Anthony Konas

CSPB 3832

11/30/25

Toxic Comment Filtering: Final Report

Abstract

My project consisted of building two different toxic comment filtering models. The first was a BERT-based pre-trained transformer model and the second was a Word2Vec + LSTM custom built model. Both models have their benefits. BERT already understands language, so the idea is to train it so that it understands what "toxic" actually means. On the other hand, when making a W2V + LSTM model, you're training this from the ground up and teaching it to understand language as well as what "toxic" means in the process.

BERT takes a while to train, which is a major downside, but ultimately it becomes more accurate and is able to consistently label toxic comments. The W2V + LSTM model takes much less time to train and can still produce great results, but it isn't as consistent with detecting toxic comments. Both models achieved over 96% accuracy on the Jigsaw Toxic Comment dataset, with BERT reaching 96.95% and Word2Vec + LSTM reaching 96.23%. The main difference between the two approaches is that BERT required less manual setup work but needed more computational resources and training time, while the LSTM approach gave me more control over the model architecture and trained faster but required building everything from scratch.

Introduction

Toxic comments exist in abundance. To name a few, you'll see toxic comments all over social media platforms, public forum pages, product reviews, and even restaurant reviews. Regardless of how people think, these comments aren't always the most productive or the best use of language to keep websites a safe space for people to communicate their thoughts and feelings about a particular topic. This isn't saying we should be censoring everything that is said online, but people need to learn that there's a productive way to state their feelings and a less productive way that only encourages negative behavior online.

Recent advancements in language models have improved the ability to detect hate speech online. In particular, Large Language Models like transformer based models like BERT have become the standard in the industry to train and detect online hate speech.

The goal of my project is to compare two different approaches for toxic comment detection: a state-of-the-art pre-trained BERT model versus a custom-built Word2Vec + LSTM model. I want to see if the simpler, custom approach can compete with the fancy pre-trained transformer, and what the trade-offs are between the two in terms of accuracy, training time, and ease of implementation.

Related Work

During my time working on this project, I read a handful of studies exploring toxic comment classification using natural language processing techniques. Two studies in particular influenced my decision making. Pawar, Garud, Kadam, and Khairnar demonstrated that preprocessing steps such as HTML removal, URL filtering, and text normalization significantly improved model performance across different architectures. Zaheri, Leath, and Stroud compared traditional machine learning methods with neural networks on the Jigsaw dataset, finding that deep learning approaches better handled nuanced cases of toxicity.

Data

I used Kaggle's Jigsaw Toxic Comment Classification dataset, which has 160,000 Wikipedia comments labeled as toxic or non-toxic. I cleaned the data by removing HTML tags, URLs, extra spaces, and any comments over 5,000 characters. This left me with about 159,500 usable comments. I applied the same cleaning steps to both models to keep the comparison fair.

For BERT, I used 75,000 samples because training the full dataset on Kaggle's GPU would have taken too long. For LSTM, I used 40,000 samples since I was training it on my own GPU. Both models used a 90/10 split for training and validation, making sure the toxic-to-non-toxic ratio stayed consistent in both sets.

One limitation of this dataset is that it only has binary labels (toxic or not toxic), so it doesn't distinguish between different types of toxicity like threats, insults, or hate speech.

Methodology

BERT Approach

For BERT, I used the pre-trained bert-base-uncased model from Hugging Face. This model has 110 million parameters and was already trained on huge amounts of text, so

it already understands language pretty well. I just needed to teach it what "toxic" looks like.

The setup was simple: I took the pre-trained BERT model, added a dropout layer to prevent overfitting, and put a classification layer on top to predict toxic or not toxic. I used BERT's tokenizer with a limit of 128 tokens per comment. Anything longer got cut off, anything shorter got padded. I trained with batches of 16 for training and 8 for validation, using the AdamW optimizer with a learning rate of 3e-5. Training took 3 epochs and about 50 minutes on Kaggle's GPU.

I originally wanted to use Kaggle's TPU to make it faster, but after three attempts that each ran for over 3 hours before crashing with memory errors, I gave up. I tried different settings like shorter sequences, smaller batches, and less data, but nothing worked. Once I switched back to the GPU, it ran just fine.

Word2Vec + LSTM Approach

I started by breaking comments into individual words using NLTK, then built a vocabulary of the 50,000 most common words. I trained Word2Vec embeddings with 100 dimensions on the actual toxic comments, so the model learned word meanings directly from this dataset instead of using generic pre-trained embeddings.

The model had an embedding layer that turns words into 100-dimensional vectors, a bidirectional LSTM with 2 layers and 128 hidden units, a dropout layer, and a final classification layer. The bidirectional part reads text both forward and backward, which helps with context. For example, if "bad" appears at the end of a sentence, it might change how the model interprets earlier words.

I trained with batches of 32, a max length of 128 tokens, and ran for 5 epochs. The whole thing took about 15 minutes on my GPU using the Adam optimizer with a learning rate of 1e-3.

Tools and Libraries

For both models, I used PyTorch as the deep learning framework. For BERT, I used the Transformers library from Hugging Face to load the pre-trained model and tokenizer. For Word2Vec + LSTM, I used Gensim to train the Word2Vec embeddings and NLTK for tokenization. I used scikit-learn for metrics calculation (accuracy, precision, recall, F1-score, AUC) and data splitting. I also used pandas for data loading and NumPy for numerical operations.

To make the comparison fair, I used identical preprocessing for both models and evaluated them with the same metrics: accuracy, precision, recall, F1-score, and AUC. Accuracy tells me the overall correctness, precision tells me how often the model is right when it flags something as toxic, recall tells me how many toxic comments the model actually catches, F1-score balances precision and recall, and AUC measures how well the model separates toxic from non-toxic at all confidence levels.

Results

Performance Comparison

Metric	BERT	Word2Vec LSTM
Accuracy	96.95%	96.23%
Precision	83.10%	85.63%
Recall	82.17%	72.92%
F1-Score	0.8263	0.7876
AUC	0.9020	0.9750

Accuracy: Both models got over 96% of predictions correct overall. BERT was slightly better at 96.95% compared to the LSTM's 96.23%, but both are really solid results.

Precision: This measures how often the model is correct when it flags something as toxic. The LSTM actually had higher precision (85.63%) than BERT (83.10%), meaning it made fewer false accusations.

Recall: This measures how many toxic comments the model actually catches. BERT had significantly better recall (82.17%) compared to the LSTM (72.92%).

F1-Score: This balances precision and recall. BERT's F1-score (0.8263) was higher than the LSTM's (0.7876), showing that BERT did a better job overall of balancing catching toxic comments.

AUC: Interestingly, the LSTM had a much higher AUC (0.9750) compared to BERT (0.9020). This means the LSTM was more confident in its predictions.

BERT I went with 3 epochs just due to the length of time it took to train the model. The training loss decreased smoothly from 1.18 in the first batch to around 0.39 by the end of training. Each epoch took about 15-17 minutes on Kaggle's GPU.

Word2Vec + LSTM I went with 5 epochs since it wasn't taking long to train. The training loss went from 0.19 in epoch 1 down to 0.02 by epoch 5. Each epoch took about 3 minutes on my GPU. The best model was actually from epoch 3 with an AUC of 0.9750.

Key Findings

1. BERT is better at catching toxic comments. Higher recall means it finds more of the bad stuff, which is probably more important for content moderation than avoiding false positives.
2. LSTM is more careful. It has higher precision, so when it does flag something, it's more likely to be right. But it's also more conservative and lets more toxic comments slip through.
3. BERT required less work but more resources. I just loaded a pre-trained model and fine-tuned it. The LSTM required building vocabulary, training Word2Vec, designing the architecture, etc. But BERT needed a powerful GPU and took 3x longer to train.
4. Both approaches work well. The fact that both models got over 96% accuracy shows that you don't necessarily need a massive pre-trained transformer to get good results. A simpler custom model can compete if it's built properly.

Discussion

Both model types were new for me, just because I haven't been exposed to them outside of this class. Since I started with my LSTM W2V model first I tried to use that code to help with my BERT model. The good news is the resources on these models are extensive. Hugging Face handled a boatload of my documentation for creating my BERT model from the ground up. It was helpful to take a peek at what others were doing with their models in Kaggle as I was beginning to code my model.I looked at other Kaggle notebooks to see how people approached the problem, though most were focused on getting the absolute best results while I just wanted to get my first working model.

Starting with LSTM was smart because it trained faster, letting me debug and understand the pipeline before tackling BERT. Kaggle's GPU worked extremely well when training BERT, and the modular code structure for LSTM made debugging easier.

The main challenge was trying to use Kaggle's TPU to train BERT. After three failed attempts, each running over 3 hours before crashing with "Your notebook tried to allocate more memory than is available" errors, I gave up. I tried adjusting sequence lengths, batch sizes, data sizes, and epochs, but nothing worked. Switching back to GPU solved the problem immediately.

BERT's pre-trained knowledge of language gave it a huge head start. It already understands things like negation ("not bad" vs "bad"), sarcasm, and complex sentence structures. The LSTM had to learn all of that from scratch with only 36,000 training examples.The attention mechanism also helps it understand long-range dependencies in text better than LSTMs. It can see how words at the beginning and end of a comment relate to each other, while LSTMs process text sequentially and might "forget" earlier context.

Conclusion & Future Work

Both approaches effectively detected toxic comments. BERT achieved 96.95% accuracy with better recall, while LSTM reached 96.23% accuracy with higher precision and faster training. So it really depends on what you're trying to achieve.

Here are a few futures I had ideas for while working on my project:

1. Instead of just toxic/non-toxic, classify different types of toxicity (threats, insults, hate speech, obscenity). The Jigsaw dataset actually has these labels, I just didn't use them.
2. Train on Wikipedia comments but test on Reddit, Twitter, or YouTube comments to see if the models generalize to different platforms and writing styles.
3. Build an API that takes a comment as input and returns a toxicity score in real-time. This would be useful for actually implementing content moderation.

Bibliography

Pawar, Vijaya R., et al. "Purging the Poison: A Machine Learning Approach to Filtering Toxic Comments." *International Research Journal on Advanced Engineering Hub (IRJAEH)*, vol. 2, no. 7, July 2024, pp. 2065-2073.

Zaheri, Sara, et al. "Toxic Comment Classification." *SMU Data Science Review*, vol. 3, no. 1, 2020, article 13.