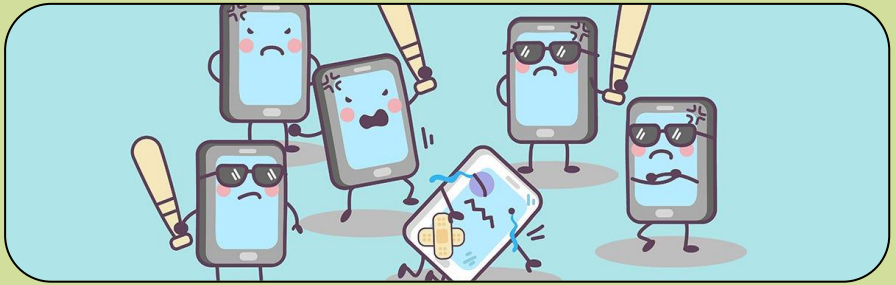
Toxic Comment Filtering

BERT vs Word2Vec + LSTM



By: Anthony Konas

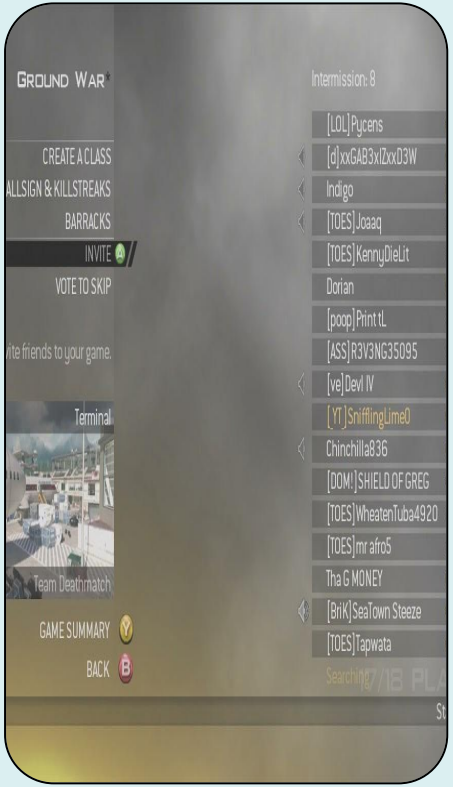
Problem Overview

Toxic and abusive comments are a persistent problem on online platforms, creating hostile environments and discouraging constructive discussion.

My project aims to build classifiers that can automatically detect toxic comments using word embeddings and neural networks.

I implemented two different approaches:

1. A BERT-based classifier that leverages pre-trained transformer embeddings,
2. A Word2Vec-based model using a simpler LSTM architecture.



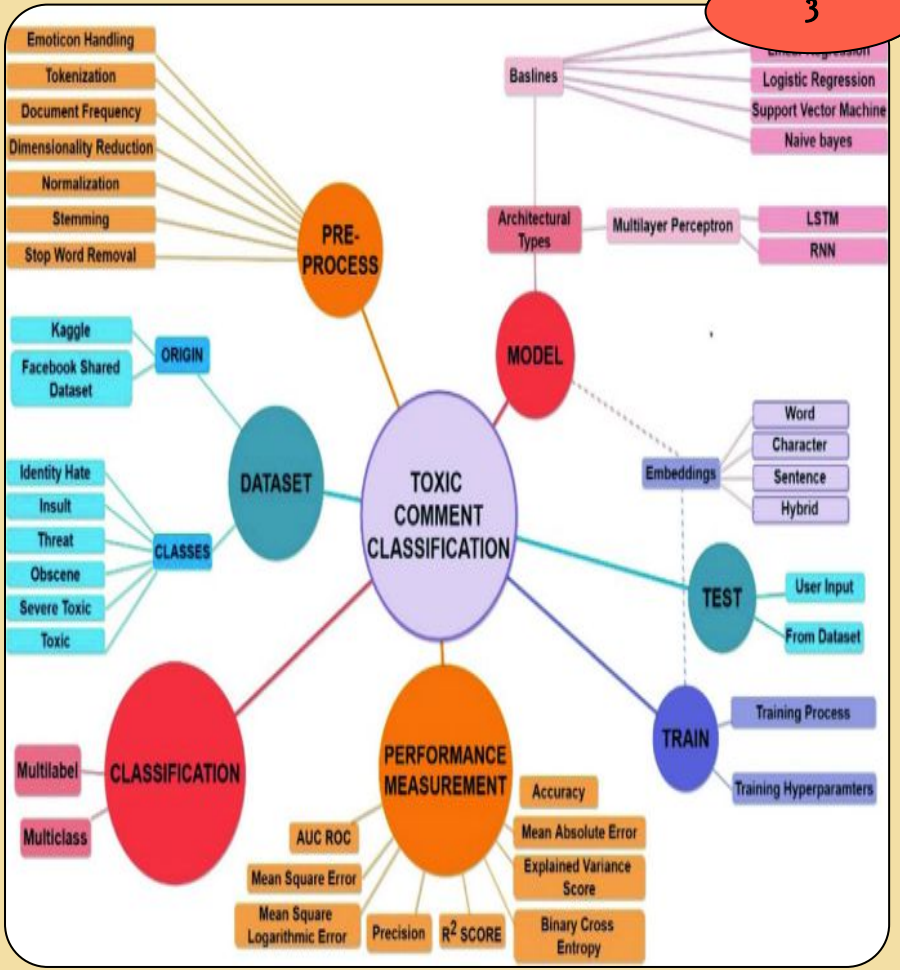
Motivation & Goals

My project consists of two different approaches:

1. BERT: A pre-trained transformer model
2. Word2Vec + LSTM: A custom built model

My goal for this project was to develop two programs that have the ability to automatically detect toxic comments in online discussions that will ultimately help moderate content and limit hate speech on social media and forum platforms.

The intent behind designing these two programs was to see which performs better at identifying toxic comments.



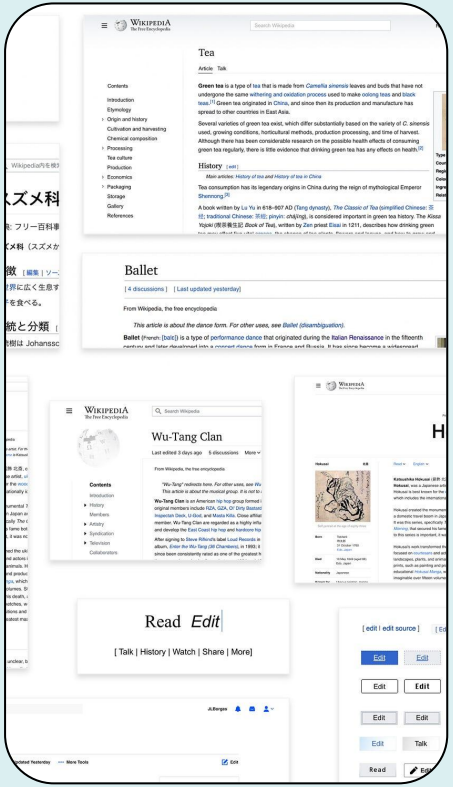
Dataset Description

Source: Jigsaw Toxic Comment Classification dataset from Kaggle

Dataset Size: 159,571 Wikipedia comments

The steps I took for preprocessing this dataset is as follows:

1. Remove HTML tags and URLs
2. Clean extra whitespace
3. Remove empty comments and extremely long comments
4. Split the data 90/10 with 90% used for training and 10% for validation



Approach Overview

My testing consisted of two separate approaches…

1. BERT:
    - I used a pre-trained BERT model
    - Fine tuning it on toxic comment data
    - BERT conveniently already knows language, it just needs to learn what “toxic” actually means
2. W2V + LSTM:
    - Train the W2V on the Kaggle data and have it learn the meanings from the comments themselves
    - Feed the embeddings into the bidirectional LSTM
    - LSTM then learns patterns in sequences to classify toxicity

Tools & Techniques

Libraries & Frameworks:

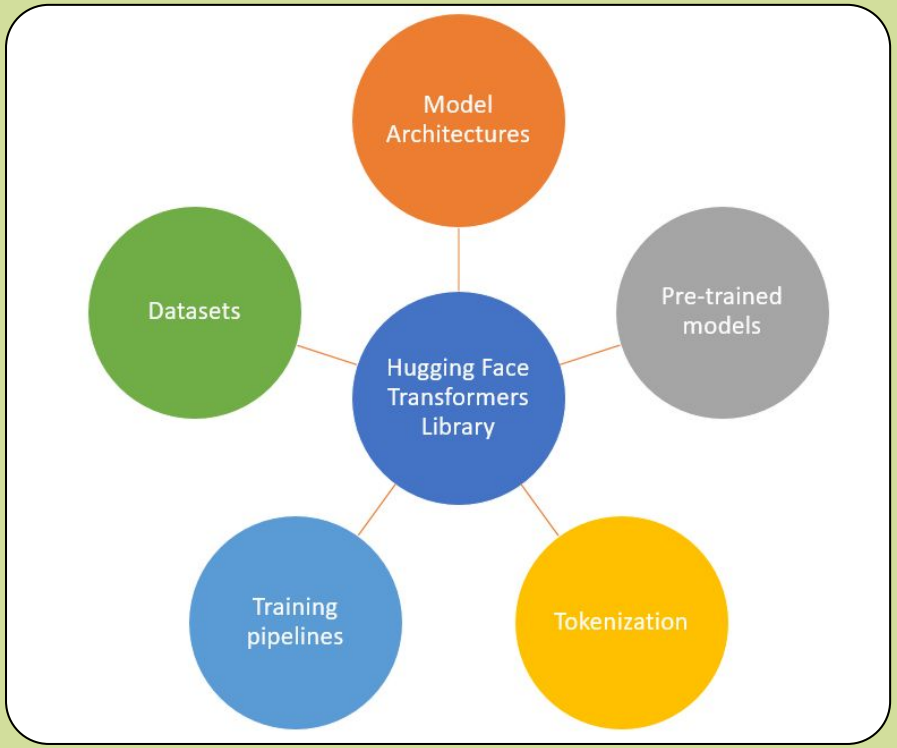
1. Pytorch
2. Transformers
3. Gensim
4. NLTK
5. Scikit-learn

Models:

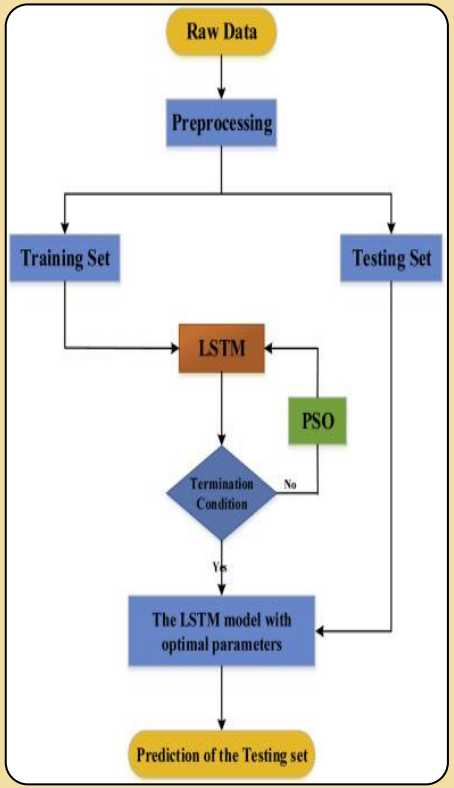
1. BERT
2. W2V + LSTM

Development:

1. BERT trained on Kaggle GPU
2. LSTM trained on my local machine



Experiments & Analysis



BERT Configuration:

1. I tested different sequence lengths, batch sizes, data sizes to try and get the TPU to work in Kaggle, but unfortunately it wouldn't work so I switched to GPU.
2. Tested sequence lengths were 64, 128, 256, and 512.
3. I tried batch sizes of 2, 4, 8, and 16.
4. I tried data sizes of 20k, 36k, 50k, and 75k.
5. Eventually I landed on using the GPU with a sequence length of 128, a batch size of 16, and a data size of 75k

LSTM Configuration:

1. Embedding dimensions were set to 100
2. I had a bidirectional architecture
3. Experimented with hidden dimensions of 128

Comparison:

1. Both models used identical preprocessing and training splits for a fair comparison
2. I also used the same evaluation methods per the project requirements (accuracy, precision, recall, F1, AUC)

"BERT's pre-trained knowledge of language helped it learn the toxic classification task faster, while LSTM had to learn both meanings and classification patterns."

Results

Performance Comparison

Metric	BERT	W2V + 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

Key Findings

1. Both models achieved a high accuracy rating ~96-97%.
2. BERT was slightly better with more balanced precision and recall, meaning it caught more toxic comments.
3. LSTM had a higher precision but lower recall, meaning it missed more toxic comments.
4. BERT required less manual work, but more computational resources.
5. LSTM required more epochs while BERT only required a few to produce these scores (3 vs 5 epochs)

Challenges & Lessons Learned

Major challenges:

Kaggle TPU nightmare:

- Spent hours trying to get BERT working on TPU
- Multiple out-of-memory crashes after long training runs
- Eventually gave up and switched back to GPU and it worked immediately…

PyTorch version issues:

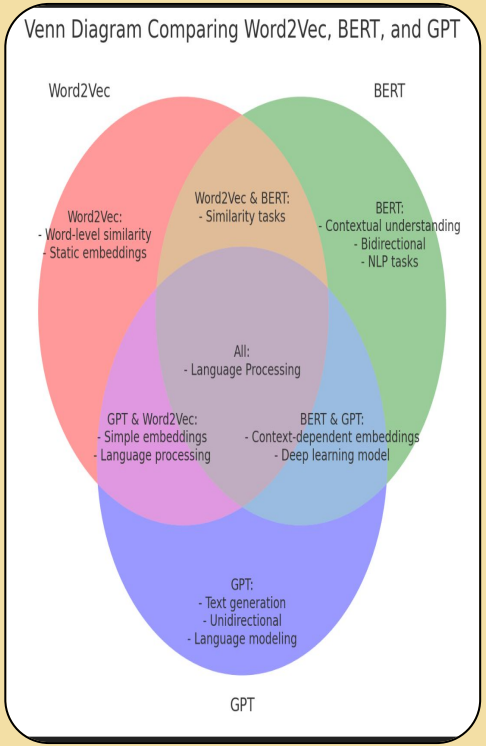
- Functions like AdamW moved between libraries in newer versions
- Had to update import statements multiple times

Memory management:

- Had to carefully balance batch sizes, sequence lengths, and data samples
- Learned that bigger isn't always better - conservative settings were more reliable

What I learned:

- Sometimes simpler is better (GPU over TPU)
- Both pre-trained models (BERT) and custom models (LSTM) can achieve good results
- Data preprocessing is crucial



Conclusion & Future Work

Key takeaways:

- Both BERT and Word2Vec + LSTM can effectively detect toxic comments
- BERT performed slightly better with less manual work, but LSTM was faster to train
- Pre-trained models like BERT give you a head start, but custom models are more controllable
- Good preprocessing and data cleaning matter more than fancy models

Future improvements:

- Try multi-label classification (different types of toxicity: threats, insults, hate speech)
- Experiment with larger BERT models (BERT-large, RoBERTa)
- Add more data augmentation techniques
- Test on real-world data from different platforms (Reddit, Twitter)
- Deploy as a simple API for real-time moderation

Final thought: Both approaches work well - the choice depends on your resources and needs. If you have the compute power, BERT is easier. If you want more control and faster training, Word2Vec + LSTM is solid.