with multiple NLP models to discern an effective method for differentiating between toxic and non-toxic comments. Our goal was to achieve a detection accuracy exceeding 80%, a threshold we deemed necessary for the reliable performance of our models. This report details the methodologies employed, the models tested, and the outcomes of our project, contributing to the broader effort to cleanse social media environments of toxic elements.

Implementation

Our project commenced with the acquisition of the dataset via the Kaggle API. Following this, we initiated a pre-processing phase, wherein the dataset's six distinct labels - “toxic,” “severe_toxic,” “obscene,” “threat,” “insult,” and “identity_hate” - were amalgamated into a singular “offensive” label. This simplification facilitated a binary classification approach, streamlining our analysis.

Subsequently, the dataset underwent tokenization in preparation for model training and validation. It is noteworthy that we allocated 80% of the dataset for training purposes and reserved the remaining 20% for validation. The dataset does not provide labels for the test set because it is a competition.

Our exploration encompassed three distinct models: a traditional Support Vector Machine (SVM), a Deep Learning Convolutional Neural Network (CNN), and a DistilBERT transformer model. This selection aimed to juxtapose simpler machine learning techniques against more sophisticated deep learning approaches, thereby ascertaining the most effective model for our objectives. DistilBERT, a streamlined yet potent derivative of the renowned BERT model, was particularly chosen for its balance of efficiency and robust NLP capabilities, aligning well with the resource constraints and high-quality requirements of our project.

While our original proposal contemplated the incorporation of adversarial training, this aspect was ultimately omitted due to time limitations and its inherent complexity. However, we were confident in this decision, considering the proven efficacy of the BERT transformer in handling complex linguistic patterns.

Each model incorporated a feature enabling users to test random statements from the validation set, thereby offering a practical demonstration of the models' capabilities.

Usage

To effectively operate our program, users must first ensure they possess a valid Kaggle API key and a correctly specified path to the Kaggle JSON file. This is a prerequisite for accessing and importing the dataset utilized in our models.

Upon satisfying these initial requirements, users have the flexibility to either execute all the cells in our notebook to run all models or selectively run cells corresponding to a specific model of interest. It is important to note that any user-defined inputs, such as custom comments for testing purposes, can be integrated within the test section of each model. As a further enhancement, a dedicated user input testing cell is available at the bottom of the notebook, featuring a demonstration of the DistilBERT model with a sample toxic comment.

During the early stages of our project, we experimented with TF-IDF and Word2Vec embeddings to gain insights into the data and assess model performance. The TF-IDF embedding was integrated with the SVM model. However, due to time constraints, we were unable to incorporate the Word2Vec embedding.

Conclusion

The outcomes of our experiment with the three models were not only satisfactory but also exceeded our expectations. Our objective was to surpass an accuracy threshold of 80%, and all three models impressively achieved this goal. The SVM model, despite its relative simplicity, demonstrated a commendable accuracy of approximately 89%. The CNN model further elevated this performance, reaching an accuracy of around 95%. However, the standout performer was the DistilBERT transformer model, aligning with our initial predictions, and achieving an outstanding accuracy of approximately 97%. These results significantly exceed our initial benchmarks, bolstering our confidence in the models' capability to effectively identify toxic comments.

The superior performance of the transformer model can be attributed to its advanced design, which is tailored to comprehend the intricacies and contextual nuances of language. This attribute is crucial for text categorization tasks, particularly when dealing with complex datasets like ours. The DistilBERT model, in contrast to the SVM and CNN models, autonomously extracts sophisticated features from the text, which contributes to its heightened accuracy. However, it's important to note that this enhanced performance comes at the cost of increased computational demands and resource requirements. This underscores a vital consideration in the field of machine learning and NLP: the selection of a model must be judiciously balanced with the specific requirements and constraints of the project at hand.

Project Contribution

Yuqianz6: Led the team; Completed proposal, progress, final report; setup coding environment, implemented the data loading, Transformer model, model evaluation and testing code, fixed code for SVM model.

Jfeddes2: Helped to get the project dataset, identified areas of improvement in preprocessing including Implementing, NLTK's word_tokenize, TF-IDF weighting and word embedding; Helped develop presentation slides, provided recording on project background/introduction.

Gznoyko2: Helped write all the reports/proposal; Implemented the deep learning (CNN) model.

Itayg2: Implemented part of the SVM model.