

Toxic Comment Classification Using Classical and Deep Learning Approaches

By - Siddharth, 2022CSB1125

Abstract

In this project, I focused on classifying toxic comments using a combination of classical machine learning models and deep learning approaches, including transformer-based architectures. By leveraging datasets from the Jigsaw Toxic Comment Classification Challenge, I aim to develop models that effectively identify and categorize toxic language to promote healthier online interactions.

Introduction

A. Problem Statement

The rapid growth of online platforms has led to an increase in user-generated content, including comments, reviews, and posts. Unfortunately, many of these platforms struggle to combat toxic language, which can include hate speech, threats, obscenity, and personal attacks. This toxicity not only creates a hostile environment but also has far-reaching societal impacts, such as mental health issues and reduced online engagement.

This project addresses the problem of toxic comment classification, aiming to create models that can accurately detect and categorize toxic language.

B. Objective

The primary objective of this project is to develop a robust multi-label classification system capable of identifying toxic language across six categories:

- Toxic
- Severe Toxic
- Obscene
- Threat
- Insult

- Identity Hate

By employing both classical machine learning algorithms and advanced deep learning models, I aim to evaluate the effectiveness of different approaches in tackling this challenge.

C. Relevance and Societal Impact

Toxic language has a significant negative impact on online communities. By automating the detection and filtering of toxic comments, this project can contribute to:

1. **Creating Safer Online Spaces:** Automated moderation tools can help platforms remove harmful content promptly.
2. **Enhancing User Experience:** Encouraging healthy interactions can foster positive community growth.
3. **Supporting Content Moderators:** Reducing manual effort required for content moderation, allowing human moderators to focus on more nuanced cases.

This work has potential applications in social media platforms, forums, online marketplaces, and more.

Dataset Overview

A. Source

The dataset for this project is derived from the **Jigsaw Toxic Comment Classification Challenge** on Kaggle. It consists of labeled comments collected from various online platforms, enabling multi-label classification across six toxicity categories.

- **Dataset Link:** [Toxic Comment Classification Challenge | Kaggle](#)

B. Description

The dataset includes:

- **Training Data** (train.csv.zip): Contains labeled comments for training the models.
 - **Rows:** 159,571
 - **Columns:** 8 (including id, comment_text, and six target labels)
 - **Target Labels:**
 - Toxic

- Severe Toxic
 - Obscene
 - Threat
 - Insult
 - Identity Hate
- **Test Data** (test.csv.zip): Contains unlabeled comments used for model evaluation.
 - **Rows:** 153,164
 - **Columns:** 2 (id and comment_text)
- **Submission Format** (test_labels.csv.zip): A template for submitting predictions for the test dataset, where each target label initially has values set to either -1 (indicating this record will not be used for evaluation) or 0 (for records concerned with evaluation). This dataset is structured with the same columns as the training data, but only id and placeholder labels are provided.
 - **Rows:** 153,164
 - **Columns:** 7 (id and six target labels)

Each comment can belong to multiple categories, making this a multi-label classification task.

C. Exploration Findings

1. Target Label Distribution

The dataset reveals a significant class imbalance, with most comments classified as non-toxic. The distribution of toxicity labels is as follows:

- **Toxic:** 9.6%
- **Severe Toxic:** 1.0%
- **Obscene:** 5.3%
- **Threat:** 0.3%
- **Insult:** 4.9%
- **Identity Hate:** 0.9%

This imbalance presents a challenge in detecting rarer toxic categories, such as threats and identity hate, which requires careful handling during model training to avoid bias towards the non-toxic class. The label distribution is visualized using bar plots, highlighting that categories like "Threat" and "Identity Hate" are especially rare compared to more frequent categories like "Toxic" and "Obscene."

2. Comment Length Distribution

Comments vary in length, ranging from a few words to a maximum of 1,411 words. Key statistics on comment length include:

- **Mean length:** 67 words
- **Median length:** 36 words
- **90th Percentile:** 152 words
- **95th Percentile:** 230 words

A histogram of comment lengths shows that the majority of comments are relatively short, with 90% containing fewer than 200 words. This informed the decision to set a padding length of 200 tokens in deep learning models, balancing computational efficiency and model performance.

3. Label Correlation Analysis

Correlation analysis using a heatmap reveals varying relationships between toxicity labels:

- There is a **high correlation** between labels such as "Toxic" and "Obscene," and between "Insult" and "Obscene," indicating overlapping characteristics in these categories.
- **Low correlation** is observed between labels like "Threat" and others, suggesting that certain categories, like "Threat," exhibit unique linguistic patterns distinct from other toxic traits.

4. Missing Values

The dataset contains no missing values in key columns, such as `comment_text` and toxicity labels, ensuring a smooth preprocessing pipeline.

D. Data Cleaning and Preprocessing

The dataset was cleaned and processed to make it ready for model training. Here's how it was done:

1. **Setting Up NLTK Tools:** We used some NLTK resources to help with text processing:
 - **WordNet** for converting words to their base forms.
 - **Stopwords** to filter out common words like "and" and "the."
 - **Punkt Tokenizer** to break down text into words.
2. **Text Cleaning:**
 - **Lowercasing:** All text was converted to lowercase for consistency.

- **Removing Punctuation and Numbers:** We took out punctuation, numbers, and non-ASCII characters to keep the text clean.
 - **Filtering Short Words:** Words with fewer than 3 characters were removed, as they're less likely to add meaning.
 - **Lemmatization:** Words were reduced to their base forms (like "running" to "run") to handle variations of the same word.
3. **TF-IDF Vectorization:** We converted the cleaned text into numbers using a TF-IDF vectorizer. This approach helps focus on words that are more informative:
- **Single Words (Unigrams):** We used single words as features.
 - **Removing Stop Words:** Common words that don't add much meaning were removed.
 - **Accent Normalization:** Accents were standardized for consistency.
 - **Frequency Weighting:** Words that appear very frequently were given less importance, highlighting more informative terms.
4. **Transformation for Model Input:** The TF-IDF vectorizer was trained on the training data and then applied to both the training and test data, creating numerical matrices that could be fed into machine learning models.

These steps helped to clean up and transform the text data, making it easier for models to find meaningful patterns in the comments.

Methodology

This section details the implementation pipeline used to build, evaluate, and optimize machine learning and deep learning models for multi-label toxicity classification.

1. Data Preprocessing and Vectorization

The preprocessing pipeline includes text tokenization, punctuation and number removal, stop word filtering, and lemmatization to standardize the text input. This step ensures that the models only focus on meaningful words, reducing noise.

1. **Tokenization:** A custom tokenize function is implemented to convert each comment to lowercase, remove punctuation, numbers, and non-ASCII characters, and lemmatize the words. Words with fewer than three characters are filtered out.
2. **TF-IDF Vectorization:** A `TfidfVectorizer` transforms text data into numerical features. By applying inverse document frequency weighting, frequently occurring terms have reduced impact, enhancing the model's ability to differentiate between comment types.

- **Configuration:** Unigrams are used for tokenization, common English stop words are removed, and accents are normalized. Only terms with a minimum document frequency of 10 are retained.
- 3. **Training and Test Transformation:** The TF-IDF vectorizer is fitted on the training data and then used to transform both the training and test sets.

2. Classical Machine Learning Models

Three classical machine learning classifiers were implemented, each assessed for effectiveness in multi-label classification.

1. Model Initialization:

- **Multinomial Naive Bayes (MNB):** Probabilistic classifier suited for text data.
- **Logistic Regression (LR):** Popular for binary classification and multi-class problems, outputting probabilities using logistic functions.
- **Linear Support Vector Classifier (LinearSVC):** Effective in high-dimensional spaces, using a linear kernel.

2. Cross-Validation:

- A 10-fold cross-validation function, `cross_validation_score`, is defined to evaluate F1 and Recall scores for each model across all labels.
- This provides insight into model consistency and effectiveness across toxicity categories.

3. Model Evaluation and Visualization:

- Performance metrics (F1 score, Recall, and confusion matrices) were calculated for each classifier using the `evaluation_metrics` function.
- Hamming Loss is computed as a holistic indicator of prediction errors across all labels.
- Box plots and bar charts visualize the performance distribution for F1 and Recall scores, aiding in model comparison.

4. Hyperparameter Tuning:

- **GridSearchCV** explores various hyperparameter combinations for each classifier, optimizing based on F1 score.
- The best-performing parameters are selected and applied to each classifier.

5. Prediction and Submission:

- Using the best model configuration, predictions are generated on the test set.
- The outputs are converted to probabilities with a sigmoid function, and the results are saved in a CSV file for submission.

3. BiLSTM Model for Deep Learning Approach

To enhance model performance, a Bidirectional LSTM (BiLSTM) model using pre-trained GloVe embeddings was implemented.

1. GloVe Embedding Preparation:

- Pre-trained GloVe embeddings are loaded, and an embedding matrix is created to map each word to its corresponding vector representation, including special tokens for padding and unknown words.

2. Dataset Class and DataLoader:

- A custom JigsawDataset class tokenizes, numericalizes, and pads each comment for consistent input length.
- Training, validation, and test datasets are prepared, and DataLoaders are configured with shuffling, batching, and parallel loading for efficient model training.

3. Model Architecture:

- **Embedding Layer:** Uses GloVe embeddings for word representation.
- **BiLSTM Layer:** Captures both forward and backward context in comments, ideal for sequential data.
- **Fully Connected Layers:** Classifies the output from BiLSTM using dense layers, with sigmoid activation for multi-label classification.

4. Training and Validation:

- Training and validation steps are defined, using binary cross-entropy loss for multi-label classification.
- Model checkpoints and learning rate tuning are used to improve training efficiency and save the best model.

5. Inference and Submission:

- After training, predictions are generated on the test set, processed with a sigmoid function for probability outputs, and saved in a submission file.

4. DistilBERT Model for Transformer-Based Approach

To explore transformer-based models, DistilBERT was fine-tuned for the toxicity classification task.

1. Data Splitting and Tokenization:

- The dataset is split into training and validation sets, and a DistilBERT tokenizer is used to preprocess each comment into tokens with a maximum sequence length.

2. Model and Dataset Class:

- A custom dataset class is created for DistilBERT, handling tokenized inputs and labels. The DistilBERT model is configured for multi-label classification.
3. **Training and Evaluation:**
- The Trainer class from Hugging Face is used to fine-tune DistilBERT.

Evaluation Metrics

The performance of the models was assessed using the following evaluation metrics:

1. **F1 Score:**
 - Definition: The harmonic mean of precision and recall.
 - Importance: Balances false positives and false negatives, making it a robust metric for imbalanced datasets.
2. **Recall:**
 - Definition: The proportion of true positives identified out of all actual positives.
 - Importance: Critical in scenarios where minimizing false negatives is essential, such as detecting toxic comments.
3. **Hamming Loss:**
 - Definition: The fraction of labels incorrectly predicted across all instances.
 - Importance: Evaluates model performance in multi-label classification tasks by capturing both false positives and false negatives.
4. **Mean Column-wise ROC AUC (for Kaggle Submissions):**
 - Definition: The average of the individual ROC AUCs for each predicted label.
 - Importance: Provides a comprehensive measure of a model's ability to distinguish between classes for each label.
 - Application: Used to evaluate the BiLSTM and DistilBERT models by submitting predictions to the Kaggle competition.

These metrics provided insights into the strengths and weaknesses of each model, guiding the selection of the best-performing approach for the task.

Results

Classical Machine Learning Models

The evaluation of classical machine learning models for toxic comment classification yielded the following results across multiple metrics:

1. **Multinomial Naive Bayes (MNB)**
 - Achieved an **average F1 score of 0.9702** and a **Recall score of 0.9724**.
 - **Hamming Loss**: 0.02697
 - **Training Time**: 0.31 seconds (fastest among the models)
2. **Logistic Regression**
 - Delivered an **average F1 score of 0.9732** and a **Recall score of 0.9742**.
 - **Hamming Loss**: 0.02579 (lowest among the models)
 - **Training Time**: 12.26 seconds (longest among the models)
3. **Linear Support Vector Classifier (LinearSVC)**
 - Produced an **average F1 score of 0.9717** and a **Recall score of 0.9715**.
 - **Hamming Loss**: 0.02849
 - **Training Time**: 5.34 seconds

These metrics indicate that while Logistic Regression achieved the best overall performance in terms of F1 score and Hamming Loss, Multinomial Naive Bayes was notably faster in training time, making it suitable for scenarios where rapid processing is essential.

Hyperparameter Tuning

For optimal performance, hyperparameter tuning was conducted using GridSearchCV. The best parameters for each model were:

- **Logistic Regression**: solver = 'lbfgs', class_weight = None
- **LinearSVC**: C = 1, class_weight = None
- **MultinomialNB**: alpha = 0.1, fit_prior = True

Deep Learning Models

The evaluation of deep learning models was performed by submitting predictions to the Kaggle competition platform, which assessed performance using the mean column-wise ROC AUC score.

1. **DistilBERT Transformer Model**
 - **Private Score**: 0.9791
 - **Public Score**: 0.9789 (highest among all models)

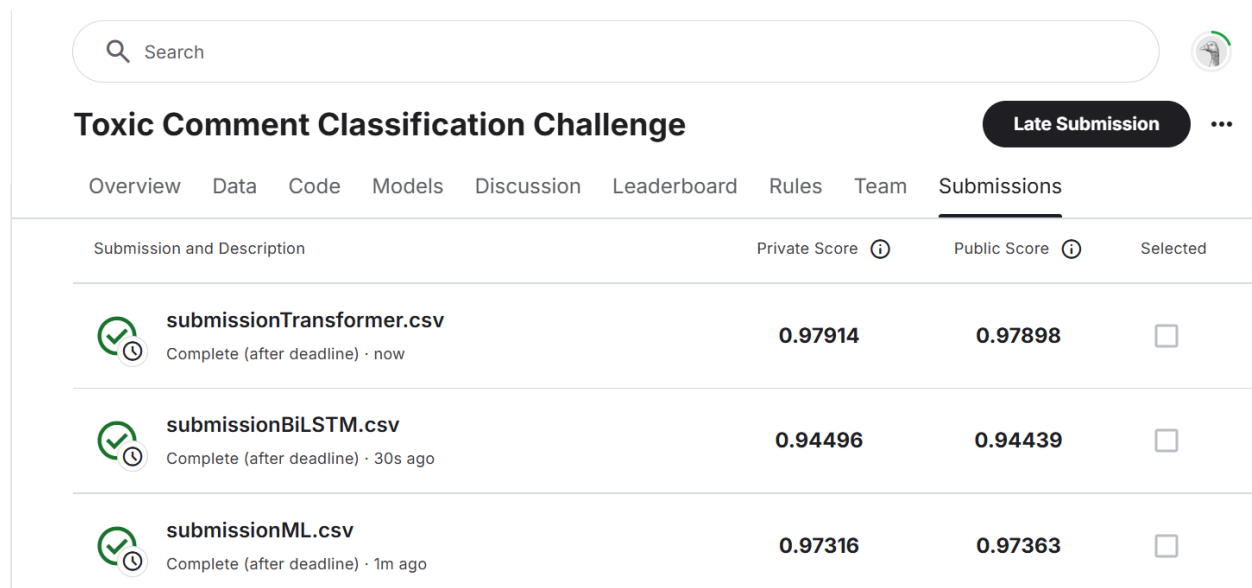
2. BiLSTM Model

- **Private Score:** 0.9449
- **Public Score:** 0.9444




3. Best Classical ML Model (Logistic Regression with tuned parameters)

- **Private Score:** 0.9732
- **Public Score:** 0.9736

These scores demonstrate that the DistilBERT transformer model significantly outperformed both BiLSTM and the classical ensemble, especially in identifying subtle toxic patterns due to its deep contextual understanding of language.



The screenshot shows the Kaggle leaderboard for the Toxic Comment Classification Challenge. At the top, there is a search bar and a 'Late Submission' button. Below the title, navigation tabs include Overview, Data, Code, Models, Discussion, Leaderboard, Rules, Team, and Submissions. The Submissions tab is active, displaying a table with columns: Submission and Description, Private Score, Public Score, and Selected. Three submissions are listed: submissionTransformer.csv (Transformer model, scores 0.97914/0.97898), submissionBiLSTM.csv (BiLSTM model, scores 0.94496/0.94439), and submissionML.csv (Classical ML model, scores 0.97316/0.97363). Each submission is marked as 'Complete (after deadline)'.

Submission and Description	Private Score	Public Score	Selected
 submissionTransformer.csv Complete (after deadline) · now	0.97914	0.97898	<input type="checkbox"/>
 submissionBiLSTM.csv Complete (after deadline) · 30s ago	0.94496	0.94439	<input type="checkbox"/>
 submissionML.csv Complete (after deadline) · 1m ago	0.97316	0.97363	<input type="checkbox"/>

Above is the screenshot of the score achieved on Kaggle from submission created with each of the models.

Conclusion

In this project, we explored multiple approaches to classify toxic comments, comparing classical machine learning models (Multinomial Naive Bayes, Logistic Regression, and LinearSVC) against deep learning models (BiLSTM and Transformer-based DistilBERT). The goal was to develop an effective solution for detecting various types of toxicity in comments, a crucial step in managing online content moderation.

Achievements:

1. **Classical Machine Learning Models:** These models, particularly Logistic Regression and LinearSVC, showed strong performance with higher scores on the Kaggle competition. They benefited from optimized hyperparameters and the use of TF-IDF vectorization, which effectively captured important word-level features.
2. **BiLSTM Model:** Although anticipated to perform well due to its ability to capture sequential context, the BiLSTM underperformed compared to the classical models. This outcome highlights the importance of feature representation and indicates that word-level TF-IDF vectors in classical models captured toxic patterns more effectively than BiLSTM with GloVe embeddings in this setup.
3. **DistilBERT Model:** The transformer-based DistilBERT achieved the highest score, leveraging its attention mechanisms to capture complex word relationships. This model's strong performance underscores the power of transformers in NLP tasks, especially for nuanced classifications like toxicity.

Analysis of Results: The BiLSTM model's relatively lower performance could be attributed to several factors, including limited dataset size for training deep networks, difficulty in capturing specific toxicity signals with simple embeddings, and the model architecture itself. Transformers like DistilBERT inherently have an advantage due to their pre-training on large datasets and ability to focus on relevant context through attention.

Future Work: To improve the results further, several avenues can be explored:

- **Data Augmentation:** Increasing the dataset with synthetic toxic comments could help the BiLSTM model learn better patterns.
- **Hybrid Models:** Combining classical models with deep learning models might capture the strengths of both approaches.
- **Advanced Pretrained Models:** Trying more advanced transformer models, such as BERT or RoBERTa, could yield even better results, especially with fine-tuning on domain-specific data.
- **Explainability Analysis:** Analyzing which words or phrases trigger toxicity predictions in the models could provide insights for model refinement.

In summary, this project demonstrates the comparative strengths of classical and modern NLP approaches for toxicity classification. While classical models delivered competitive performance with simplicity, transformer models provided state-of-the-art results, illustrating their suitability for complex NLP tasks. Future work focusing on data augmentation, model ensembling, and even more sophisticated transformer models could further enhance the detection of nuanced toxic language in online content.