Artificial Intelligence CSC4025Z: Application of Neural Networks

Furman, Gregory
FRMGRE001@myuct.ac.za

Hamilton, Joel
HMLJOE001@myuct.ac.za

Thesen, Theo
THSTHE004@myuct.ac.za

November 2021

1 Background

1.1 Problem Formulation

Gender-based violence (GBV) is a widespread issue in South Africa [1], which disproportionately affects women and girls [3]. Many victims of GBV find comfort in sharing their experience with other victims on social media (a good example of this is the #metoo movement). However, social media can be a large trigger of post-traumatic stress for GBV victims, and thus some social media users may prefer not to see social media posts with subject matter which surrounds GBV, or specific sub-categories thereof. We believe it would be useful if Twitter were to enable their Tweet-filtering algorithm (i.e. the algorithm which determines which users see which tweets on their timeline) to filter out tweets containing subject matter surrounding a specific sub-category of GBV, if a user's behaviour/interactions on the website, (or explicit preferences) have somehow indicated that they do not wish to see tweets about this given topic.

We aim to develop a neural network-based model which classifies tweets into different categories of GBV-related topics, namely *sexual violence*, *emotional violence*, *harmful traditional practices*, *physical violence* and *economic violence*, based on the content thereof.

This classification problem is non-trivial, as classification will be based on the content of the tweet, and thus it will require us to tokenise the tweet, normalise text, remove punctuation, and train our model to classify GBV Tweets based on textual content.

We believe this model will be very useful, as social media sites (in this case, Twitter) could use our tool to classify tweets into GBV-related topics as they are posted, and can immediately use this classificaiton to determine who the tweets are displayed to. We foresee that this model, along with similar ones built to support other social media sites (e.g. Facebook, Instagram, etc) will be very useful for making social media a lesser source of post-traumatic stress for victims of GBV.

Ethically speaking, while this problem deals with very sensitive and personal subject matter, there is no user-specific data in this dataset, and thus people whose tweets may be in this data set are not having any identifying information revealed, and as such, this data set has been made publicly available under a Creative Commons license 4.0 (a CC-BY SA 4.0 license). This data has been made available by Zindi for any commercial, non-commercial, research or education purposes, and given that this project falls under our university Artificial Intelligence course, our usage of this data falls firmly at the intersection of research, and education purposes.

1.2 Data

This data was obtained from `Zindi.africa`, made available under a CC-BY SA 4.0 license. It was initially used for a classification challenge/competition, and contains three fields, namely *Tweet Id*, which is simply a unique identifier for each tweet, *Tweet*, which is the text of the tweet in question, and *type*, which is the sub-category of GBV to which the content of this tweet most directly relates. The *type* variable has five categories, which are *sexual violence*, *emotional violence*, *harmful traditional practices*, *physical violence* and *economic violence*.

In terms of the tweet itself, the vocabulary size is 44456, the number of tweets in our dataset is 39650, the maximum tweet length is 148 words/tokens, and the mean tweet length is approximately 40 tokens long.

We split this dataset into three subsets, to be used for training, validation (when performing hyperparameter tuning), and testing (i.e. evaluating the final model's performance), respectively.

2 Neural Network Structure

Transformers, as a machine learning model, [6] have shown success in modelling a diverse range of high-dimensional problems at scale [2, 5, 4]. The ability to employ transfer-learning and fine-tuning also means that existing models such as Bidirectional Encoder Representations from Transformers (BERT) models and Generative Pre-Trained (GPT) can be utilised and trained in a stable manner to perform a variety of language based tasks.

We propose fine-tuning a pre-trained BERT model for sequence classification. BERT models produce bidirectional embeddings and thus, are powerful networks for comprehending the underlying context and meaning of a sequence. Furthermore, BERT models train faster than alternative sequence modelling methods, such as recurrent neural networks, as encoder inputs are processed simultaneously.

Such an architecture sees each Tweet be tokenised using a pre-trained BERT tokeniser and passed as input directly into the BERT model. BERT outputs an encoded representation of the textual input that feeds into a linear layer for classification. As the classification layer takes the BERT model's output as input, its input size is 768. In addition, the output dimension is of size 5, corresponding to each Tweet class in the dataset. The argmax of the output is then taken as the category into which the sequence is placed.

We used the pre-trained *bert-based-uncased* model along with its pre-trained tokeniser, due to the case of text being deemed unimportant to the classification task. The BERT model can take a maximum of 512 tokens as *context*. As the longest Tweet in the dataset is 148 tokens, a significant proportion of the input is context padding. Thus, we treat the context length size as a hyper-parameter, assessing the extent to which the amount of context used to classify a Tweet affects model performance.

3 Baseline

In order to compete with a BERT model, our baseline encodes sequences of natural language text, following which we classify the sequence into one of five categories. We use a long short-term memory (LSTM) network whose output feeds into a single linear layer for classification. In addition, we learn embeddings for each word in the vocabulary using an embedding layer.

To this end, current works propose LSTMs as a viable solution to the vanishing gradient problem facing recurrent neural networks (RNNs). These networks leverage cells and gates to learn long-term and relevant dependencies in sequences of data – allowing for more accurate predictions and a deeper understanding of sequence context. LSTM's learn parameters by utilizing truncated backpropagation through time (TBPP) with LSTM cells at each hidden unit that are leveraged to prevent exploding or vanishing weights occuring on long sequences. An LSTM model with fine-tuned hyperparameters should suffice as a fair baseline comparison to the BERT model.

We selected the number of hidden layers and the size of each layer via hyper-parameter tuning. In addition, we also determined the dimensionality of the embedding layer through hyper-parameter tuning. 27 plausible combinations of these hyper-parameters were built and tested for net accuracy, with the best model eventually being selected as the baseline. Further explanation and results are detailed in section 4.2.2.

4 Training & Evaluation

4.1 Evaluation

4.1.1 Metrics

As the dataset being used is heavily imbalanced, in order to fairly gauge a model's ability to classify tweets correctly, we opted to measure network performance using the macro F1 score. This metric takes the overall F1 scores corresponding to each class and finds an unweighted average. Thus, we can assess performance despite the uneven class distribution in the train, validation, and test set.

As this is a multi-class classification problem, *Categorical cross-entropy* is used to calculate a given model's loss during training.

4.1.2 Data split

The dataset contains 39.65K Tweets. Therefore, we opted for a 40/20/40 train/validation/test split using functionality from PyTorch – allowing us to split according to a random seed (42) for reproducibility. The over-representation of certain classes in the dataset and the impracticality of manually constructing a balanced sample meant that the test set required a large portion of the data to represent all classes. Hence, the test set contained a random sample of 40% of the data to evaluate the model performance. This was not an issue as 60% of the dataset is still over 23.7K Tweets. Thus, we posit this to be a sufficiently large dataset with which to train and validate the model.

4.2 Hyperparamter Tuning

Table 1: Figure showing maximum validation accuracy of BERT model when different context window sizes are used for training.

Context window size	Accuracy
50	0.9851726281
100	0.9967827141
150	**0.9989497959**

Table 2: Figure showing the results of a grid-search for hyper-parameter tuning. The maximum accuracy observed on the validation set along with the corresponding hyperparameters are shown in the table. The optimal set of parameters and corresponding accuracy are in bold.

Hidden size	Embedding dimension	Number of layers	Accuracy
128	256	1	0.9190595947
128	256	2	0.9515908526
128	256	3	0.9487973132
128	512	1	0.9179768057
128	512	2	0.9601908259
128	512	3	0.9544112131
128	1024	1	0.9244998226
128	1024	2	0.9584817531
128	1024	3	0.9538969389
256	256	1	0.9580633421
256	256	2	0.9457929649
256	256	3	0.9572513505
256	512	1	0.9388962010
256	512	2	0.9386501359
256	512	3	0.9581000086
256	1024	1	0.9735721176
256	1024	2	0.9571605772
256	1024	3	0.9592267650
512	256	1	0.9454051855
512	256	2	0.9650802021
512	256	3	0.9753192168
512	512	1	0.9506545302
512	512	2	0.9656397019
512	**512**	**3**	**0.9906050659**
512	1024	1	0.9494042489
512	1024	2	0.9544760896
512	1024	3	0.9483437402

Table 3: Figure showing final tuned hyperparameters used when training the final models. A asterisk (*) indicates that a hyperparamter value was determined via tuning.

	LSTM	**BERT**
Hyperparameters	**Value**	
Number of layers	3*	12
Number of attention heads		12
Hidden size	512*	
Embedding dimension	512*	768
Batch size	25	25
Context window		150*
Max Epochs	50	5
Learning rate	1×10^{-4}	1×10^{-5}
Adam betas	(0.9, 0.95)	
Grad norm clip	0.25	
Weight decay	0.1	
Learning rate decay	Linear warmup and cosine decay	

4.2.1 BERT

By design, our architecture's utilising of a fine-tuned and pre-trained BERT model and a single linear layer means that few hyper-parameters are available to be tuned.

The maximum Tweet length was 148 tokens with a mean and standard deviation of 39.698 and 15.127, respectively. This indicates a high level of variability in Tweet length and necessitates testing a range of context window sizes to best suit the data.

Three different models were trained with context window sizes of 50, 100, and 150 respectively. On the validation set, we found a positive association between larger context window sizes and higher validation accuracy. Hence, we the optimal context window size contained 150 tokens and corresponded to a validation accuracy of 0.9989. This makes sense considering the maximum tweet length of 148 words as a context window of 150 would be able to garner meaningful context from almost every token in every single Tweet analyzed. These results can also be seen in Table 1.

4.2.2 LSTM

To fairly compare the baseline to our original proposed model, we need to test a variety of embedding, hidden sizes, and number of layers. Hidden sizes of 128, 256, and 512 were used as well as embedding dimensions of 256, 512, and 1024. Additionally, we tested 1, 2, and 3 layers. The results seen in Table 2 indicate larger hidden sizes are associated higher validation accuracy scores. No obvious trend was observed with embedding dimension, number of layers, and accuracy. The combination of a higher hidden size in tandem with a moderately sized embedding dimension indicated superior performance to models trained with higher dimensional embeddings. The optimal combination of parameters has a hidden and embedding size of 512, with 3 hidden layers – yielding an accuracy of 0.991.

4.3 Training Details

All models were trained on Google Colab, which leverages cloud based Nvidia Tesla T4 GPUs that are specifically optimised for AI inference.

Hyperparameters used for training can be found in Table 3. Both models were trained with a batch size of 25. The BERT model ran for 5 epochs, or iterations, whereas the LSTM model ran for 50. A learning rate of 0.0001 (1×10^{-4}) and 0.00001 (1×10^{-5}) was used for the LSTM and BERT models, respectively.

At the end of every epoch, following validation, the accuracy of the current model is compared to the accuracy of the best model before said epoch. If the accuracy is higher, the current model takes it's place as the best model so far. In this way, it is ensured that we evaluate on the test set with the highest scoring iteration of the model we

have trained.

We selected Adam as the optimiser of choice since that it handles noisy data and sparse gradients well. Adam Beta values of 0.9 for the LSTM model and 0.95 for the BERT model were selected.

5 Analysis of Model Performance

5.1 Validation set (hyperparameter tuning)

The validation accuracy of the LSTM, found in Table 2, indicates near-perfect performance, achieving 0.9906. However, the BERT model achieves an accuracy of 0.9989, outperforming the baseline on the validation set. Despite comparable model accuracy on the validation set, The LSTM's evaluation time was significantly lower than the BERT model when classifying Tweets.

5.2 Test set

Table 4: Average precision, recall, and f1-score of LSTM (baseline) and BERT model for all classes when evaluated on the test set. *Macro Average* refers to an unweighted average of classification statistics.

	Macro Average		**Weighted Average**	
	LSTM	**BERT**	**LSTM**	**BERT**
precision	0.96121	0.99441	0.99712	0.99950
recall	0.94800	0.99634	0.99710	0.99950
f1-score	0.95389	0.99537	0.99709	0.99950
support	15860			

Despite similar performance on the validation set, the test results, shown in Tables tables 4 and 5, indicate the BERT model to outperform the LSTM baseline. The baseline achieved an unweighted f1-score of 0.9537. These highly accurate classification abilities indicate that the selected architecture, coupled with the chosen hyperparameters, provides a suitable baseline to improve. However, the LSTM's weighted accuracy of 0.9971, being approximately 4% higher than the previously discussed macro f1-score, is undoubtedly a result of the uneven class distribution affecting the model's performance.

The BERT model scored a macro f1-score of 0.9954 on the test set, outperforming the LSTM in both test and validation accuracy. In addition, the weighted average f1-score of 0.9995 differs from the macro score by less than 0.5%. Therefore, we can assume the BERT model's accuracy is significantly less affected by the imbalanced training dataset. Hence, BERT's performance was likely a result of its pre-existing language understanding.

5.2.1 How did class type affect accuracy?

Table 5: Precision, recall, and f1-score of LSTM (baseline) and BERT model for each class when evaluated on the test set.

	Sexual violence		**Physical violence**		**Emotional violence**		**Harmful traditional practice**		**Economic violence**	
	LSTM	**BERT**	**LSTM**	**BERT**	**LSTM**	**BERT**	**LSTM**	**BERT**	**LSTM**	**BERT**
precision	0.99854	0.99962	0.99751	1.00000	0.96899	0.99603	0.85366	0.98750	0.98734	0.98889
recall	0.99862	0.99985	0.99793	0.99834	0.99206	0.99603	0.87500	0.98750	0.87640	1.00000
f1-score	0.99858	0.99973	0.99772	0.99917	0.98039	0.99603	0.86420	0.98750	0.92857	0.99441
support	13027		2412		252		80		89	

The LSTM successfully classified Tweets containing Sexual violence, Physical violence, and Emotional violence with a high success rate of over 98%. However, on class types with less support, the LSTM's under-performance was notable. Such classes saw f1-scores of 0.8642 for Harmful traditional practice and 0.9286 for Economic violence, which indicates the LSTM's language understanding comprehension abilities failed to generalise to infrequently appearing Tweet classes.

We found BERT to have outperformed the LSTM concerning all class-specific f1-scores – achieving f1-scores of above 99% for all class types on the test set. These results indicate that a fine-tuned BERT model successfully discriminates between Tweet classes despite the heavily imbalanced dataset.

5.3 Analysis

We posit that the BERT model's pre-existing language understanding was the primary reason for its superior performance over the LSTM. Furthermore, BERT has the benefit of bidirectionality, allowing it to use context from both the left and right to classify a Tweet. This is in contrast to the LSTM, whose unidirectionality may have negatively affected classification performance.

6 Software & Tools

All code was written in Python 3.

6.1 Architecture

Our program architecture consists of a trainer, dataset, and model class with a single `main.py` as a driver class. The model class contains objects for both the baseline and BERT models. The trainer class is agnostic to the type of model it is passed and allows for the training, saving, and evaluating models. The dataset class has two different types of `dataset` objects as both BERT, and the LSTM require slightly different functionality for loading in data and retrieving data. Finally, the main/driver class takes in several command-line arguments that allow for hyperparameter selection, what type of model is to be trained, directory paths for saving/loading data, and more.

6.2 Tools

We used PyTorch for model construction in conjunction with HuggingFace's Transformers for the pretrained BERT model. Google Colab[1] was used for model training and evaluation. GitHub[2] was used for version control.

7 Reproducibility

Train Logs & Classification Report

All training logs are available in the `training_logs` directory within our code repository. For hyperparamter tuning, each log file for the LSTM is named with the following convention `[model type]_[hidden size]_[embedding dimension]_[number of layers].json`. For the BERT model, each log file is called `bert_[context window size].json`. The final model's log file is called `bert.json` with the final baseline called `baseline.json`. We also generate JSON files for the classification report for each model on the test set wherein the phrase "test_class_report" is added to the filename.

Datasets

A predetermined seed was also used for the train/validation/test split. We also attached the test data with Tweets, expected class, as well as the LSTM and BERT predictions in `data/test_set_results.csv`.

Running instructions

Instructions for training and evaluation can be found in the code repository's `README.md`.

[1] Link to Google Colab
[2] Link to GitHub

References

[1] Gender based violence in south africa. https://www.saferspaces.org.za/understand/entry/gender-based-violence-in-south-africa, 2020.

[2] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[3] Decker, M. R., Latimore, A. D., Yasutake, S., Haviland, M., Ahmed, S., Blum, R. W., Sonenstein, F., and Astone, N. M. Gender-based violence against adolescent and young adult women in low- and middle-income countries. 188–196.

[4] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092* (2021).

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008.