Gabriela Palacios
Beau Smit
Antonia Sanhueza

Identifying toxic comments using deep learning

Github: https://github.com/Beau-Smit/toxic-comments

Abstract

The increased use in social media has led to more human interactions, but at the cost of more toxic language. In this paper, we attempt to produce an automated classification system to flag toxic comments in social media using deep learning techniques. We broadly define toxic language as any comment that includes profanity, offensive language, or hate speech. We compared the performance of a Logistic Regression, a Bidirectional LSTM Neural Network, and a Bidirectional Encoder Representations from Transformers (BERT) model. The best model, in terms of F1 score and recall, is BERT, followed by the Bidirectional LSTM and the Logistic Regression. We observe high overfitting in the Bidirectional LSTM model, which we propose to improve by including more training data. Compared to other state-of-the-art classifiers, all our models are more robust in adversarial contexts, where users obfuscate toxic language. We propose more thorough preprocessing to recognize toxic text, such as using spell check.

1. Introduction and objectives

The increased use in social media in recent years has multiplied the number of user interactions through the various platforms. As a byproduct, toxic comments have also multiplied. According to Vice, hate speech has increased by 40% since 2015. Many people consider hate speech as the biggest problem for open forums and social media.

There are several definitions of toxic language in the literature. Davidson et al (2018) define hate speech as *"any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic."* Meanwhile, Gaydhani et al. divide toxic language into two categories: hate speech and offensive language. They define hate speech as Davidson et al (2018) do, and offensive language is defined as the text which uses abusive slurs or derogatory terms. Jigsaw defines toxicity as a rude, disrespectful, or unreasonable comment that is likely to make one leave a discussion. For the purposes of this paper, we will base our classification of toxic comments on Gaydhani et al.'s definition. Thus, all comments including profanity, offensive language, or hate speech will be classified as toxic.

In this paper, we share our implementation of an automated classification system to flag toxic comments in social media. The literature shows a number of attempts using common machine learning models like logistic regression or SVM, while an increasing number of deep learning models are being developed to identify these comments. In particular, we will develop a benchmark model using logistic regression, and then try more sophisticated deep learning models including a bidirectional LSTM model and BERT.

To train our model, we use the Kaggle Toxic Comment Classification Challenge dataset[1] supplemented with data from Davidson et al (2017). The former is a collection of Wikipedia comments. The latter is a series of tweets collected from the Twitter API. We attempt to improve upon previous models by enhancing text preprocessing. Specifically, we will implement spell checking and improve tokenization.

Finally, we measure our results using F1 to balance precision and recall, and focus on recall to check how many truly toxic comments our models predict. We demonstrate the effectiveness of our classifier by showing examples of comments and the resulting label prediction.

2. Literature review

Early literature focuses on traditional machine learning models like Support Vector Machines and Logistic Regression in combination with term frequency-inverse document frequency

[1] https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data

(TF-IDF) and n-grams, while more recent models focus on deep learning approaches (Brassard-Gourdeau and Khoury, 2019).

Gaydhani, A. et al. aim to detect hate speech and offensive text using Twitter data and traditional machine learning models. The classifier model is trained using n-gram and TF-IDF as features and evaluates it for metric scores. Their goal of using TF-IDF is to reduce the effect of less informative tokens that appear very frequently in the data corpus. They consider L1 and L2 (Euclidean) normalization of TFI-DF while performing the experiments, where L1 normalization is defined as:

$$v_{norm} = \frac{v}{|v_1| + |v_2| + \cdots + |v_n|}$$

And L2 normalization is defined as:

$$v_{norm} = \frac{v}{|v_1^2| + |v_2^2| + \cdots + |v_n^2|}$$

Where v_i represents each element of the vector being normalized.

The machine learning classifier models used in this paper are Logistic Regression, Naïve Bayes and Support Vector Machines (SVM). The training of each model was done by performing a grid search for all the combinations of feature parameters and performing 10-fold cross-validation. They analyze the performance of each algorithm based on the average score of the cross-validation for each combination of feature parameters, where the performance of these three algorithms is compared using accuracy. Further, the hyperparameters of two algorithms giving best results are tuned for their respective feature parameters, which gives the best result. 10-fold cross-validation is performed again to measure the results for each combination of hyperparameters for that specific model. The model that results in the highest cross-validation accuracy is evaluated against the test data.

The performance metrics that they used in this paper to measure the model performance are accuracy and recall. The results showed that Logistic Regression performs better with the optimal n-gram range 1 to 3 with the L2 normalization of TF-IDF. Upon evaluating the model on test data, they achieved 95.6% accuracy.

On the other hand, Koratana and Hu (2018) attempt to identify hate speech using the Kaggle dataset of the Toxic Classification Challenge[2], which consists of comments from Wikipedia's talk page edits. They evaluate the models using F1 and test accuracy. The baseline model is a Logistic Regression trained on word embeddings, the input is the average of the word embeddings and the output is the probability of a specific class membership. They also develop

[2] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

two neural network models, a Very Deep Convolutional Neural Network (VDCNN) and a Gated Recurrent Units (GRU) + Long Short Term Memory (LSTM) model. The VDCNN is constructed with maxpooling blocks and no residual connections; it has an initial convolutional layer and two convolutional layers in each subsequent block. On the other hand, the GRU+LSTM model is constructed with attention. The run-based model has a hidden dimension of 500, and two layers. The bidirectional GRU+LSTM takes as input the word embeddings and outputs a sentence embedding, which is then fed by the attention layer into a fully connected classifier. They use pre-trained word embeddings called FasText, which have a dimension of 300. For VCNN they use a batch size of 128, Stochastic Gradient Descent (SGD) with momentum of 0.9 and weight decay of 1.e-4. For GRU+LSTM they use a batch size of 64, and the same momentum and weight decay as VCNN. All model specifications have high accuracy (over 0.96), the F1 score varies between 0.44 and 0.66. The highest F1 and accuracy is from bi-LSTM (Attention + FasText Embeddings) model, and the lowest performance in both measurements is from Logistic Regression.

Hosseini et al. (2017) uncovers potential problems with current models to flag toxic comments. For instance, slight misspellings and inserting punctuation to words can trick the model into lowering the toxicity rating of a given comment.

TABLE I: Demosntration of the Attack on the `Perspective` Toxic Detection System . All phrases in the first column of the table are chosen from the examples provided by the `Perspective` website [1].

Original Phrase (Toxicity Score)	**Modified Phrase (Toxicity Score)**
Climate change is happening and it's not changing in our favor. If you think differently you're an **idiot**. (84%)	Climate change is happening and it's not changing in our favor. If you think differently you're an **idiiot**. (20%)
They're **stupid**, it's getting warmer, we should enjoy it while it lasts (86%)	They're **st.upid**, it's getting warmer, we should enjoy it while it lasts (2%)
They are liberal **idiots** who are **uneducated** (90%)	They are liberal **i.diots** who are **un.educated** (15%)
idiots. backward thinking people. **nationalists**. not accepting facts. susceptible to **lies**. (80%)	**idiiots**. backward thinking people. **nationaalists**. not accepting facts. susceptible to **l.ies**. (17%)
They are **stupid** and **ignorant** with no class (91%)	They are **st.upid** and **ig.norant** with no class (11%)
It's **stupid** and wrong (89%)	It's **stuipd** and wrong (17%)
If they voted for Hilary they are **idiots** (90%)	If they voted for Hilary they are **id.iots** (12%)
Anyone who voted for Trump is a **moron** (80%)	Anyone who voted for Trump is a **mo.ron** (13%)
Screw you trump supporters (79%)	**S c r e w** you trump supporters (17%)

The above table highlights the importance of thorough data cleaning and preprocessing. The authors make a series of suggestions to improve preprocessing. For instance, adversarial training and spell checking can improve the effectiveness of the model. We implement these suggestions in our data cleaning.

3. Data and Methodology

3. 1 Data

There are several datasets available publicly for classifying toxic comments. For this study, we use the following:

1. *Kaggle Dataset for Toxic Comment classification.* This dataset provides over 159,000 Wikipedia comments which have been labeled by humans. Comments were labeled for containing the following types of toxicity:
    a. Toxic
    b. Severe_toxic
    c. Obscene
    d. Threat
    e. Insult
    f. Identity_hate

    Since the model we will use will be binary, any comment that falls into one of these categories would be classified as toxic.
2. Automated Hate Speech Detection and the Problem of Offensive Language (Davidson et al (2017)) dataset. In this paper they built a dataset using the Twitter API and searching for tweets containing terms from the lexicon[3]. Collecting tweets from 33,458 Twitter users and taking a sample of all of their tweets, they collected 25,000 tweets that were manually labeled into three categories: hate speech, offensive but not hate speech, or neither offensive nor hate speech. Since our definition of toxicity includes both hate speech and offensive language, most of the tweets we classify as toxic. Only a small percent of the ~25k tweets contained no hate speech or offensive language, so this data helped balance the Kaggle Dataset, where only 10% of the observations were classified as toxic.

3.2 Preprocessing

Preprocessing includes tokenization for any model. Tokenization has implications for classification as shown in Hosseini et. al. We will address problems with previous classifiers, like Google's Perspective API by enhancing text preprocessing. Specifically, we will implement spell checking and punctuation cleaning. When commenters obfuscate words, such as "id.iot" we still want to identify these words as toxic.

The image below shows examples of original text along with the preprocessed version of the original text.

Original data versus cleaned data

[3] The lexicon was built from words and phases identified by internet users as hate speech and computed by Hatebase.org.

	original_text	text
0	What a fucking day	what a fucking day
1	This is a normal comment.	this is a normal comment
2	They are liberal i.diots who are un.educated	they are liberal idiots who are uneducated
3	I am not ugly, do not say that.	i am not ugly do not say that
4	Anyone who voted for Trump is a mo.ron	anyone who voted for trump is a moron
5	Hey you smell nice. Lovely day, is it not?	hey you smell nice lovely day is it not

This preprocessing allows our model to train on dictionary words, rather than words riddled with tricky punctuation. It is even more useful in the prediction stage. With a robust preprocessor, it is harder to trick the trained model because obfuscated offensive words are changed into words that the model has already seen.

3.3 Models

We split data into train, testing, and validation sets. We used the training set to train the following models, and the test set to evaluate their performance. The validation set will be used in the Bidirectional LSTM Model for selecting hyperparameters.

3.3.1 Model 1: Naive approach

The first and most basic model chooses the most common class, which in this case is “non-toxic”. This simple benchmark provided a useful comparison for the more complex models, which outperformed the naive approach.

3.3.2 Model 2

The second model is a logistic regression with term frequency - inverse document frequency (TF-IDF) to classify toxic comments. The TF-IDF is used to transform text data to numeric data. It compares the relative frequency of each word (or n-gram sequence) in a document to its base-rate frequency in the corpus. Based on these frequencies, the logistic regression learns optimal weights for each n-gram feature. The prediction output is the probability that the comment is toxic given the words that appear in the phrase. Like a bag of words, this method is limited because it only considers the set of words instead of the sequence of words. The following method will attempt to consider the order of the words.

3.3.3 Model 3

For our most complex model, we followed the structure of the most successful models in the literature. The set up consists of a bidirectional LSTM neural network with pretrained embeddings.

Abburi et al. used several word embeddings to feed into the model. In addition, they train an off-the-shelf BERT embedding with unlabeled examples of sexism for their sexism classifier. The following schema shows how input text is transformed into output.

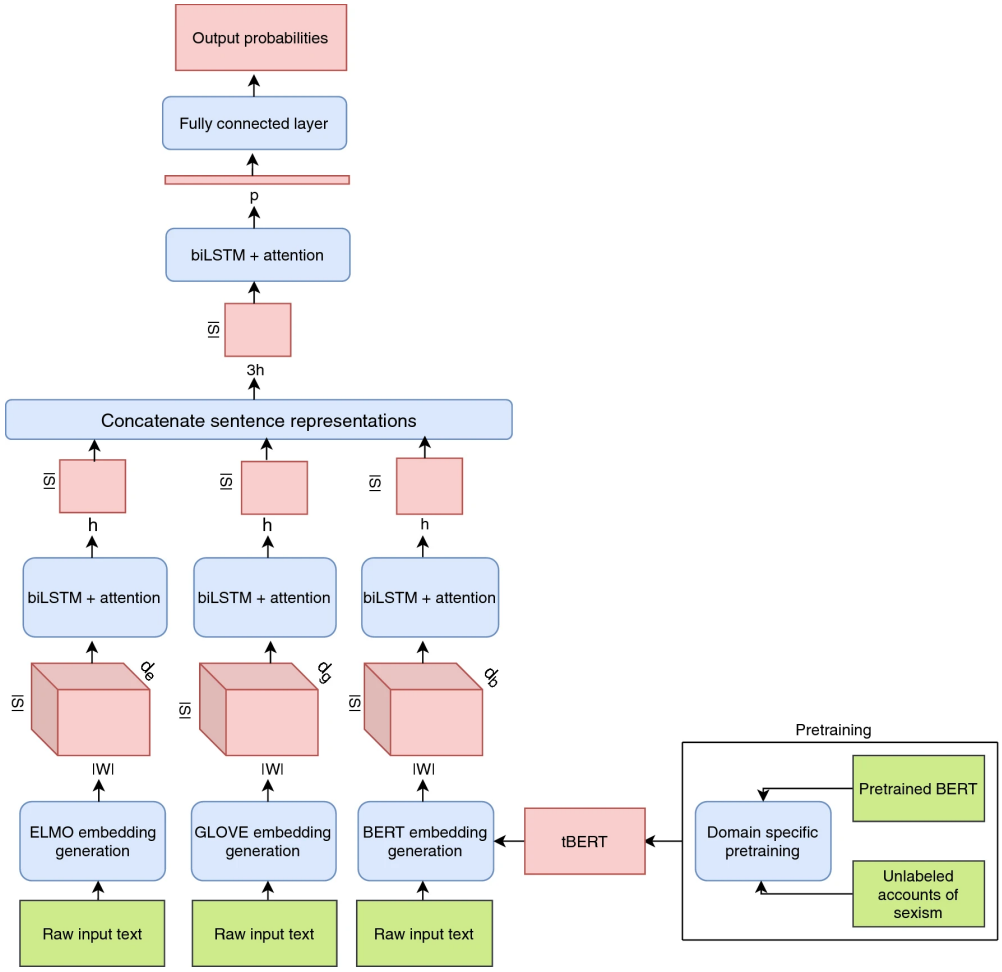
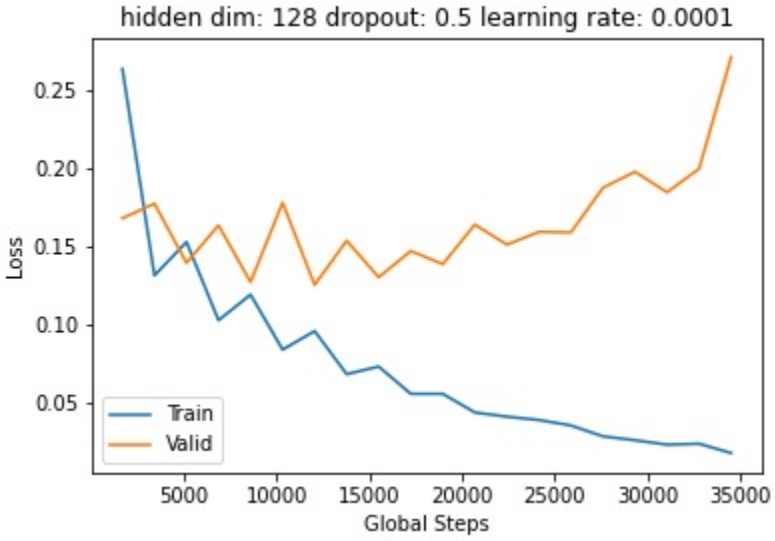


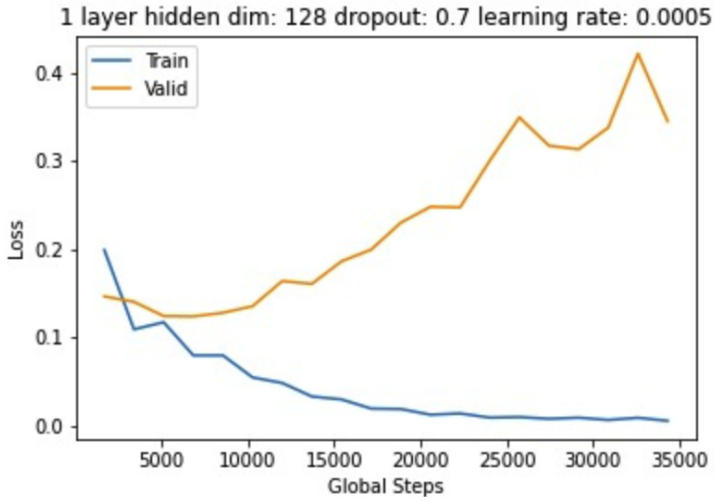
Image Source: Abburi, H., Parikh, P., Chhaya, N. et al. Fine-Grained Multi-label Sexism Classification Using a Semi-Supervised Multi-level Neural Approach. Data Sci. Eng. 6, 359–379 (2021). https://doi.org/10.1007/s41019-021-00168-y

We created our vocabulary using FastText pretrained word embeddings based on our training sample words. To create the batches of our train loader we sorted the training sample by the length of the comment to minimize the amount of padding necessary for each example. This allows the model to concentrate on the relevant information rather than trying to ascribe meaning to the padding.

Bidirectionality allows us to process the text up to a given token and to consider the text after it. In other words, the model learns from both directions. We chose to run 2 layers, and a hidden layer dimension of 128. We trained the model using Binary Cross Entropy loss and an Adam optimizer. To prevent overfitting, we used a dropout rate of 0.5. The graph below shows the loss during training, using training and validation datasets.



The upward trend in the validation set loss indicates overfitting relatively quickly during training. To fix this issue we tried manipulating different parameters, such as reducing the number of layers to 1, increasing dropout, reducing the number of nodes to half (64), using regularization in the optimizer with weight decay, and including a gradient clipping of 1. However, none of these attempts solved the overfitting problem. We decided to move forward with the model with 1 layer, a hidden dimension of 128, dropout rate of 0.7 and learning rate of 0.0005.



The image above shows the training and validation losses with these modifications. The model overfits the training data quicker than the previous model because the learning rate has increased. However, we observe an improvement in the metrics (F1 and recall) produced when applying the model to the testing dataset (shown in the results section). Thus, we chose these set of parameters for our deployment..

3.3.4 Model 4

The fourth model is a fine-tuning BERT model for sequence classification. This technique involves taking a pre–trained BERT model, adding an untrained layer of neurons at the end, and training a new model for the classification task. Some of the advantages of using pre-trained BERT models, in comparison to training a deep learning model, such as Bidirectional LSTM, are: a) quicker development, since the pre-trained model already encodes information about the language, taking less time to train the fine-tuned model b) we can use less data, similar to the latter, the model requires a smaller dataset than when developing a model from scratch, given that it already encodes information about the language, and c) it has shown to achieve better prediction results.

3.4 Evaluation

Our evaluation metrics are F1 and recall. F1 allows us to compare between models, and recall is the most relevant because we want to identify all toxic comments correctly. For benchmarking, we compare our results to those from the literature and the naive model. Please see results section.

3.5 Software

We used Python, Pytorch, and Transformers libraries to program our classification models. To build the machine learning models, we used scikit-learn. To build the deep learning models, we used PyTorch and TorchText. For preprocessing the text, we used NLTK and the SpellChecker library. As a common practice in deep learning for NLP, we used word embeddings (BERT, FastText). The Transformers library allows us to use the BERT tokenizer and pretrained model.

4. Results

4.1 Model 1: Naive approach

The results of the Naive approach are shown in the table below. The model shows a high accuracy as we would expect since labels are imbalanced. Choosing only “non-toxic” does quite well because about 80% of the data are labeled as “non-toxic.” Precision and recall are both 0, since there are no true positive cases. This also produces an F1 score of 0.

Model 1 Scoring Metrics

	Naïve
Accuracy	0.8
Precision	0
Recall	0
F1	0

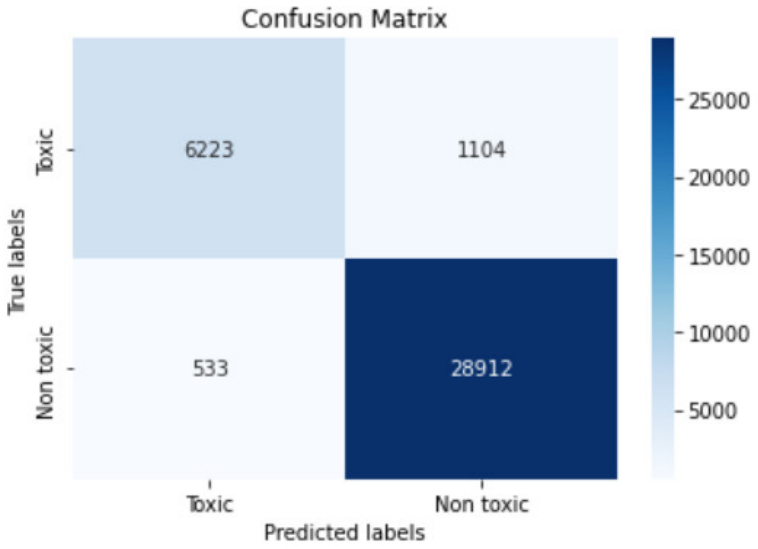
4.2 Model 2: Logistic TF - IDF

The table below shows the results of the Logistic TF-IDF model with preprocessing.[4] The results show an accuracy of 0.95, a recall of 0.85, and a precision of 0.92. The F1 score of these models is 0.88. This model shows significant improvement over our naive model.

Model 2 Scoring Metrics

	Logit
Accuracy	0.95
Precision	0.92
Recall	0.85
F1	0.88

[4] The results of the model without preprocessing are almost the same, and can be found in Appendix X



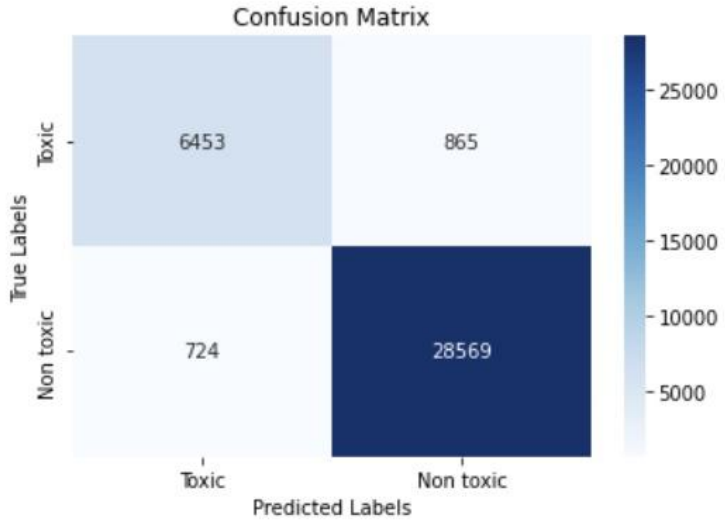
The confusion matrix above indicates that of all comments predicted to be toxic (left hand side), most of them are correctly classified as toxic (top left), with the rest being false positives (bottom left). While the precision is good, the model struggled with a key metric: recall. We have too many false negatives (top right). To combat this issue, we could lower the threshold, which would decrease false negative cases, while increasing false positives.

4.3 Model 3: Bidirectional LSTM

The table below shows the scoring metrics for the bi-LSTM model. This model outperforms the logistic model, but not significantly. Most importantly, the recall increased by 0.3% while maintaining good precision.

Model 3 Scoring Metrics

	bi-LSTM
Accuracy	0.96
Precision	0.90
Recall	0.88
F1	0.89



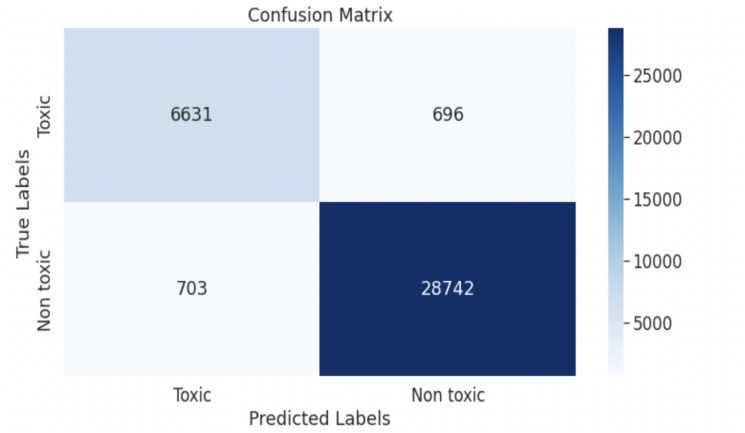
The confusion matrix shows a more even balance between false positives and false negatives.

4.4 Model 4: BERT for sequence classification

The table below shows the scoring metrics for the BERT model for sequence classification. We observe an improvement in recall and F1 over all other models.

Model 4 Scoring Metrics

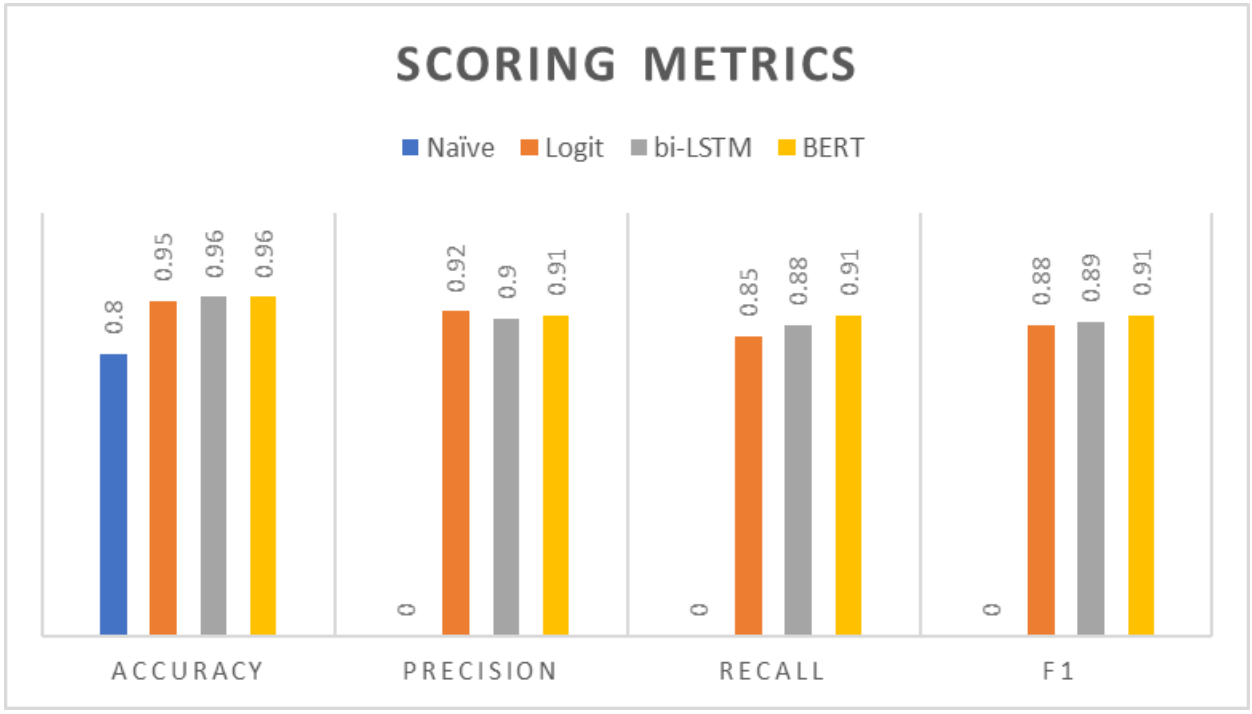
	BERT
Accuracy	0.96
Precision	0.91
Recall	0.91
F1	0.91



The confusion matrix shows that for the first time, false positive cases outnumber the false negative cases

4.5 Comparing the models

The accuracy between the three complex models is similar. Because the data is imbalanced, the Naive approach has a rather good accuracy. Thus, instead of focusing on this evaluation metric, we focus on recall and F1-scores. The naive model scored 0 in both. The graph belows compares the different scoring metrics of our models.



The logistic, bi-LSTM, and BERT models performed similarly. However, focusing on recall and F1 score, we have a clear ranking for our models. In terms of classifying toxic comments, BERT performed the best, followed by Bi-LSTM, followed by logistic, and finally the naive approach. These findings are consistent with the literature as the more sophisticated techniques fare better in complex tasks such as identifying toxic language. Both BERT and Bi-LSTM consider the order of the words, which is one benefit of using these models.

4.5.1 Comparison with literature

Gaydhani et al. (2018) achieve a recall that ranges between 0.93 and 0.98, and a F1 score between 0.94 and 0.97. Koratana and Hu (2018) model's F1 score varies between 0.44 and 0.66. All of our models (without considering the Naïve approach) perform better than Koratana et al, however, our metrics are slightly lower than Gaydhani et al. results. To fully understand why this is the case, we would have to know characteristics about the dataset they are using (size, balance, source).

5. Did the preprocessing help?

In section 3.1 we proposed an innovative way to clean the text that would improve the performance of our model. We compare this with the Google Perspective API[5]

Bi-LSTM model predictions vs Google Perspective

	original_text	text	probability_toxic	GooglePerpective
0	What a fucking day	what a fucking day	0.999956	0.7970
1	This is a normal comment.	this is a normal comment	0.111613	0.0176
2	They are liberal i.diots who are un.educated	they are liberal idiots who are uneducated	0.981001	0.1500
3	I am not ugly, do not say that.	i am not ugly do not say that	0.348495	0.1615
4	Anyone who voted for Trump is a mo.ron	anyone who voted for trump is a moron	0.990556	0.1300
5	Hey you smell nice. Lovely day, is it not?	hey you smell nice lovely day is it not	0.299573	0.2713

The first row shows a comment that includes an offensive word causing both models to assign it a high probability of being toxic. In the second case, a normal comment, both predict a low probability of toxicity. In the third comment, the benefits of performing spell checking on the text becomes evident. Google Perspective API assigns it a low probability of being toxic (0.15)[6], while our model assigns 0.98 probability of toxicity. We can see the same pattern in example 4. Hence, our preprocessing does help the model improve its predictions.

6. Caveats and Future Work

[5] https://perspectiveapi.com/
[6] Google Perspective has recently updated their algorithms to rectify this mistake.

As shown in the results section, our model performs well when compared to the literature, however we observe a high level of overfitting in the training stage, even when including a validation stage and modifying the parameters of the model, such as the learning rate, dropout rate, number of layers, and weight decay.

One potential solution to this problem is including more training data, which would allow the model to learn from a wider variety of toxic and non toxic comments. This can be done as an extension of this study.

Another improvement to our model would be to distinguish between sentences that contain offensive words and toxic comments. In the previous section, we observed that the sentence making reference to having a bad day is classified as a toxic comment by our model and by Google Perspective API. Because of the definition that we used since the beginning, we labeled all data containing a bad word as toxic, which is causing the model to classify sentences that might not fall into the strict definition of toxicity. In effect, we have created a model for detecting swear words. However, toxicity might be more nuanced, requiring a distinction between curse words and personal attacks.

7. Conclusion

In this report, we built several models to classify a comment as toxic or not. We explored traditional machine learning models and deep learning models such as bidirectional LSTM neural networks, and transformers. Our results indicate that BERT and LSTM have high performance based on our evaluation metrics, recall and F1. However, we expected much more performance gain over the logistic regression. BERT achieves an F1 score of 0.91, while LSTM achieves 0.89, and logistic 0.88. Our preprocessing methods generally proved to make more robust predictions on unseen text, even though it may not boost model performance. One key improvement on previous toxic classifiers was the use of spell checking. Commenters often purposefully misspell words, including offensive words. In this adversarial context, our model outperforms other state-of-the-art toxic comment classifiers.

References

Abburi, H., Parikh, P., Chhaya, N. et al. Fine-Grained Multi-label Sexism Classification Using a Semi-Supervised Multi-level Neural Approach. Data Sci. Eng. 6, 359–379 (2021). https://doi.org/10.1007/s41019-021-00168-y

Brassard-Gourdeau, E., & Khoury, R. (2019, August). Subversive toxicity detection using sentiment information. In *Proceedings of the Third Workshop on Abusive Language Online* (pp. 1-10).

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).

Dylan Grosz, Patricia Conde-Cespedes. Automatic Detection of Sexist Statements Commonly Used at the Workplace. Pacific Asian Conference on Knowledge Discovery and Data Mining (PAKDD), Wokshop (Learning Data Representation for Clustering) LDRC, May 2020, Singapour, Singapore. Ffhal-02573576f

Gaydhani, A., Doma, V., Kendre, S., & Bhagwat, L. (2018). Detecting hate speech and offensive language on twitter using machine learning: An n-gram and TF-IDF based approach. *arXiv preprint arXiv:1809.08651*.

Hosseini, H., Kannan, S., Zhang, B., & Poovendran, R. (2017). Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Koratana, A., & Hu, K. (2018). Toxic Speech Detection. *URL: https://web. stanford. edu/class/archive/cs/cs224n/cs224n*, *1194*.

Appendix

Appendix 1: Logistic model results without preprocessing

	Precision	Recall	F1-Score	Support
1	0.92	0.85	0.88	7,327
accuracy			0.95	36,772
macro avg	0.94	0.91	0.93	36,772
weighted avg	0.95	0.95	0.95	36,772