FINAL REPORT

Toxic Comment Classifier: Identifying Toxic Comments on Online Platforms using Text Classification Models

GitHub

Irsa Ashraf Yifu Hou Ken Kliesner

Abstract

The online environment has become increasingly hostile and abusive in recent years, with hate speech and other forms of abusive behavior being prevalent on social media platforms. Research has shown that marginalized groups such as women, racial minorities, and members of the LGBTQ+ community are disproportionately targeted with online harassment (Suler, 2019). A toxic online environment has a significant and wide-ranging negative impact. Victims of online harassment may experience psychological distress, anxiety, depression, and even suicidal ideation. In addition, the spread of hate speech online can contribute to the normalization of discriminatory attitudes and behavior in society (Duggan & Smith, 2013).

Project Overview

Our goal is to build a classifier that can accurately classify and categorize toxic comments on social media platforms. The model will be trained on a pre-labeled dataset containing toxic and non-toxic online comments to identify different levels of toxicity on media platforms.

At the most fundamental level, the project aims to develop an NLP model that can accurately perform binary classification between toxic and non-toxic comments. In addition, the project aims to identify different categories of toxicity, such as identity-based hate, threats, insults, and others. Ultimately, the project sets the goal to develop a generalized, scalable language model as a solution to classifying different types of toxic comments across different platforms. This required training on different datasets, fine-tuning the model for high accuracy, and performance optimization–some of it was out of the scope of the course and so we adjusted the implementation according to time and resources available to us.

Description of Dataset

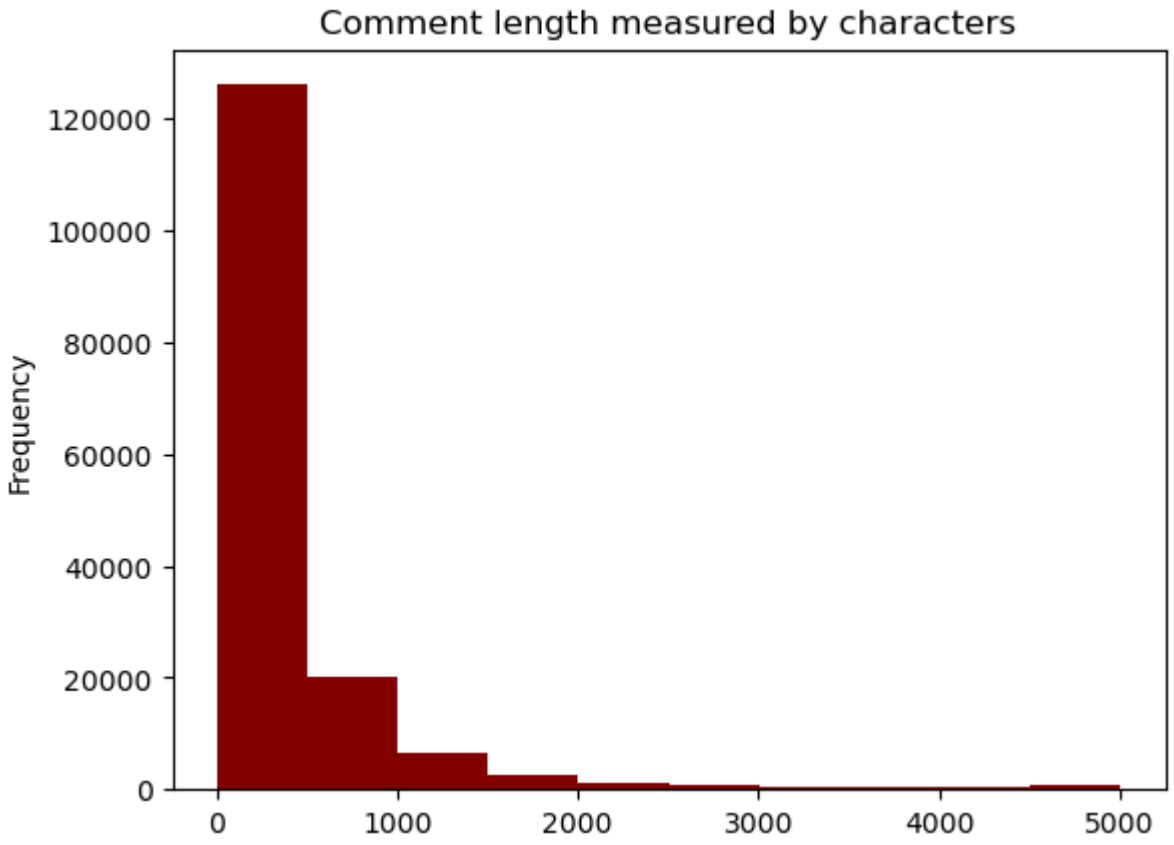
Jigsaw Toxic Comment Classification Dataset.

The dataset we used for this project is the Jigsaw Toxic Comment Classification Challenge dataset on Kaggle, which consists of a large number of comments from Wikipedia's talk page edits, along with binary labels indicating whether each comment is toxic or not. The data is in a CSV format, with each row representing a single comment and its associated labels. The dataset contains around 310,000 comments, and each comment is labeled on 6 different types of toxicity: toxic, severe toxic, obscene, threat, insult, and identity hate. One comment can be categorized as more than one type of toxicity.

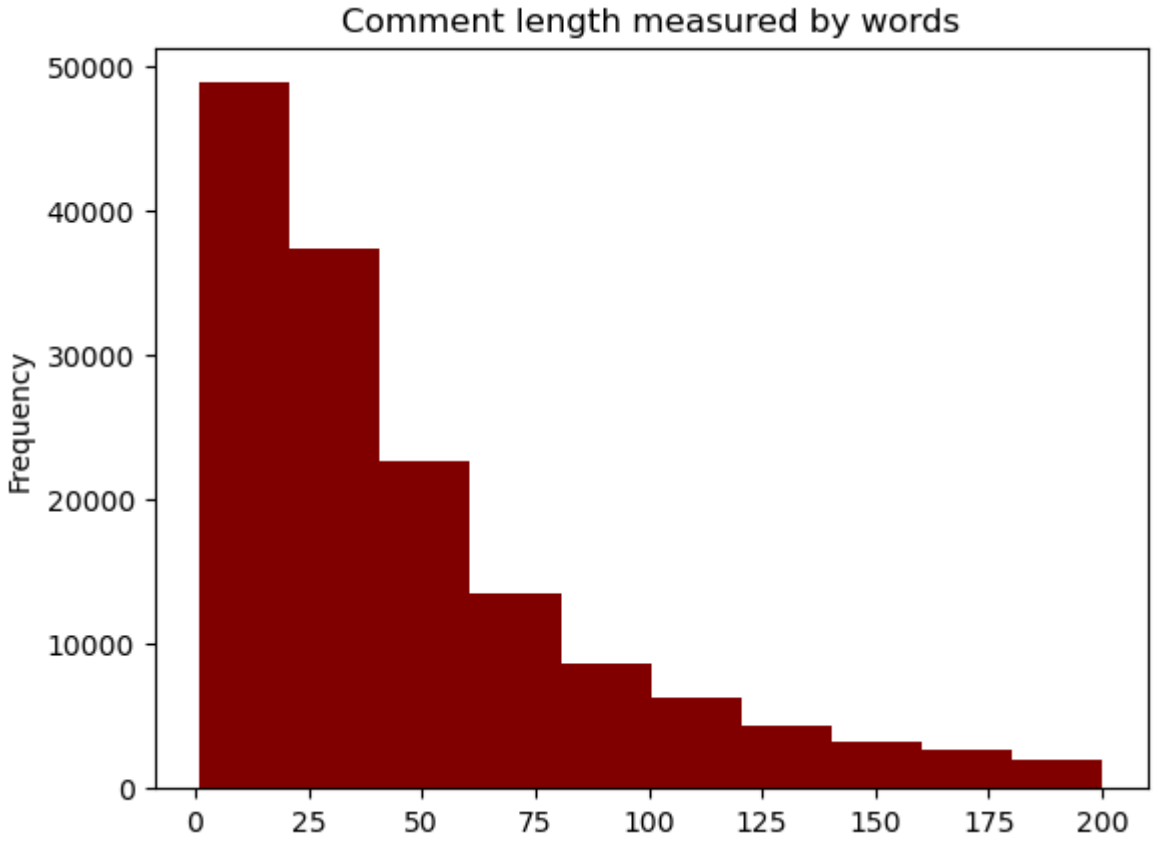
Data Pre-Processing:

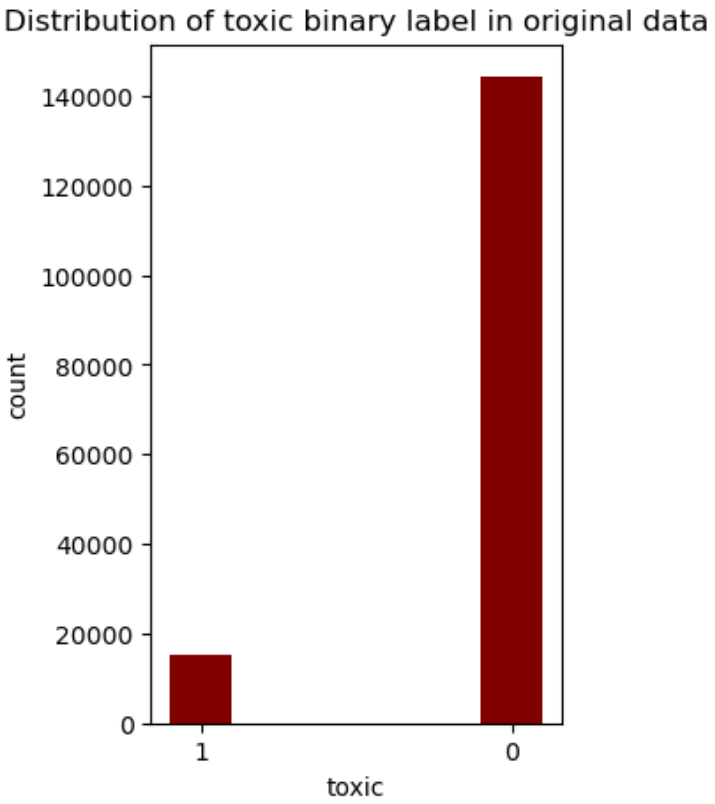
The training dataset that we have been using (which comes from the *Jigsaw Toxic Comment Classification Challenge* dataset on Kaggle) contains 159,571 online comments. The average length of any individual comment's raw text is 68, measured in the number of characters of the comment. The proportion of comments in the dataset that have over 60 words is 32%. The proportion of comments that have over 100 words is 18%. And the proportion of comments that have over 200 words is only 6%.

The character count distribution is as follows:



The word count distribution for comments with no more than 200 words is as follows:





BOW:

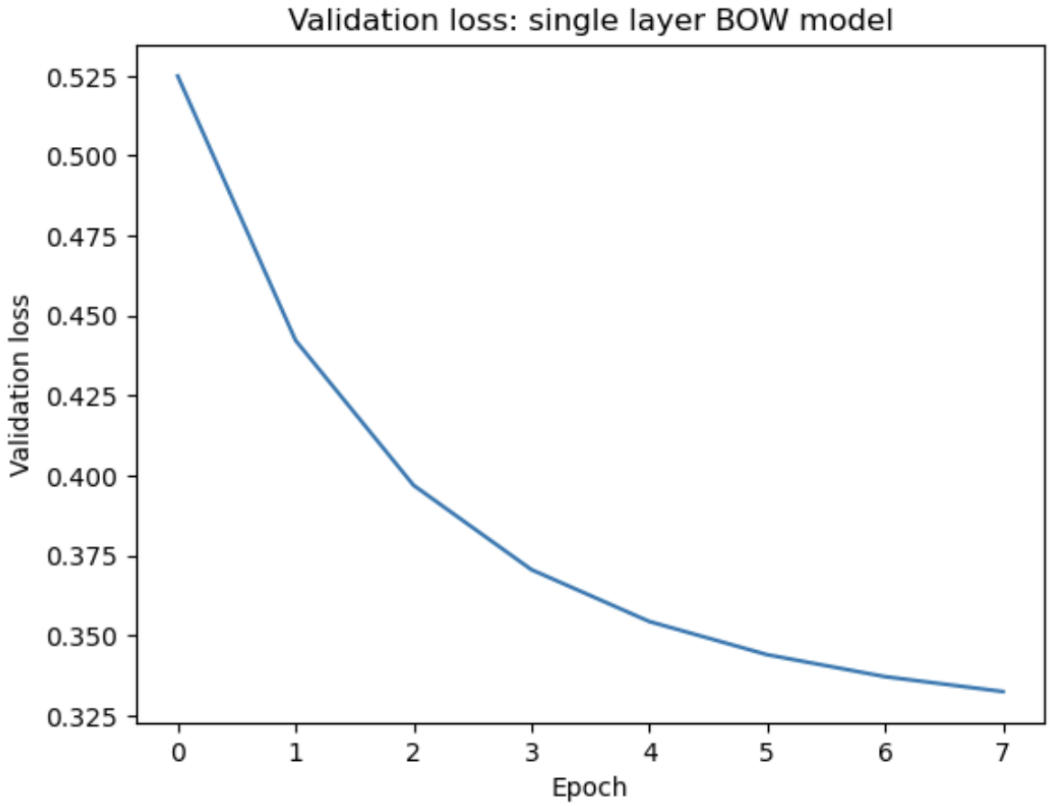
In our first attempt with creating our BOW model, we separated out our data into 70% training, 20% validation, and 10% test datasets. Therefore, there were 111,699 comments in the training dataset, 32,074 comments in the validation dataset, and 15,798 comments in the test dataset. Then we tokenized our data and built a vocabulary set using TorchText, which contained 6,246 tokens that we were able to collect. From there, we were able to start setting up our model as a Single-Layer CBOW Model, skipping any unidentified sentences. In doing this approach, we got the below training losses between 0.681 and 0.531 for iterations between 500 and 6500.

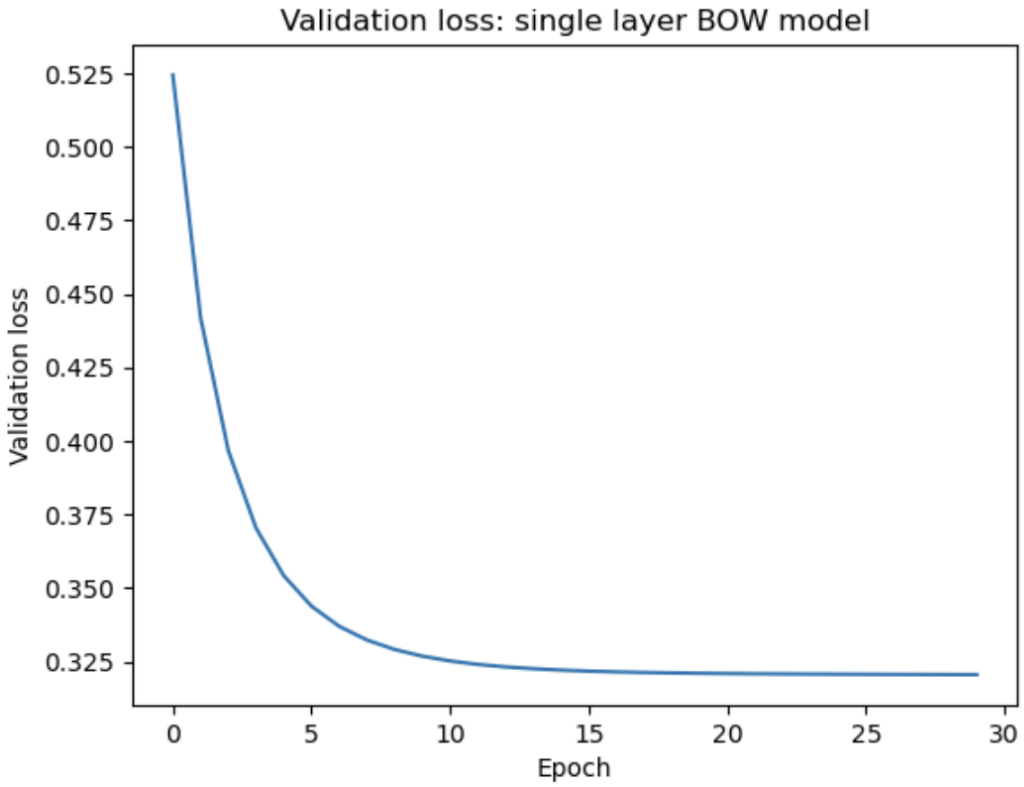
However, there were not enough words being caught because when there were no words identified from a comment, the loss was returning as ***nan***, which was breaking the evaluation process.

With the above in mind, we decided to move onto another approach. In this second attempt, we switched to a pre-trained embedding GloVe Model. Again, we separated out our data into 70% training, 20% validation, and 10% test datasets. Therefore, there were 111,699

comments in the training dataset, 32,074 comments in the validation dataset, and 15,798 comments in the test dataset.

After around 8-10 epochs, the reduction of validation loss reaches a plateau. Below are the results of experimenting with 8 and 30 epochs.





For the final iteration of our CBOW classifier, we ran 10 epochs and the best model we got was one with the following features and parameters:

```
: best_model

: BoWClassifier(
    (linear): Linear(in_features=6246, out_features=2, bias=True)
  )
```

We computed accuracy as the ratio of correct classifications and all classifications. After training on the 10 epochs, our model gave an accuracy of around **85%.** Although this isn't the highest accuracy score, it was ideal for a baseline model, keeping the following key factors in mind:

1. The baseline model is currently only considering the accuracy of predicting toxic comments, which can be often associated with certain keywords (cursing words, slurs, etc.). Moving forward to other labels such as distinguishing between threats and insults, we implemented models with more complex structures.

2. The training data has an imbalance in labels. The majority of the comments are non-toxic. Therefore, the high accuracy achieved by the model was probably because of the overwhelming amount of non-toxic labels. We listed some potential solutions to this problem in the following part.

CNN

Binary Classification Using Keras

To achieve better accuracy, we moved on to a more complex model–a convolutional neural network. We started off with a binary classification model to identify whether comments are toxic or not toxic. Since all our code is written in Python, we decided to implement a CNN using both PyTorch and Keras and compare the results from both.

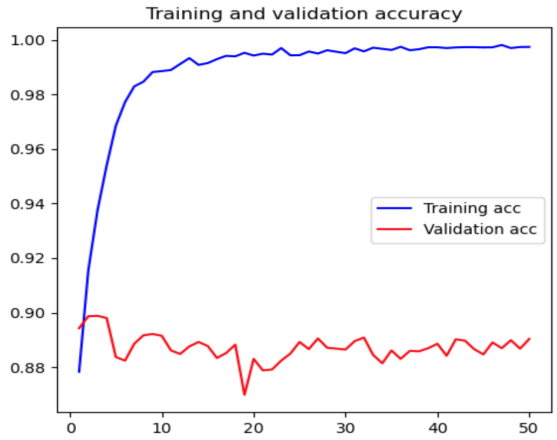
The dataset used was the same one used for the BOW model, with balanced classes for toxic and not-toxic comments and we used pre-trained GloVe Embeddings with Embedding Dimension = 300.

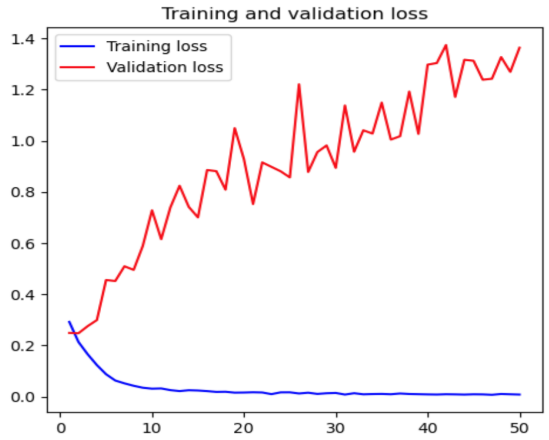
Using the keras library, the parameters selected for the best model are:

Parameter	**Value**
Max_sentence_length	200
Batch Size	16
num_words	10000
Filters	64
Kernel_size	5
Activation	ReLu
Units	64

The model was trained for 50 epochs and tested on the validation and testing dataset. The plots below show that after around 15 epochs, the training accuracy and the loss started to plateau.

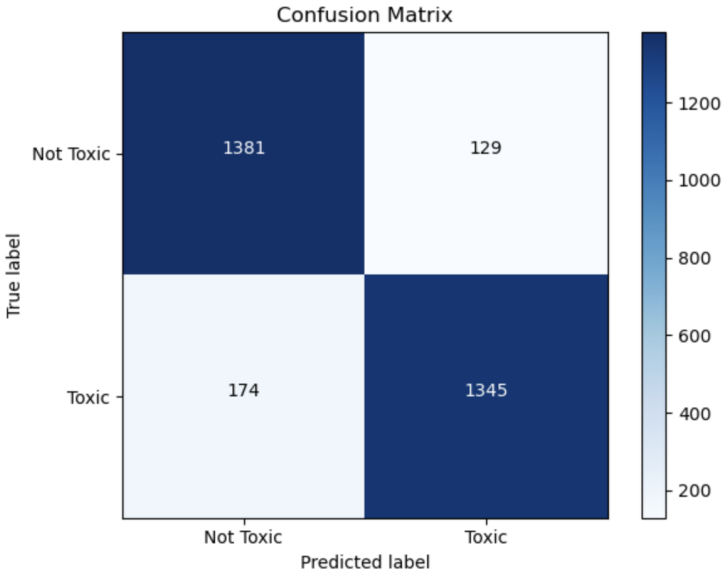
```
Training Accuracy: 0.9991
Validation Accuracy:  0.8904
Testing Accuracy:  0.9000
```

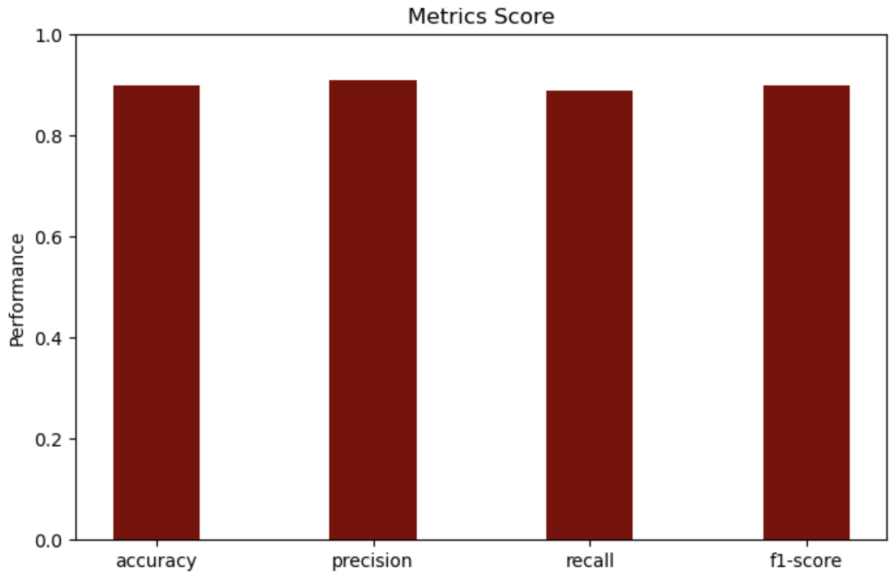




The model does not seem to be overfitting since the testing accuracy is much lower than the training accuracy.

The model was trained on an M1 Mac and took 35 minutes for 50 epochs to complete. We first checked how well the model performed by looking at a confusion matrix of our actual and predicted values and then we calculated the accuracy, precision, recall and f1-score.
The metrics we computed to check how well the model performed were accuracy, precision, recall and F1 score.

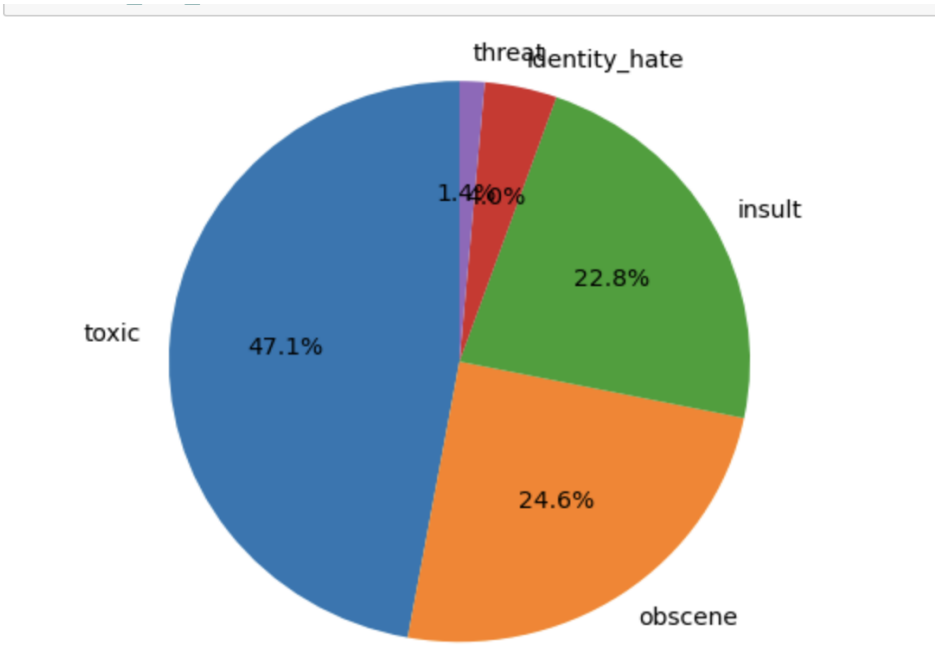




Multi-Label Classification

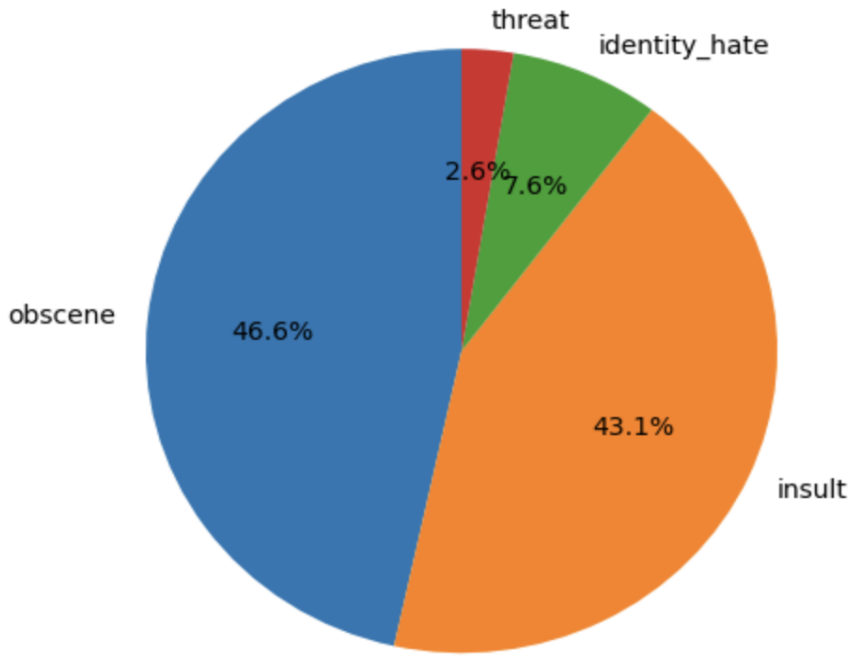
Our dataset has more columns besides toxic. More specifically, we aimed to build a CNN that can accurately predict *why* a comment is categorized as toxic. As such, the extra columns we have are ‘severe_toxic’, ‘obscene’, ‘threat’, ‘insult’ and ‘identity_hate’. Starting off, we decided to drop the ‘severe_toxic’ column since we had already trained a model on ‘toxic’ and the distinction between ‘toxic’ and ‘severe_toxic’ was vague. The rest of the training data, including pre-processing, was the same for this model as well. However, we dealt with imbalance classes, particularly for ‘identity_hate’ and ‘threat’ which made up less than 5% of the dataset.

Proportion of Labels in the Entire Dataset



To deal with this imbalance, we reduced our training dataset to only toxic comments. So the size of our new training data for the multi-label classification using Keras is 15,294 with a 100% toxic comments.

Proportion of Labels in Dataset of Only Toxic Comments



While ‘identity_hate’ and ‘threat’ still make up only ~8% and ~2% of the dataset, it still a an upgrade from the ~4% and ~1% we had earlier. Additionally, we will handle this problem by using metrics that are used specifically for imbalance data, such as F1 score.

The parameters used to train our multi-label classification model in Keras are

Parameter	**Value**
Max_sentence_length	200
Batch Size	16
num_words	10000
Filters	128
Kernel_size	5
Activation	ReLu
Dropout	0.5
Units	64

The results we obtained from our multi-label classification model after testing it on the test dataset were not the highest.

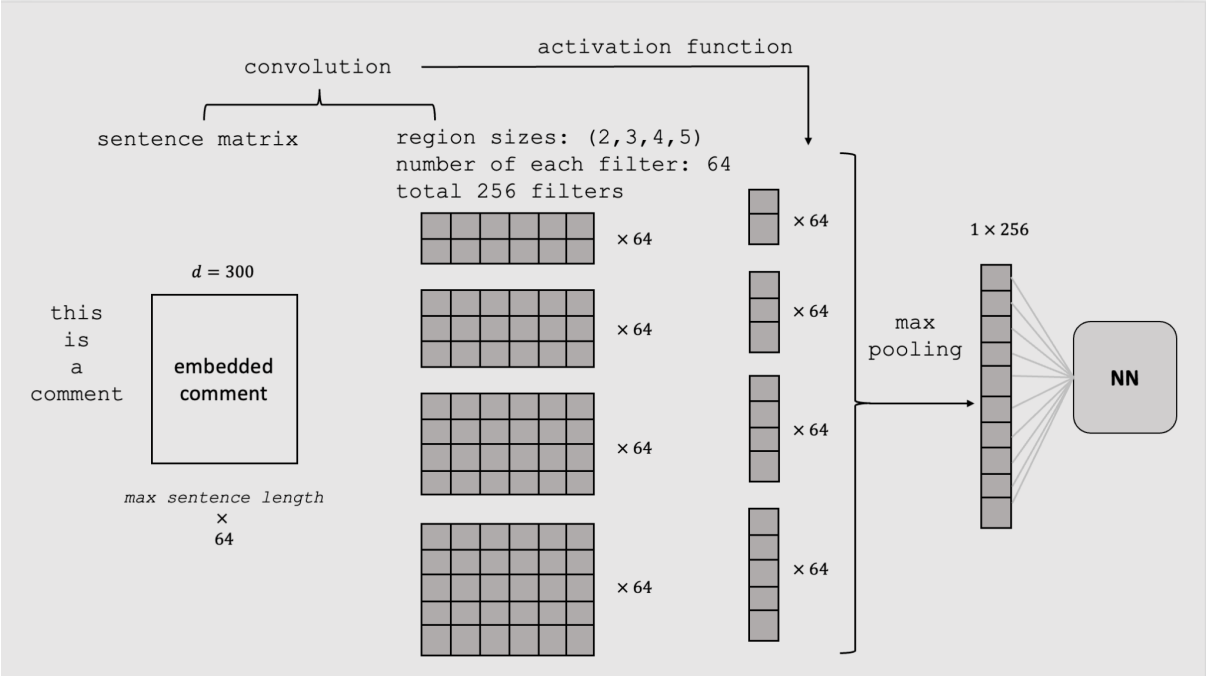
	Precision	**Recall**	**F1-Score**
obscene	0.82	0.77	0.79
threat	0.73	0.39	0.51
insult	0.71	0.70	0.71
identity_hate	0.56	0.35	0.43

This was expected because of how imbalanced the dataset was, especially for the classes ‘identity_hate’ and ‘threat’. The model we trained was also quite simple and so we suspect it was not able to learn the patterns of different classes well enough to generalize well on unseen data.

However, we still wanted to expand our model beyond just being able to classify a comment as either toxic or not-toxic. We therefore decided to look at each class as a separate binary classification problem to understand which classes a CNN model can predict the best. This was implemented using PyTorch (explained below).

CNN Model (First Version - base CNN model) Using Pytorch

The first model has 4 different filter sizes: 2, 3, 4 and 5, each focusing on different sizes of N-grams. We assigned 64 filters of each size for this model. The fully-connected neural network has 256 input features and binary outputs. The model's dropout rate is set to be 0.5 to avoid over fitting. This model was trained on a MacBook Pro M2 CPU for 10 epochs, which took around 30 minutes. This model is used to identify major label (toxic) and minor labels (severe_toxic, obscene, threat, insult, identity_hate).



PARAMETERS:

Description	Values
input word vectors	GloVe
embedding size	300
filter sizes	(2, 3, 4, 5)

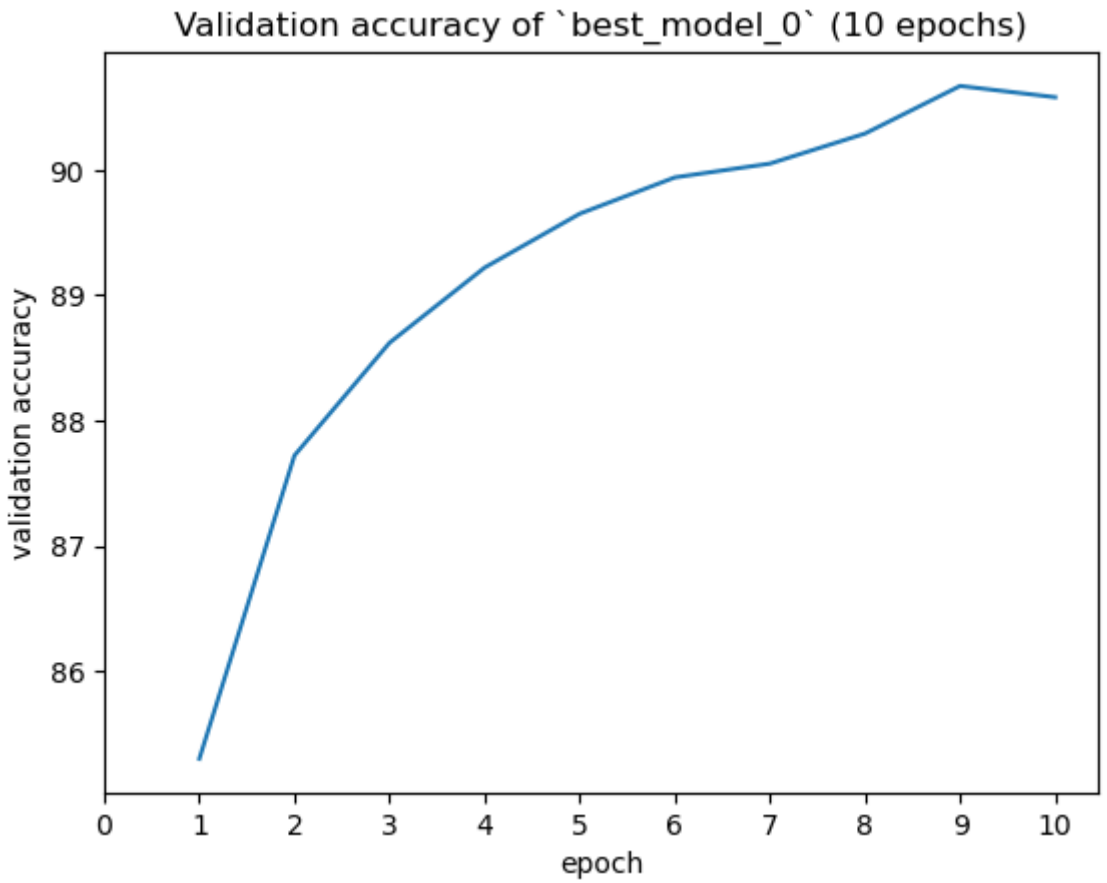
num filters	(64, 64, 64, 64)
activation	ReLU
pooling	1-max pooling
dropout rate	0.5

PERFORMANCE:

Identifying toxic comments with 10 epochs of training:

For training and validation, we see the following performance:

Epoch	Train Loss	Val Loss	Val Acc	Elapsed
1	0.515071	0.387810	85.50	222.95
2	0.357476	0.314949	87.84	214.95
3	0.307458	0.286911	88.58	217.30
4	0.283827	0.271694	89.29	212.41
5	0.269294	0.261957	89.29	217.79
6	0.257123	0.254316	89.50	238.22
7	0.247575	0.248300	89.76	223.09
8	0.239426	0.243885	89.81	216.53
9	0.233904	0.239794	90.08	210.70
10	0.227885	0.236559	90.17	221.15

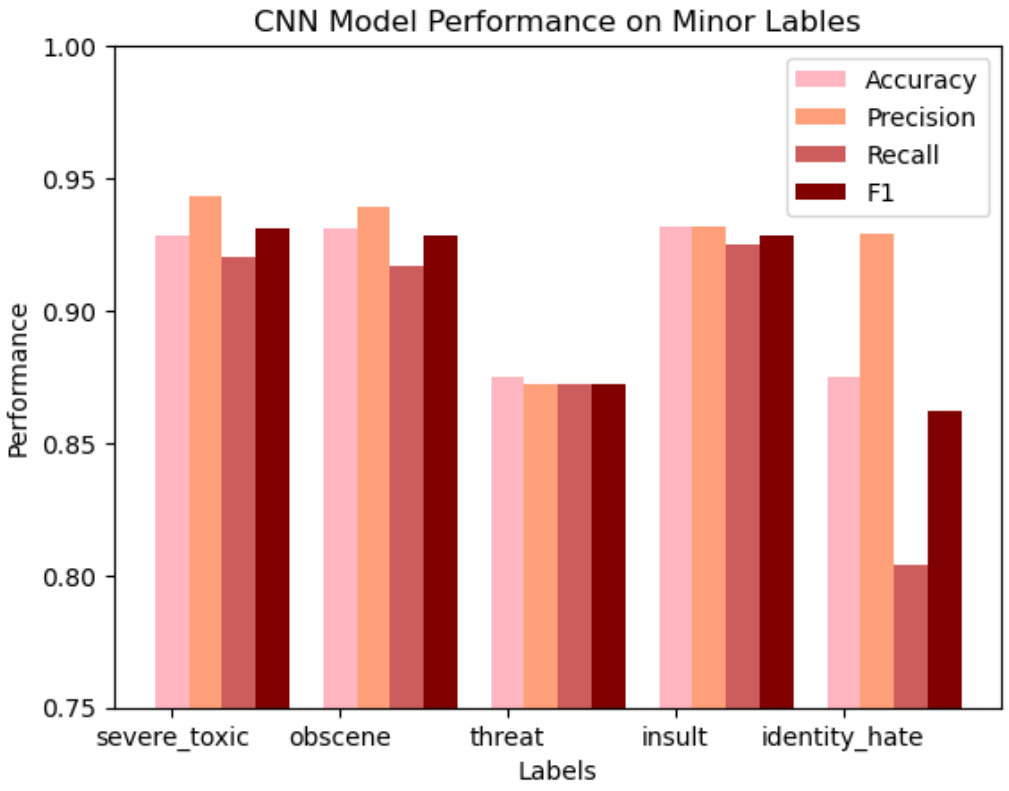


PERFORMANCE:

For testing, the first CNN model performed well on 2865 testing data:

Accuracy	Precision	Recall	F1
0.914455	0.920194	0.910714	0.91543

Below are the results for classifying each label of comments:



PATTERNS IN FAILED PREDICTIONS:

We focus on improving the prediction accuracy on major label `toxic`. The binary-label CNN model performed well on identifying toxic comments. Below are some examples of failed predictions:

False Negatives

- misspell or uncommon toxic language
  - ID: e22a2557c33d5df3
  - *"tno thanks mate p i s s offe"*
  - ID: f16ec7cafd4ff73c
  - *"you obviously know shit-nothing about physics, if the buildings were ..."*
- mitigated or oddly-worded insults
  - ID: 06a44c69b4c3fb43
  - *"In response to your recent comment on my talk page. I suggest you contract cancer."*
- ambiguous connotations for language models
  - ID: e8d66a843390f637
  - *"Do it and I will cut you"*

False Positives

- triggering words used in non-toxic context
    - ID: *289b9ebd8ee46b91*
    - *"This article is useless without pics"*

Ambiguous Labels (up for debate of interpretation):

- It is worth mentioning that the labeling of original dataset is not perfect - there are a few comments that might be incorrectly labelled.
    - ID: *583c3800a5b3b464*
    - *"Do what you want, but you'll never get rid of me, that's a promise. Give your sister a kiss for me."*
        - Labeled as `not toxic`, we identified as `toxic`
    - ID: *d4090f8db8939d73*
    - *"Yo who the heeck wrote this and how the heck do they even know what happends."*
        - Labeled as `toxic`, we identified as `not toxic`

With 20 epochs of training:

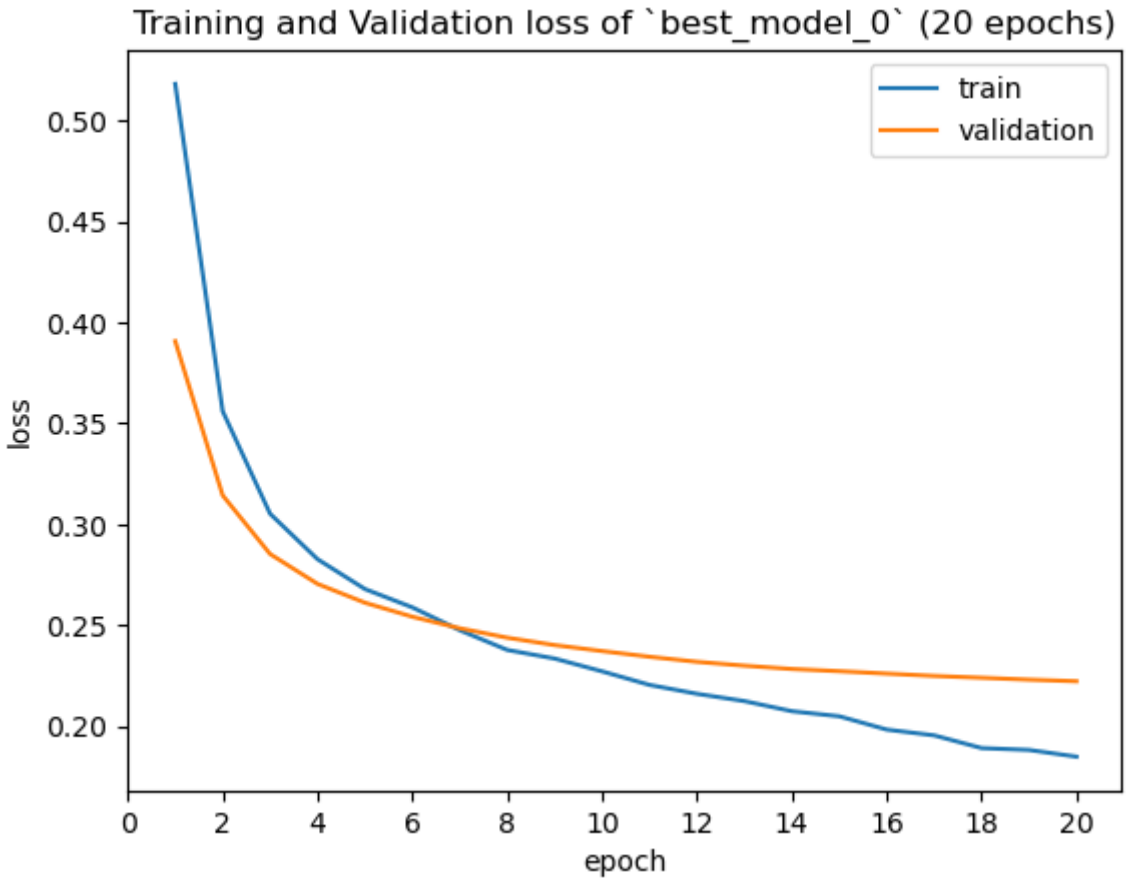
To improve the model, we increased the number of epochs to 20, and selected the best-performed model from this training process (before the model shows overfitting).

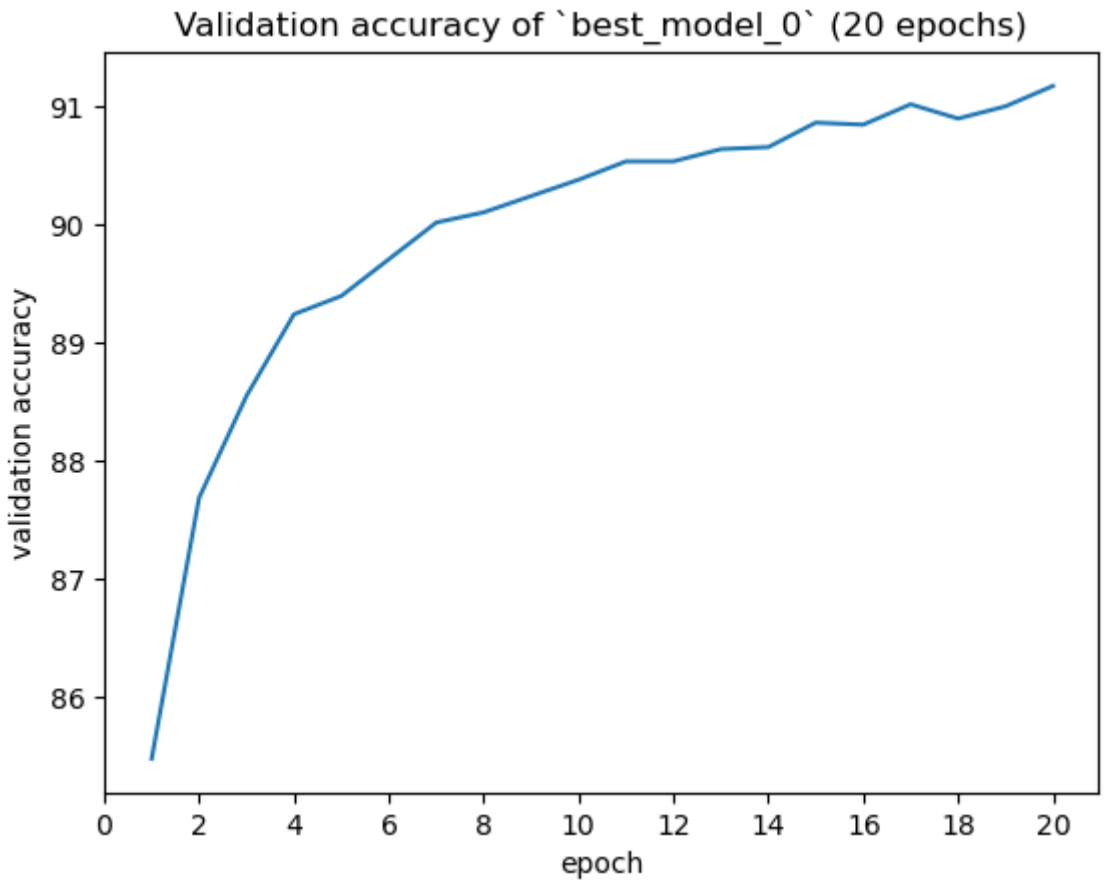
The exact same model has been trained for another 10 epochs. We see slight improvement in accuracy, recall and f1. However, the validation accuracy start to reach a plateau after around 15 epochs.

PERFORMANCE IN TRAINING:

Epoch	Train Loss	Val Loss	Val Acc	Elapsed
1	0.518177	0.390717	85.49	206.55
2	0.356052	0.314333	87.69	213.67
3	0.305211	0.285256	88.55	216.24
4	0.282695	0.270456	89.24	276.86
5	0.267897	0.261121	89.39	263.94

6	0.258804	0.254165	89.70	269.75
7	0.247487	0.248454	90.01	231.39
8	0.237717	0.243787	90.10	206.41
9	0.233450	0.240119	90.24	205.30
10	0.227105	0.237166	90.38	205.57
11	0.220388	0.234342	90.53	205.35
12	0.215990	0.231839	90.53	205.72
13	0.212377	0.229920	90.63	210.60
14	0.207373	0.228286	90.65	208.69
15	0.204846	0.227199	90.86	209.43
16	0.198263	0.225998	90.84	207.99
17	0.195395	0.224842	91.01	207.73
18	0.189093	0.223940	90.89	208.72
19	0.188172	0.222998	91.00	203.57
20	0.184841	0.222217	91.17	351.67


Training and Validation loss of `best_model_0` (20 epochs)
train
validation
loss
epoch
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
2
4
6
8
10
12
14
16
18
20



PERFORMANCE IN TESTING:

Accuracy	Precision	Recall	F1
0.92074	0.918882	0.925824	0.92234

CNN Model (Second Version - higher number of filters)

To further improve model accuracy on major label toxic, we implemented CNN Model Second Version, and increased the number of filters for each size (2, 3, 4, 5) from 64 to 128:

PARAMETERS:

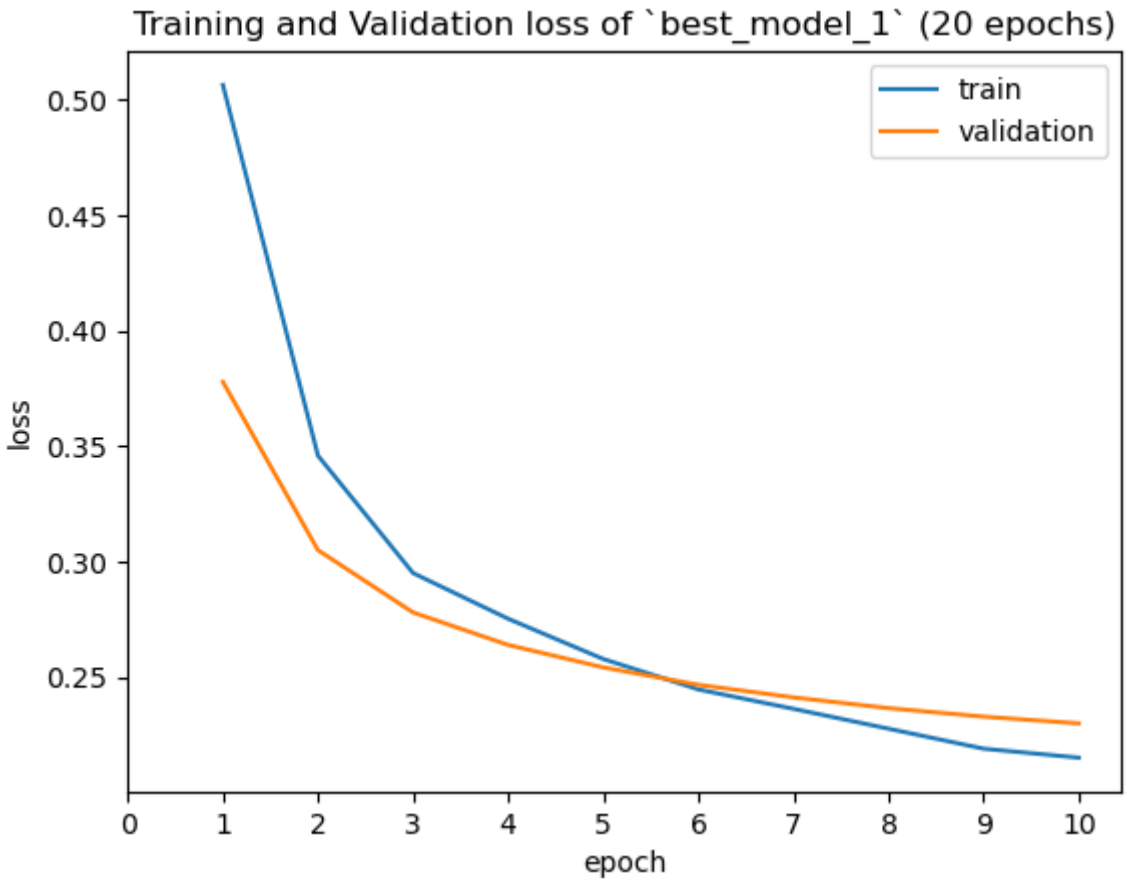
Description	Values
input word vectors	GloVe

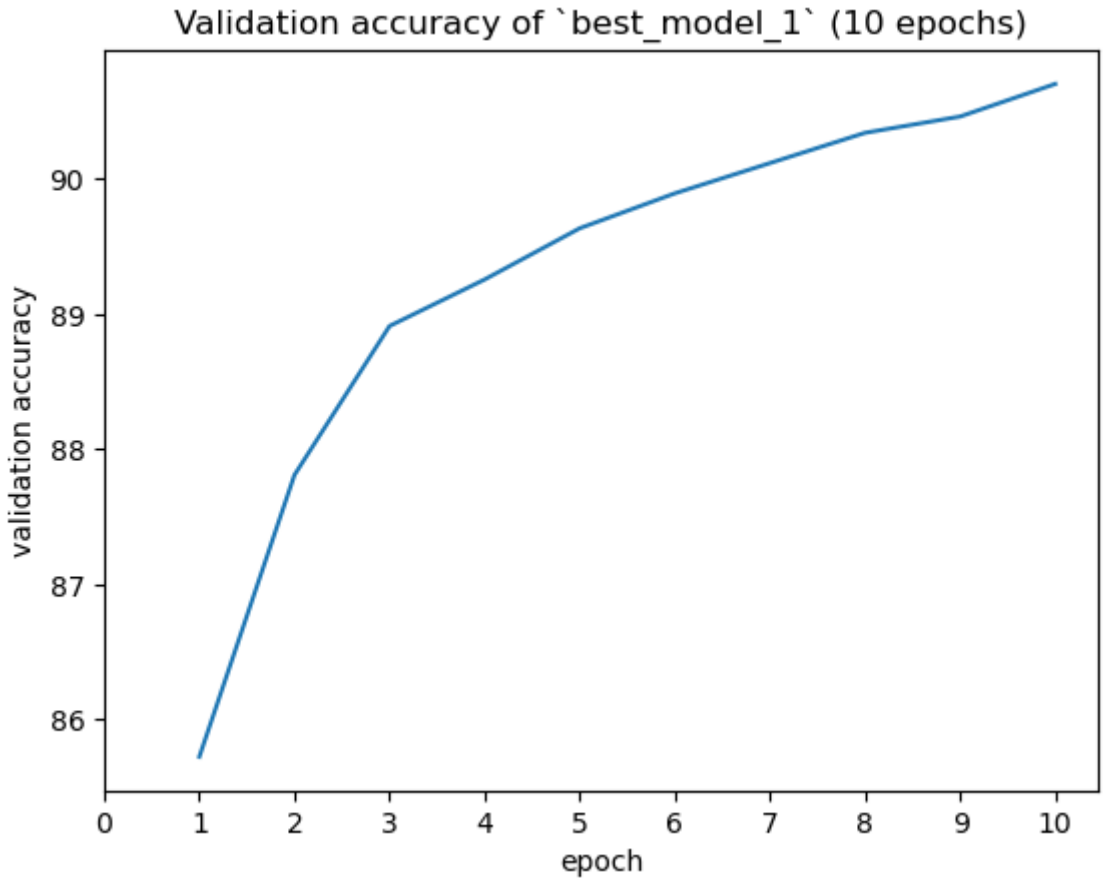
embedding size	300
filter sizes	(2, 3, 4, 5)
num filters	(128,128,128,128)
activation	ReLU
pooling	1-max pooling
dropout rate	0.5

PERFORMANCE:

With 10 epochs of training:

Accuracy	Precision	Recall	F1
0.915852	0.917526	0.916896	0.917211





CNN Model (Third Version - with low dropout rate)

Since CNN Model Second Version did not show a clear sign of overfitting, we decreased Dropout rate from 0.5 to 0.2, with the hope that it captures more features from the training data. We see a slight increase in the performance metrics.

PARAMETERS:

Description	Values
input word vectors	GloVe
embedding size	300
filter sizes	(2, 3, 4, 5)
num filters	(128,128,128,128)

activation	ReLU
pooling	1-max pooling
dropout rate	0.2

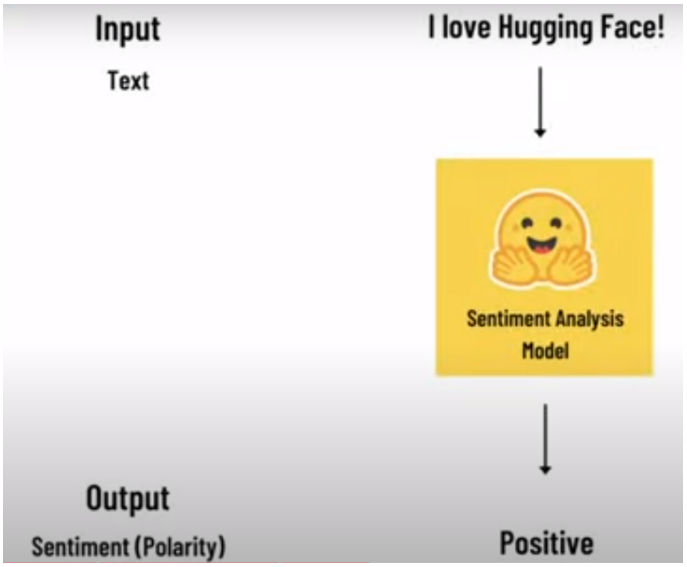
PERFORMANCE:

With 10 epochs of training:

Accuracy	Precision	Recall	F1
0.92074	0.918882	0.925824	0.92234

Transformer Model:

We also used a basic Transformer model, adapted from the Hugging Face library, for sequence classification tasks. The methodology we used covers fine-tuning a pre-trained Transformer model using the Transformers library. It also includes data pre-processing, model configuration, tokenization, model training, and evaluation. We used popular Transformer architectures like BERT and RoBERTa to achieve results in binary sequence classification tasks (with either a positive or negative classification of each comment's sentiment analysis).

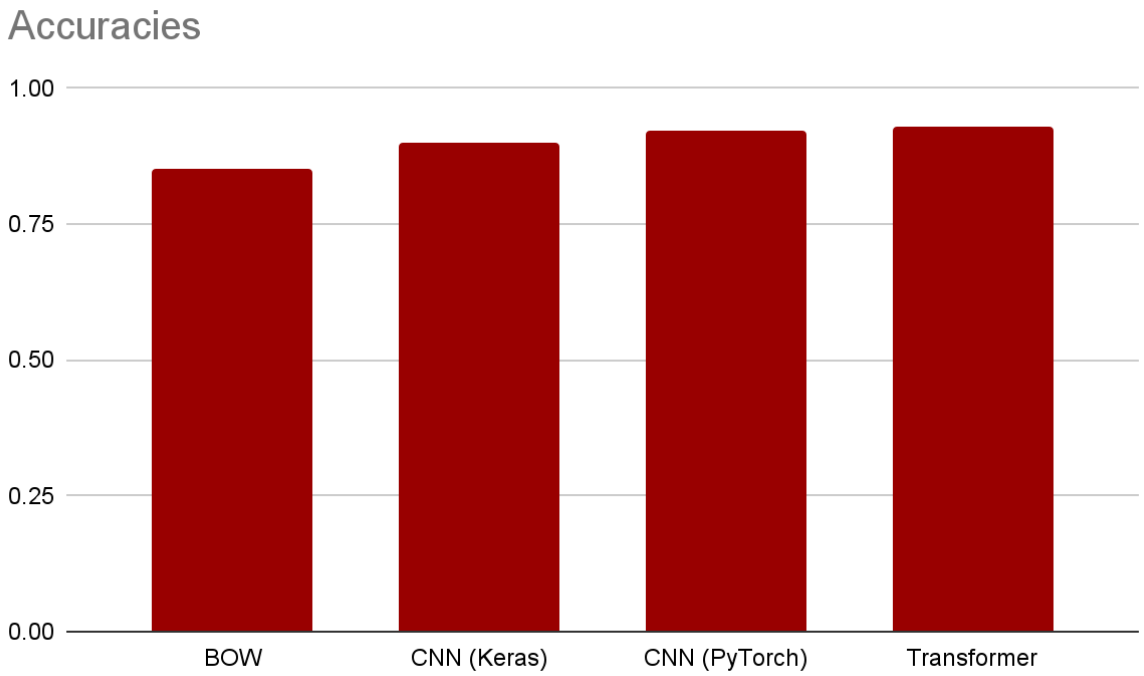


Epoch	Training Loss	Validation Loss	Accuracy

1	0.231200	0.193241	0.926120
2	0.151500	0.234670	0.929600

Comparison and Discussion of All Models

Overall, and as expected, the Transformer gave the highest accuracy for a binary classification model.



Issues we faced:

- Imbalanced dataset. We fixed this by sampling the dataset.
- Limited amount of time (9 weeks is a short amount of time to do a full research project on multiple approaches and datasets to test the effectiveness of toxic online presences), so we could not expand to more complex and better fine-tuned models.
- For multi-label classification, dataframe had imbalanced data for other classes and threats and identity_hate represented less than 5% of the data. Fixed this by working with only the toxic dataframe. However, even after this, threats were still less than 5% of the data. Used metrics like hamming distance and F1 score.
- For the Transformer, memory ended up being a challenge. Even with Google Colab, it took hours to train, but this could be mitigated by using the university's compute clusters or AWS.

Future Work

This project should look into additional language models, including more complicated Transformer models that are multi-label focused and that can have more specified hyperparameter optimization, in order to improve prediction performance.

Most of the training process of this project was done on CPU. So, by using scalable solutions such as GPU and AWS, the project would be expected to significantly reduce run-time for each model, and therefore be able to increase epochs and potential for improved results.

The project hopes to expand dataset to other sources, such as Instagram comments, Twitter comments, etc. Currently, due to shortage of time, we were able to train our models on a single dataset and so we anticipate these models will not perform very well on new datasets from different sources, such as YouTube or TikTok comments. Using adverse set of training data will improve model performance.

Finally, future work could include learning additional labels that were not in our training dataset. Such new labels could include a “sarcasm” sentiment label, which would be assigned to a comment as follows (which is currently not labelled as any of the other categories, although one could perceive sarcasm as “toxic”):

- id: "60d8df790e635658"
- comment: "Your contempt is noted. I hope I don't disappoint, however, somehow I suspect your judgement has been rendered already."
- scores: 0,0,0,0,0,0

Description of Effort

All three of us spent time reading researching papers when working on the initial proposal and incorporating instructor feedback into the mid-quarter report. While all of us were familiar with PyTorch, Irsa and Ken spent some time familiarising themselves with Keras and Hugging Face’s Transformer library (which is based in BERT, PyTorch, and TensorFlow). Each team member worked on a different model.

BOW: Yifu
CNN Binary (PyTorch): Yifu
CNN Binary (Keras): Irsa
CNN Multi (Keras): Irsa
Transformer: Ken
Poster: Ken and Yifu
Write-Ups: All

Bibliography

D. Dessì, D. R. Recupero, and H. Sack, “An Assessment of Deep Learning Models and Word Embeddings for Toxicity Detection within Online Textual Comments,” *Electronics*, vol. 10, no. 7, p. 779, Mar. 2021, doi: https://doi.org/10.3390/electronics10070779.

Duggan, M., & Smith, A. (2013). Online harassment. Pew Research Center.

H. Fan *et al.*, “Social Media Toxicity Classification Using Deep Learning: Real-World Application UK Brexit,” *Electronics*, vol. 10, no. 11, p. 1332, Jun. 2021, doi: https://doi.org/10.3390/electronics10111332.

Kaggle. (n.d.). TikTok Video Comments - David Dobrik's Top Videos. Kaggle. Retrieved from https://www.kaggle.com/datasets/jhosn13/tiktok-video-comments-david-dobriks-top-videos

Kaggle. (n.d.). Youtube Toxicity Data. Kaggle. Retrieved from https://www.kaggle.com/datasets/reihanenamdari/youtube-toxicity-data

N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences," in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, Maryland, USA, Jun. 2014, pp. 655-665, doi: 10.3115/v1/P14-1062.

S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

Suler, J. (2019). The online disinhibition effect: Understanding the psychological mechanisms of trolling. Journal of Media Psychology, 31(1), 1-12.

T. Chavan, S. Patankar, A. Kane, O. Gokhale, and R. Joshi, “A Twitter BERT Approach for Offensive Language Detection in Marathi.” Accessed: Apr. 12, 2023. [Online]. Available: https://arxiv.org/pdf/2212.10039.pdf

T. Davidson, D. Warmsley, M. Macy and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," Proceedings of the Eleventh International AAAI Conference on Web and Social Media, 2017, pp. 512-515. [Online]. Available: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665/14843. [Accessed: Apr. 12, 2023].

T. Rekatsinas, Y. Park, M. Polignano and B. Srivastava, "Using deep learning for cyberbullying detection," in Proceedings of the 2018 SIAM International Conference on Data Mining, San Diego, CA, USA, 2018, pp. 225-233, doi: 10.1137/1.9781611975321.26.

TikTok. (n.d.). Research API. TikTok Developers. Retrieved from https://developers.tiktok.com/products/research-api/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Wei, J., Liu, S., & Shang, J. (2019). Exploring Yelp Review Data with Sentiment Analysis. SMU Data Science Review. Retrieved from https://scholar.smu.edu/cgi/viewcontent.cgi?article=1134&context=datasciencereview

Zhou, P., Shi, W., Zhang, J., Huang, Q., & Chen, X. (2018). Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. Proceedings of the 27th International Conference on Computational Linguistics. Retrieved from https://aclanthology.org/W18-5105.pdf