CE807-24-SP – Assignment Report

STUDENT ID: 2312089 email: db20496@essex.ac.uk

Abstract

This report details all of my research into the project of classifying tweets as toxic or not. It shows my design choices and how I implemented my chosen models. I also critique the results I obtained during the project, and assess the project as a whole.

My research started here [4].

I will also put a warning here that trying to test my Transformer model takes over an hour, I have mentioned this in the code, and would suggest that the reader consults the given test dataset, rather than generating their own.

Materials

- Video Presentation
- Google Colab
- Google Drive

1 Task 1: Model Selection

1.1 Summary of 2 selected Models

Summary of both models and additional models implemented.

1.1.1 Transformer

This is my chosen generative model.

A transformer model is a type of neural network architecture that utilizes attention mechanisms to process data. Unlike regular neural networks, transformers are capable of processing sequential data, such as text, of any input length. The attention mechanisms enable the model to better understand the data by selectively focusing on certain parts of it. Transformers fall into one of three types, encoder, decoder, and encoder-decoder. I have used the encoder-decoder for this project. The use of a deep learning model should allow for better overall results. [6] [2] [13]

1.1.2 Logistic Regression

This is my chosen discriminative model.

A logistic regression models predict a binary outcome by assigning probabilities to the two classes, based on trends in the input data. These are somewhat basic models that can take any type of input. These are commonly used in text classification tasks. [8]

1.1.3 Additional Models research

When working on this project, I decided that researching and implementing additional models would give more results to compare. The others were a Latent Dirichlet Allocation (LDA) model, a Long Short-Term Memory (LSTM) model, and a Bayesian network model.

Firstly, LDA is a generative model that was overshadowed by my transformer, so was not selected but was implemented.

Secondly, LSTM is a discriminative deep learning model, this wasn't chosen as I couldn't complete the debugging of the implementation, but the rough code for it is provided in the .py document.

Finally, I researched Bayesian networks, as I initially believed I would use it as my generative model. I eventually decided not to, as I could not distinguish between a generative and discriminative Bayesian neural network. The references here show the potential implementation of one if I had chosen it. [7] [5]

1.2 Critical discussion and justification of model selection

Critical comments on both models and why I selected them. I tried to use deep learning as much as I could, as I know that it can be very useful in text classification tasks.

1.2.1 Transformer

I selected this as my generative model because transformers can take any length of data input, where the tweets in these datasets are of variable length. Also, there has been a large amount of other research into the effectiveness of transformers for binary classification tasks, so I trusted that one would be effective for this task. [10]

However, it did take a long time to decide on this specific model as I wanted to do deep learning, but generative models, especially in deep learning, are not particularly useful for binary text classification. This contributes to my research, and dismissal, of Bayesian networks.

1.2.2 Logistic Regression

I selected this as my discriminative model because I was originally going to use an LSTM model, but couldn't complete its implementation, and I was going to use a Logistic Regression model as my non-deep learning model for comparing performances. Logistic Regression is well suited for this task, but not as suited as deep learning, as it utilizes a sigmoid function, which returns a value between 0 and 1, and can be used to round to the closest integer, which gives a binary answer. This model can also take any inputs, so strings, tokens, or vector formats can be used. [8]

1.2.3 Additionally Implemented Models

As previously stated I implemented 2 other models. First, LDA models are used for topic modelling, which, in this case, consists of two topics (toxic and not toxic). The model considers each comment as a document and its tokens as keywords, it then learns the topics, based on collecting similar keywords. [3]

Second, LSTM models are more useful than other traditional neural networks in this task, as they are designed to handle sequential data, which text is, and avoid vanishing gradients. They are also effective in memorizing patterns in the texts/data. [9]

2 Task 2: Design and implementation of classifiers

2.1 Datasets

The following shows the data distribution of the given datasets. This will be revisited later.

Dataset	Total	Not Toxic	Toxic	% Not Toxic	% Toxic
Train	8699	7568	1131	85	15
Valid	2920	2537	383	85	15
Test	2896	n/a	n/a	n/a	n/a

Table 1: Dataset Details

2.2 Preprocessing

To preprocess correctly for this task, I implemented and tested a few methods. Firstly, I clean the tweets given by removing all unwanted tokens (such as sdata 9), making the text lowercase, and removing any special characters. This leaves the tweets in a digestible form.

Secondly, I then tokenize the tweets so that I can perform one of two methods, stemming or lemmatization. Stemming converts the most tokens into their most basic form, so was more beneficial for this task, as toxic words are still toxic in their most basic form. The functions used here are PorterStemmer and WordNetLemmatizer from NLTK.

2.3 Feature Extraction

For my deep learning models, I had to perform feature extraction to allow the data to be properly handled by the neural networks and Pytorch. Firstly, I converted the tweets to their vector format by using one of two methods, count and tf-idf vectorization, these are both functions in sklearn. Secondly, I convert the vectors into tensors that are passed into a Dataloader for the models. The tensors are formatted from lists that are the vector format of the tweet and appending the toxicity to the end, so that the tensor has a shape of the length of the vector format plus one.

2.4 Model Implementation

2.4.1 Transformer

My implementation of the transformer is adapted from code at this link and GitHub repository [10] [11] . These detail how a transformer should be constructed and how to train one.

I did have to change certain parts, namely the end of the training and validation loops as the article used a different type of dataset. The hyperparameters used are shared with the LSTM so see the later section for those.

The neural network has the following form: two embedding layers, a block (see next paragraph), two normalising layers, followed by a few more layers in sequence. These other layers use leaky ReLU and a dropout in between linear layers, that end in a softmax function.

The block is a class, defined by the GitHub, that creates the attention mechanism that makes this a transformer. This uses a MultiheadAttention, followed by two linear layers that are split by leaky ReLU layer.

2.4.2 Logistic Regression

My implementation of the logistic regression model was adapted from the sample code shown in Lab 10. This was simple as sklearn has this model, so I created a pipeline that would preprocess the data, vectorize it, and perform the logistic regression.

2.4.3 Additional Models

The implementation for the LDA model was adapted from this link [3] . Gensim has the model and the dictionary/corpus needed, so this implementation was quite simple. The model converts the training data to a corpus and uses it to create the model, then, for validation, each tweet in the validation dataset is converted to a document and the model predicts its topic.

My LSTM implementation is based on the transformer implementation [11] [13], but doesn't have the attention mechanism, and uses LSTM layers rather than linear. I did some research into what the architecture should consist of and recreated it in my model.

2.4.4 Hyperparameters

For deep learning, I defined a set of hyperparameters that were tested to find optimum results for my models. Both the transformer and LSTM use these (LSTM doesn't use all). They are as follows:

The block size is 12; embedding size is 8; number of classes in the output is 2 (binary); dropout probability is 0.1; batch size is 8; epochs trained for is 30; number of heads in MultiheadAttention is 4; head size is embedding size divided by number of heads so 2.

2.4.5 Dataset Usage

To keep this project fair, all models used ten percent of the datasets in training, because the deep learning models would have taken days to train (which I didn't have, sadly) with that much data, and I had to keep this the same for the non-deep learning models. However, they were tested with all the testing data, for the deep learning models, this takes over an hour, which means I wouldn't recommend trying this.

3 Task 3: Analysis and Discussion

This section includes a critical discussion of performance comparing all models and with SoTA.

Model	F1 Score	Accuracy Score	Recall Score	Precision Score
LDA	0.152	0.159	0.501	0.502
Transformer	0.467	0.877	0.500	0.438
Logistic Regression	0.497	0.846	0.508	0.527
LSTM	n/a	n/a	n/a	n/a
HateGAN [1]	0.532	0.782	0.522	0.652

Table 2: Model Performance Metrics

I also found this paper [12] in my research, which gives many results from multiple types of classifiers, but wasn't as useful as the HateGAN paper.

3.1 Justification of Model's performance

This section details the theoretical and practical explanations that justify the performance of my models and compares them to SoTA performances. Refer to Table 2 for performance metrics.

3.1.1 Main Explanation

For this explanation, I will only discuss the two selected models and the SoTA model, HateGAN.

The HateGAN model is a combination of three networks, a CNN, an LSTM, and a GAN. This means that the model is quite complex, and that alone justifies the performance of such a model to be better than my models. By better, I mean that it shows more desirable performance metrics. As my models only have higher accuracy scores than the HateGAN, I am led to believe that my models are overfitting for the training and validation data.

For my models only, the Logistic Regression model is better than the transformer, this is because discriminative methods are more suited for a binary text classification task, as generative are better when new data has to be generated (useful in image generation for example). However, as the transformer is a deep learning model, I assumed that it would still be better than the non-deep learning models. This either shows that my implementation of the transformer model is poorly optimized, or clarifies that generative methods are worse for binary text classification. By poorly optimized, I mean either the hyperparameters are not what they should be, the model is too small, or the data I gave the model was not in an optimum form.

3.1.2 LSTM/LDA Explanation

The LSTM has no values as I wasn't able to train or test it, due to a problem with how data was passed through layers. This is due to my time constraints, which are discussed in more detail in the summary.

The LDA model struggled to obtain good performance metrics as it looks at each tweet as a whole document, which is poor when looking for patterns and goes against the whole task of this project.

3.1.3 Revisiting Data Distribution

We will now look at the test data predicted by each model, and compare it to the distribution of data in the train and valid datasets.

Dataset	Total	Not Toxic	Toxic	% Not Toxic	% Toxic
Train	8699	7568	1131	85	15
Valid	2920	2537	383	85	15
out label model Gen	2896	125	2771	4.5	95.5
out label model Dis	2896	2796	100	96.4	3.6

Table 3: Dataset Details

This shows that the generative model has its binary classification flipped wrongly, where 0's should be 1's and vice versa. If this is the case, the generative model, as it used deep learning, is better at this task, because it found more toxic tweets.

The low percentage of tweets being toxic shows that the classifiers haven't learned all the ways a tweet can be toxic, and miss roughly 60 percent of toxic tweets. So, these classifiers have a large margin to improve.

3.1.4 Preprocessing and Feature Extraction

I tested multiple ways to perform preprocessing and feature extraction. I found that stemming in preprocessing produced better results, as stemming reduces all tokens to their most basic form, and this is useful as there will be fewer unique tokens, such that a model can learn patterns more easily without having to learn each form of toxic words. Lemmatization is slightly lenient with its reductions, so isn't as viable.

For feature extraction, I found that using a count vectorizer would be more efficient than a tf-idf vectorizer for my task. This is because tf-idf takes longer to produce an output, and models can be simpler when receiving data in a count vector form.

3.2 Example and other Analysis

Here are 5 examples that I found had interesting results.

3.2.1 Examples

Comment ID	GT	Generative	Discriminative
173094630	1	1	0
216631608	0	1	0
17067053	0	0	0
175123015	1	1	1
343828918	0	1	0

Table 4: Comparing two Models with diverse examples.

The first comment is clearly filled with toxic text, the interesting part is that the discriminative model didn't pick up on any of these toxic words.

The second comment is clearly not toxic, this interesting part is that the generative model found that it was toxic, even though none of the words would be considered toxic.

The third comment is quite short, and both models predicted correctly, showing that the length of tweets plays a part in the prediction.

The fourth comment is a large tweet that has been correctly classified by both models, this is interesting due to my previous statement.

The fifth comment was quite hard for me to decide on a ground truth, and the models have split predictions on the toxicity.

3.2.2 Other analysis

At the end of training, my transformer model gave a total training loss of 63.5, and a total validation loss of 168.9. I understand these are high, but are due to the sheer amount of data that they were trained on.

4 Summary

4.1 Discussion of work carried out

To summarize, for this project, I did plenty of research into both discriminative and generative methods. I was able to choose which models I thought were best for this task, and implemented them with decent success, as well as implement a couple other models. I was able to generate test data for both types of model. I have presented my research, along with the performances of my models and any performance metrics that I found.

To improve upon this project, I would require plenty more time and better hardware to train larger and more complex neural networks, that would produce better results. Text classification is an important subject, especially with social media, and it deserves plenty of research conducted on it.

4.2 Lessons Learned

Throughout this project, I have learned plenty of valuable information in the field of Text Analytics, especially, but not limited to, text classification. I also learned the importance of trying different methods for tasks like this, this includes different models, but also data handling approaches and saving/loading models. My newfound knowledge of pytorch will prove valuable in the future when creating neural networks from the ground up.

If I were to start this project again, I would have used my time more effectively, I spent a large amount of time working on both the LSTM and Transformer models, and only one of those works.

4.3 Closing Statements

I'm glad I had the opportunity to work on this assignment, it has given me great insight into this field, and hope to work on similar projects in the future, possibly even as a career.

I didn't have as much time as I would have liked to complete this project in, as other university projects needed my attention too. I do believe I have completed this project fairly well despite the time crunch, and hope that you can see this from my research. Thank you for reading.

References

[1] *aclanthology.org*. https://aclanthology.org/2020.coling-main.557.pdf. [Accessed 22-04-2024].

[2] *arxiv.org*. https://arxiv.org/pdf/1706.03762.pdf. [Accessed 22-04-2024].

[3] Aravind CR. *Topic Modeling using Gensim-LDA in Python — medium.com*. https://medium.com/analytics-vidhya/topic-modeling-using-gensim-lda-in-python-48eaa2344920. [Accessed 22-04-2024].

[4] *Generative models vs Discriminative models for Deep Learning. — turing.com*. https://www.turing.com/kb/generative-models-vs-discriminative-models-for-deep-learning#discriminative-model. [Accessed 22-04-2024].

[5] *GitHub - IntelLabs/bayesian-torch: A library for Bayesian neural network layers and uncertainty estimation in Deep Learning extending the core of PyTorch — github.com*. https://github.com/IntelLabs/bayesian-torch. [Accessed 22-04-2024].

[6] Himanshu. *Generative AI for Text Classification — memrhimanshu*. https://medium.com/@memrhimanshu/generative-ai-for-text-classificatio 1ceee4a0da79. [Accessed 22-04-2024].

[7] *Improve Trust in Deep Learning Models with Bayesian-Torch — intel.com*. https://www.intel.com/content/www/us/en/developer/articles/technical/improve-trust-deep-learning-models-bayesian-torch.html. [Accessed 22-04-2024].

[8] *Logistic Regression - an overview | ScienceDirect Topics — sciencedirect.com*. https://www.sciencedirect.com/topics/computer-science/logistic-regression. [Accessed 22-04-2024].

[9] Shraddha Shekhar. *What is LSTM for Text Classification? — analyticsvidhya.com*. https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/. [Accessed 22-04-2024].

[10] *Text Classification with Transformers — packtpub.com*. https://www.packtpub.com/article-hub/text-classification-with-transformers. [Accessed 22-04-2024].

[11] *tiny-transformer/classifier$_m$odel.pyatmainsaeeddhqan/tiny−transformer−−−github.co* https://github.com/saeeddhqan/tiny-transformer/blob/main/classifier_model.py. [Accessed 22-04-2024].

[12] *Toxicity Detection with Generative Prompt-based Inference — arxiv.org*. https://arxiv.org/abs/2205.12390. [Accessed 22-04-2024].

[13] *Transformer Building Blocks — packtpub.com*. https://www.packtpub.com/article-hub/transformer-building-blocks. [Accessed 22-04-2024].