ELEC-E5550 - Statistical Natural Language Processing
Project Report

AUTOMATED ENGLISH TOXICITY DETECTION

April 19, 2024

Erald Shahinas - 906845
Oben Yozgyur - 101725145
Doğa Türkseven - 101729837
Aalto University

Table of Contents

1 Introduction

Social media has seamlessly integrated into our everyday routines. While social media platforms enable extensive connection and self-expression, they also have a downside with the widespread spread of hateful and biased content. To address this issue, there's an increasing demand for automated solutions capable of identifying toxic and harmful posts. In this project, our goal is to develop and compare text-based models that can classify short texts as either toxic or non-toxic based on their content.

When talking about toxicity in written language we must specify what it is meant by toxicity. Toxic language can have different forms and there could be multiple levels of toxicity. [1] According to Davidson et al., the two main forms of toxic language are hate speech and offensive language. The UN defines hate speech as "offensive discourse targeting a group or an individual based on their inherent characteristics (such as race, religion or gender)" [2]. While offensive language is similar to hate speech, as both include offensive terms, the prior is more harmful as it is actively trying to harm the social peace of an individual. Thus when detecting toxic language is important to focus on hateful speech especially.

The automatic detection of online toxicity is a laborious task for many reasons. Firstly, it is not possible to accurately classify toxic speech simply by the presence of harmful terms. [1] Davidson et al. argue that many homophobic and racist slurs have found their way as part of everyday languages in online communities on platforms such as Twitter. While speech containing harmful terms is offensive, it is not always hate speech. In this project we want to be able to classify hate speech, thus simply looking for offensive terms is not enough to build a reliable classifier. [1]

Additionally, hate speech does not always include offensive terms. Davidson et al. argue that it is important to look at the context of a part of speech to classify it more accurately. Other papers argue it is important to look at the syntactic structure of a sentence. [3]

Anther important point is to have accurate and reliable training data for training toxicity classifiers. [1] The training data will first be classified by a human classifier. This process will inject the implicit biases that human classifiers have into the training data and the automatic detector. Thus the training data must be carefully selected before being used.

In this project, we will start by studying existing literature to shape our approach in building language processing models for text classification. Our strategy involves several steps: preprocessing, experimenting with different features, and exploring both custom and pre-trained models.

2 Literature Survey

In today's digital age, people interact more prominently through social media and chat platforms. Microblogging apps allow for instant sharing of thoughts globally. This accessibility, combined with user anonymity and the desire to express opinions and engage in debates, has created a space where aggressive and harmful content can easily spread. [3] The purpose of this literature review is to look at the challenges associated with identifying it, and the suggested techniques for hate speech detection.

The specific definition of hate speech remains problematic, with no universal agreement, and particular aspects of that definition are still being debated and to decide if a statement qualifies as hate speech, its factual accuracy must be verified using external sources. [4] The latter can be another challenge because verifying each statement manually is time-consuming.[6] Cultural background and personal sensitivities can affect whether a particular statement qualifies as hate speech. [5] Another difficulty is the scarcity of publicly available datasets that explicitly categorize hateful, aggressive, and insulting text. [4]

In response to the complex challenges of hate speech recognition, researchers have proposed various techniques. These techniques include both traditional machine learning algorithms and more advanced deep learning models, each with strengths and limitations. An important step for the process is feature extraction, which transforms text data into numerical form that captures significant attributes. The bag-of-words assumption is frequently employed in text categorization, which treats a post as if it were a collection of words or n-grams regardless of the order.[4] Other common feature extraction methods include TF-IDF and word embeddings. [3] TF-IDF [7] can be utilized to give weights to the terms. Word2Vec [8] represents the words in vector space by grouping similar words, whereas GloVe [9] uses global word co-occurrence statistics to learn word embeddings.

In machine learning models, labeled text is used to train a classifier that separates hate speech from normal content. For the text classification tasks, linear classifiers are commonly regarded as strong benchmarks. [13] Support Vector Machines with TF-IDF features, part-of-speech tags, sentiment lexicon is proposed in [1]. On the other hand, [4] proposes Multi-View SVM, where each type of feature is fitted with an individual Linear SVM classifier for hate speech detection. Naive Bayes classifier utilized for classification of racist tweets in [11].

Deep learning models have become increasingly popular for text classification tasks. FastText [13] uses a shallow network for text classification and provides a strong baseline. HARE [14] utilizes large language models (LLMs) for hate speech detection. To recognize hate speech on Twitter, Convolution-GRU [15] combines convolutional neural networks (CNN) and gated recurrent networks (GRU). In their research, [16] experimented with CNN models using word embeddings and n-grams. Three CNN-based models to classify sexist and racist abusive language are proposed in [17]. Ensemble models are also proposed where different models are combined to improve performance. Deep learning-based models explored in [18] and using embeddings learned from deep neural networks along with gradient boosted decision trees gave the best accuracy. Additionally, [19] proposed an ensemble method that aggregates the outputs of ten convolutional neural networks.

3 Methods

This section presents a summary of the methodology used to detect toxicity in text data, including preprocessing techniques, feature engineering methods, model exploration, and evaluation metrics.

3.1 Preprocessing

In the preprocessing stage, we employed a framework that involved tokenization, lowercasing, stopword removal, and lemmatization. Tokenization divides the text into separate words or tokens, whereas lowercasing ensures uniformity by converting all text to lowercase. The frequent words which do not contribute to the semantic context such as “the” or “is” are filtered out by stopword removal. We also utilized lemmatization, a technique that assigns each word, its base or dictionary form.

3.2 Feature Engineering

For feature engineering, we experimented with two different strategies for converting text input into numerical representations. First, we employed TF-IDF which evaluates the significance of words in the dataset. TF-IDF, an abbreviation for Term Frequency-Inverse Document Frequency, assigns a weight to each word considering both its frequency in the document and its scarcity across the entire corpus. Next, we experimented with word embeddings, specifically with Wikipedia2Vec [25], which is a pre-trained model that maps words into high-dimensional vector spaces, capturing semantic relationships between words. Using these feature engineering methods, we tried to extract meaningful representations of textual data.

3.3 Model Exploration

With the obtained numerical representations, we experimented with simple models like SVM, which have shown effective results in classification tasks, as well as deep learning models and ensemble models. We tried different models with the results from TF-IDF and Word2Vec. Additionally, we utilized a pretrained model, Detoxify [21], to obtain toxicity scores and further experimented with ensemble models based on those results with our data.

3.4 Evaluation Metric

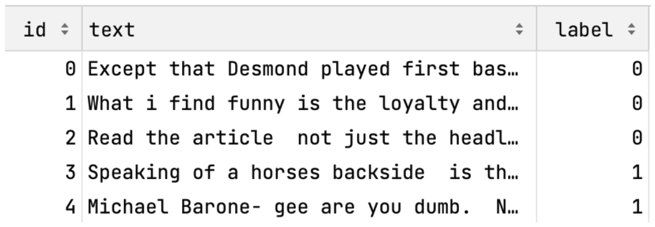
For evaluation, we used F1 score because it considers both false positives and false negatives by balancing precision and recall. In hate speech detection, it is important to correctly identify hateful content and minimize false detections.

4 Experiments

In this section, we explained the details of experimental setup and explored various models and feature engineering techniques for toxicity detection.

4.1 Data

The data is obtained from Kaggle for Toxicity Detection Competiton [29]. There are 99000, 11000 and 12001 rows in the training set, validation set and test set respectively.



id	text	label
0	Except that Desmond played first bas…	0
1	What i find funny is the loyalty and…	0
2	Read the article not just the headl…	0
3	Speaking of a horses backside is th…	1
4	Michael Barone- gee are you dumb. N…	1

Fig. 1. Data Example From Training Set

4.2 Preprocessing

Lemmatization was performed using the Stanza Python package [22], while stopwords were removed using the English stopword corpus from the NLTK library [23].

4.3 TF-IDF Embeddings

To convert the text into a matrix of TF-IDF features, we utilized TfidfVectorizer from scikit-learn library [24]. The TfidfVectorizer was fed preprocessed and lemmatized sentences and outputted a 10000-dimensional vector representation for each sentence. We utilized the resulting matrix to train three distinct models: logistic regression, support vector machines, and random forest classifiers. The F1 scores obtained from these various models are presented in the Results section. For implementing the classifiers, we used the scikit-learn library.

4.4 Wikipedia2Vec Embeddings

Wikipedia2Vec [25] is a tool for learning embeddings of words and entities which is trained with Wikipedia data. In our experiments, we employed pre-trained embeddings for the English language, which map words into 100-dimensional vectors. For each sentence, we computed the average embedding of the words in that sentence. However, our dataset contained words that were not present in the pre-trained model. If none of the words in the sentence were found in the pre-trained model, we assigned them the average embedding obtained from other sentences.

For preprocessing, we experimented with both lemmatization and without lemmatization. We obtained better results without lemmatization, so we included the results accordingly.

Additionally, we tried logistic regression, support vector machines, and random forest classifiers from scikit-learn library with the obtained embeddings. The F1 scores are included in the Results section.

4.5 LSTM

As a deep learning model, we implemented an LSTM (Long Short-Term Memory) model using PyTorch [28]. LSTMs are a type of recurrent neural network (RNN) architecture designed to capture long-range dependencies in sequential data. Text classification frequently requires examining word or character sequences to understand the meaning. In our implementation, the model consists of an embedding layer, an LSTM layer, and a fully connected layer. The embedding layer learns a dense representation of input tokens specified with embedding dimension, mapping each token to a vector space. The embedded sequences are then processed by the LSTM layer, which detects sequential patterns and relationships. The LSTM layer's output is averaged to create a fixed-size representation of the input sequence. Then, this representation goes through a fully connected layer and a sigmoid activation function to produce output probabilities. For all the modules, we used PyTorch. For the optimizer we used Adam and for the loss we used the Binary Cross Entropy loss.

Furthermore, we experimented with integrating TF-IDF and Wikipedia2Vec embeddings, instead of using the embedding layer, into the same model. Unfortunately, this adjustment did not improve the F1 score on the validation set. Therefore, we proceeded with hyperparameter tuning using the embedding layer.

Hyperparameter tuning was applied in order to optimize the results of the LSTM. The parameters that were tuned are: the embedding dimension, the linear layer dimension, the learning rate, adding weight decay, and also adding dropout. Full grid search was unfeasible due to the computational requirements, however, we empirically tuned the hyperparameters that we believed would produce the best results.

4.6 Pretrained Models

- ◇ **Detoxify:** The model is obtained from Hugging Face. [26] and includes trained models and code to predict toxic comments. There are three different models, original, unbiased and multilingual. We employed the original model. The results for each sentence includes the following labels: *toxic, severe_toxic, obscene, threat, insult, identity_hate*, with a score for each of them. We utilized the outputs as features for each sentence in our dataset and experimented with ensemble models.

- ◇ **BERT:** BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained model developed by Google [20]. We focused on text-classification task, and adapted the existing classification code [27] into our dataset. Due to time constraints, we were only able to train for 3 epochs, which is quite short. Nevertheless, the training error was close to zero.

4.7 Ensemble Models

With the output of the Detoxify model including 6 scores of *toxic, severe_toxic, obscene, threat, insult, identity_hate*, we trained several machine learning models to learn one single score that matches with the toxicity predictions on our training set. We evaluated multiple classification models and examined each one with/without PCA. The ensemble voting classifier with Logistic Regression, Random Forest Classifier, AdaBoost Classifier yielded the best result with less overfitting.

5 Results

The table presented below illustrates the F1 scores achieved by various models in our study. The F1 score is a metric that balances precision and recall, allowing a complete evaluation of model performance.

Table 1. Results

Model	**Validation F1**	**Public Test F1**	**Private Test F1**
TF-IDF, SVM	0.896	0.922	0.928
TF-IDF, LR	0.874	0.901	0.902
TF-IDF, RF	0.890	0.921	0.932
W2V, SVM	0.683	0.760	0.841
W2V, LR	0.673	0.726	0.738
W2V, RF	0.708	0.744	0.753
LSTM, tuned, early stopping	**0.906**	**0.938**	**0.946**
LSTM, no tuning, early stopping	0.891	0.919	0.932
LSTM, no tuning	0.858	0.892	0.902
Detoxify - Ensemble	0.884	0.938	0.921
BERT	0.89	0.922	0.940

In order to achieve a cleaner appearance in the table, the abbreviation "W2V" was used instead of "Wikipedia2Vec". The LSTM model yielding the best results is highlighted in bold text.

Hyperparameter tuning was necessary for the LSTM to achieve optimal results. We found that larger embedding dimensions gave the best performance. The LSTM model was quick to overfit after 5-15 epochs thus early stopping was necessary in order to produce the best results. In order to regularize the model we tried both weight decay and using a Dropout layer. The weight decay parameter outperformed both the Dropout layer and the Dropout combined with weight decay. The best results in hyperparameter tuning were obtained with the following parameters: embedding_dim=1000, learning_rate=0.01, weight_decay=0.001.

The training and validation losses and F1 score progress during training for our best model are shown in Figure 2 and Figure 3 respectively.

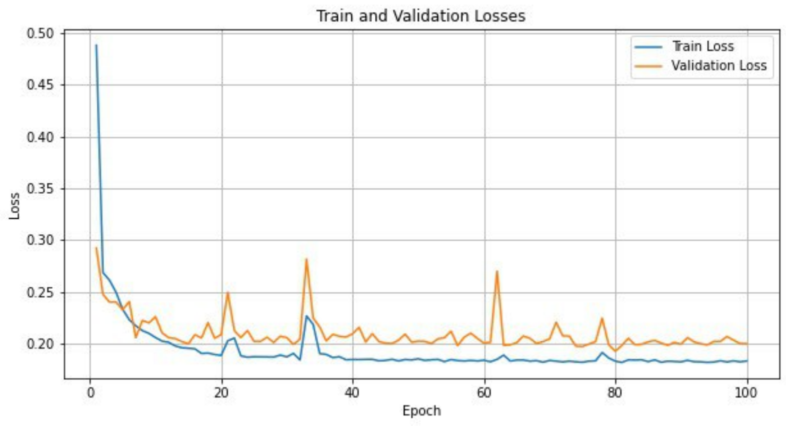


Fig. 2. Train and Validation Losses For LSTM Model

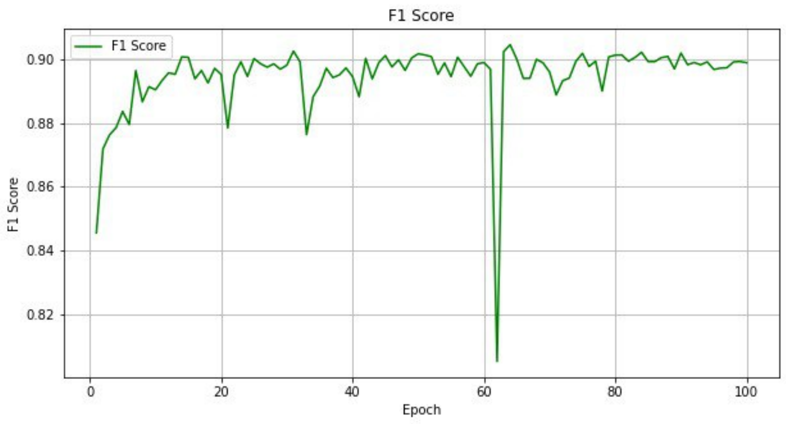


Fig. 3. F1 Score For Validation Set During Training

6 Conclusions / Discussion

Our research focused on the development and testing of text-based models for detecting toxicity in social media content. In summary, our approach included three main steps: preparing the data, trying out different features, and testing both custom and pre-trained models. Our tests showed that both conventional machine learning algorithms and deep learning models can successfully classify text as either toxic or non-toxic. Linear classifiers such as logistic regression and support vector machines performed reasonably well, especially when coupled with TF-IDF embeddings. Despite not being a very complex deep learning model, with the help of hyperparameter tuning, we achieved the best result through LSTM. These results follow quite closely Occam's razor principle, which states that the simplest model that can explain the data is usually the correct one. Interestingly, Wikipedia2Vec produced poorer results than expected, and when we used the embeddings obtained from TF-IDF or Wikipedia2Vec instead of the embedding layer in LSTM, the increase in scores was not observed.

From these results, we concluded that the choice of feature engineering techniques significantly influenced the performance of the classification models and also can be utilized to further improve the models. We found that TF-IDF embeddings effectively captured the importance of words in the dataset, while Wikipedia2Vec resulted in weaker scores. This shows the importance of preprocessing alongside feature engineering. Further improvements could involve more detailed stopword removal or careful handling of punctuation and numbers during sentence preprocessing.

Moving forward, there are several paths for future improvements for the models that we experimented. Although we have experimented with some pre-trained models, we have not achieved the expected results. For instance, further exploration of BERT, particularly focusing on fine-tuning it for our dataset, could be beneficial. The scores that we used as features for ensebmle model from Detoxify could be employed differently. Rather than using Wikipedia2Vec, we could investigate Word2Vec for feature engineering purposes. Another path to look for is performing data analysis on the misclassified data points, to look for wrong assumptions in our model.

7 Division Of Labor

We regularly held group meetings and worked together. Erald focused on tf-idf and LSTM hyperparameter tuning, Oben implemented ensemble models and LSTM, while Doğa worked with Wikipedia2Vec and BERT. In addition, we all worked together on Detoxify. We wrote the report together.

References

[1] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017, May). Automated hate speech detection and the problem of offensive language. In Proceedings of the international AAAI conference on web and social media (Vol. 11, No. 1, pp. 512-515).

[2] United Nations, *Understanding hate speech* https://www.un.org/en/hate-speech/understanding-hate-speech/what-is-hate-speech

[3] Jahan, M. S., & Oussalah, M. (2023). A systematic review of Hate Speech automatic detection using Natural Language Processing. Neurocomputing, 126232.

[4] MacAvaney, S., Yao, H. R., Yang, E., Russell, K., Goharian, N., & Frieder, O. (2019). Hate speech detection: Challenges and solutions. PloS one, 14(8), e0221152.

[5] Ross, B., Rist, M., Carbonell, G., Cabrera, B., Kurowsky, N., & Wojatzki, M. (2017). Measuring the reliability of hate speech annotations: The case of the european refugee crisis. arXiv preprint arXiv:1701.08118.

[6] Popat, K., Mukherjee, S., Yates, A., & Weikum, G. (2018). Declare: Debunking fake news and false claims using evidence-aware deep learning. arXiv preprint arXiv:1809.06416.

[7] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613-620.

[8] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

[9] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[10] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[11] Kwok, I., & Wang, Y. (2013, June). Locate the hate: Detecting tweets against blacks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 27, No. 1, pp. 1621-1622).

[12] https://www.kaggle.com/competitions/aalto-snlp-course-competition-2024/data

[13] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[14] Yang, Y., Kim, J., Kim, Y., Ho, N., Thorne, J., & Yun, S. Y. (2023). Hare: Explainable hate speech detection with step-by-step reasoning. arXiv preprint arXiv:2311.00321.

[15] Zhang, Z., Robinson, D., & Tepper, J. (2018). Detecting hate speech on twitter using a convolution-gru based deep neural network. In The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece,

June 3–7, 2018, Proceedings 15 (pp. 745-760). Springer International Publishing.

[16] Gambäck, B., & Sikdar, U. K. (2017, August). Using convolutional neural networks to classify hate-speech. In Proceedings of the first workshop on abusive language online (pp. 85-90).

[17] Park, J. H., & Fung, P. (2017). One-step and two-step classification for abusive language detection on twitter. arXiv preprint arXiv:1706.01206.

[18] Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April). Deep learning for hate speech detection in tweets. In Proceedings of the 26th international conference on World Wide Web companion (pp. 759-760).

[19] Zimmerman, S., Kruschwitz, U., & Fox, C. (2018, May). Improving hate speech detection with deep learning ensembles. In Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018).

[20] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[21] https://github.com/unitaryai/detoxify

[22] https://stanfordnlp.github.io/stanza/

[23] https://www.nltk.org/

[24] https://scikit-learn.org/stable/

[25] https://wikipedia2vec.github.io/wikipedia2vec/

[26] https://huggingface.co/unitary/toxic-bert

[27] https://github.com/icmpnorequest/Pytorch_BERT_Text_Classification

[28] https://pytorch.org/

[29] https://www.kaggle.com/competitions/aalto-snlp-course-competition-2024/data

8 Appendix

[30] https://github.com/eralds/Automated-English-Toxicity-Detection