Detecting Offensive Language in Tweets with Attention-fused Text and User Embeddings

Viha Gupta,[1] **Casey Primel**[1]

[1] Dept. of Computer Science
vg2237@nyu.edu, ctp219@nyu.edu

Abstract

Transformer-based models have come to dominate natural language processing tasks including difficult problems in computational linguistics such as semantic and sentiment analysis. One application for these models is automated content moderation for social media platforms where they can be deployed to flag and remove abusive or harmful content. In the context of offensive language, recent research demonstrates that the performance of such models can be improved by incorporating information about the communities where that content is produced. For our project, we experiment with a novel method for incorporating community structure features and text features to classify offensive language via attention mechanisms and positional encoding. In comparison to existing published solutions, we incorporate a "corrected" procedure for computing attention on graph-structured data to produce our user embeddings and a better performing pre-trained language model for our text embeddings. Our model achieves an F1 score of **FINAL SCORE** outperforming our baseline model. Our code is available at https://github.com/guptaviha/GF-OLD.

Introduction

Content moderation is a pressing concern for social media platforms. For over a decade, automated systems have been employed to identify abusive behaviors and offensive language in user-generated content. Early systems relied on hand-crafted rules to extract linguistic features from textual data. Such rule-based systems have largely been replaced by neural network-based systems that can identify more subtle patterns in users' linguistic behaviors. However, research has shown that textual features are not always sufficient for identifying the nature of user-generated content. The rise of social media and the corollary ability to query the social graphs formed by users' activity has opened up a possible data source for constructing the requisite context. The remaining question is how to most effectively provide such context from the available data. Our project focuses on one solution in the context of Twitter: the fusion of text and user embeddings via attention mechanisms.

Related work

Recent work exploring the application of deep learning for offensive language detection can be divided into two different approaches. The first approach focuses solely on text data (e.g., tweets) and the use of pre-trained language models fine-tuned for this domain-specific task. Liu, Li, and Zou (2019). SEMANTIC ANALYSIS, DISCUSS TWEETEVAL

The second approach combines text data with information about users and community structure. Typically, this involves training two models: a deep learning model from which user embeddings can be extracted and a classification model that takes as input a text representation and the user embeddings. Qian et al. (2018) and Mishra, Yannakoudakis, and Shutova (2019) are recent exemplars of this approach. Qian et al. (2018) train a bidirectional long-short term memory network (Bi-LSTM) to produce an intra-user representation based on a users' historical behavior and a reinforced Bi-LSTM network to selectively pull inter-user representation based on similar tweets. In this case, the intra-user representations are learned prior to the training of the reinforced Bi-LSTM network. Mishra, Yannakoudakis, and Shutova (2019) train a graph convolutional network (GCN) on a heterogeneous graph of users and tweets. The output of the first layer of the GCN is then used as input, along with a bag-of-words representation of the text data for a logistic regression model. Qian et al. (2018) and Mishra, Yannakoudakis, and Shutova (2019) achieve an F1 score of 0.774 and 0.854, respectively.

Miao et al. (2022) introduce a method for learning user and text embeddings concurrently and fusing them together before via attention mechanisms. Miao et al. (2022) derive their user embeddings from a graph attention network (GAT) trained on a homogenous social graph of users and the follower relationships between them. Text embeddings are taken from a pre-trained BERT model. These embeddings are then concatenated and fused via attention mechanisms before being passed along to a feedforward classification layer. Miao et al. (2022)'s end-to-end model, which serves as the starting point for our project, achieves an F1 score of 89.94%.

Methodology

The aim of this project is to evaluate attention fusion as a means of combining text and community structure features.

In practical terms, our first objective was to replicate the results of Miao et al. (2022). With this as our baseline, we then attempted to surpass the baseline by incorporating modules which, when evaluated in isolation, outperformed the modules used by Miao et al. (2022) in their end-to-end model. Simply put, if more effective base modules were incorporated, then attention fusion would carry those improvements forward to the end-to-end model.

In the rest of this section, we introduce our dataset, describe the architecture of the end-to-end model and our modifications to it.

Dataset and preprocessing

For this project, we use the English-language dataset collected, labelled and made publicly available on GitHub by Miao et al. (2022).[1] The data was obtained via the Twitter API[2] and contains data from 1,260 users and 12,780 of their tweets posted between January 2018 and March 2021. The dataset also captures the social community in terms of the follower-network between these users with 8,877 such relationships. Unlike other Twitter datasets, Miao et al. (2022) construct a dataset to preserve community structures and better reflect real-world distribution of offensive language. They accomplish this by constructing a lexicon of topic-related words rather than offensive words from which they then query for users and their first- and second-order friends whose tweets are also collected. The result is a dataset where the ratio of offensive tweets is 7.90%, i.e., 1,009 out of the 12,780 total.

To represent the social graph, the users and relationship data are used to construct a graph structure with the users represented by nodes and the follower relationships represented by edges. Drawing on the work of Mishra, Yannakoudakis, and Shutova (2019), Miao et al. (2022) use the users' historical behavior to represent the propensity of the user to post offensive language. The number of tweets labelled non-offensive and offensive in the training data are used to describe users' historical behavior, i.e., the number of non-offensive tweets and the number of offensive tweets from the training data made by a user are each set as node attributes.

Several text preprocessing steps are taken to prepare the raw text data: emojis are converted to words with similar meanings using third-party libraries[3], URLs replaced with "*http*", hashtags segmented into phrases, and text like user references, dates and email addresses are converted into uniform placeholders, e.g., "*@anonymous2134*" becomes "*<user>*".

Model architecture

We follow Miao et al. (2022) in using a graph attention network for learning user embeddings, a language model for deriving the text embeddings and attention mechanisms for fusing user and text embeddings together before passing the fused embeddings to a classification layer. However, we introduce a modification to GAT proposed by Brody, Alon, and Yahav (2021) for computing what they call *dynamic attention*. In addition, rather than using text embeddings from BERT, we use two variations of RoBERTa language model: the base RoBERTa model (Devlin et al. 2018) and a variant pre-trained on tweets and fine-tuned for offensive language detection (Barbieri et al. 2020).[4]

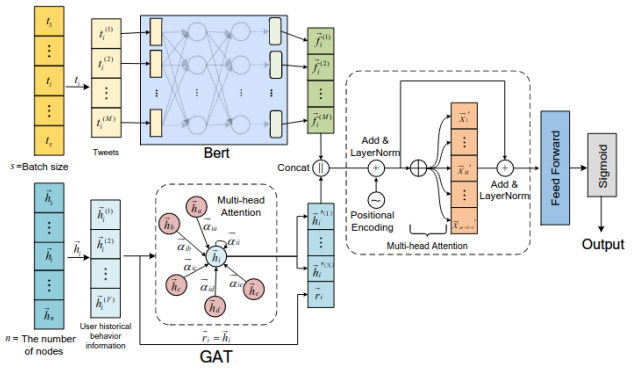

Figure 1: End-to-end offensive language detection model (Miao et al. 2022).

All the models discussed here use Adam optimizer (Kingma and Ba 2014) and focal loss function (Lin et al. 2017), a modified cross entropy loss designed for dealing with class imbalance. It introduces a scaling factor $(1-p_t)^\gamma$ such that setting $\gamma > 0$ reduces the relative loss for easy examples and puts more emphasis on hard to classify examples. The function can be written

$$FL(p_t) = \alpha \cdot (1-p_t)^\gamma \log p_t$$

where $p_t = \begin{cases} p, & \text{if } y = 1 \\ 1-p, & \text{otherwise} \end{cases}$, i.e., the magnitude of the estimated probabilty irrespective of the class label. In the special case where $\gamma = 0$, the focal loss function simplifies to cross entropy loss.

Transformer-based language models As discussed above, Miao et al. (2022)'s end-to-end model uses BERT (Devlin et al. 2018), a transformer-based language model designed for easy fine-tuning on downstream tasks(Devlin et al. 2018). Liu et al. (2019) found BERT to be significantly understrained and presented a set of improved BERT design choices and training strategies. The product of Liu et al. (2019)'s improved recipe, roBERTa, includes: (1) training the model longer, with bigger batches (8K), over more data; (2) removing the Next Sentence Prediction (NSP) loss to impove downstream tasks which require reasoning about the relationsps between sentence pairs; (3) training on longer full-length sequences; and, (4) dynamically changing the masking pattern applied to the training data compared to the single static masking performed in BERT. A byproduct of

[1] https://github.com/mzx4936/GF-OLD-Dataset

[2] https://developer.twitter.com/en/docs/twitter-api

[3] emoji (https://github.com/carpedm20/emoji) and ekphrasis (https://github.com/cbaziotis/ekphrasis).

[4] https://huggingface.co/cardiffnlp/twitter-roberta-base-offensive

(2) is particularly important for the present context: cutting NSP makes it more suitable for downstream tasks where "sentence" inputs are not part of a larger text, e.g., Tweets.

We incorporate RoBERTa into our end-to-end model as a replacement for BERT in Miao et al. (2022)'s end-to-end model keeping the model hyperparameters the same. We try both a base RoBERTa model and a variant fine-tuned for the specific task of offensive language detection in Tweets, TwitterRoBERTa (Barbieri et al. 2020). TwitterRoBERTa retrains RoBERTa-base model on a 60 million Twitter corpus. We use TwitterRoBERTa fine-tuned sepcifically for offensive language detection.[5] Barbieri et al. (2020) report that their retrained model achieved an F1 score of 81.6% on the offensive language detection task surpassing RoBERTa-base model by 1.9%.

Graph attention networks Graph attention networks (GAT) are an attention-based graph neural network (GNN) architecture for performing node classification of graph-structured data introduced by Veličković et al. (2017). Here, attention is used as a means of neighborhood aggregation wherein the hidden representation of each node in the graph is computed by attending over all its neighbors and selecting those which are most relevant. Mathematically, GAT computes a score of every edge in the graph which indicates the importance of the features of a given node's neighbor to the given node, $e(\mathbf{h}_i, \mathbf{h}_j) = LeakyReLU(\boldsymbol{a}^T \cdot [\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j])$, where both $\boldsymbol{a}$ and $\mathbf{W}$ are learned. From here, it computes a learned weighted average of the representations of a given node's neighbors followed by a nonlinearity.

As noted by Brody, Alon, and Yahav (2021), the expressiveness of GAT as formulated by Veličković et al. (2017) is constrained because the ranking of node importance computed as part of its scoring function ends up being shared by all nodes in the graph. Brody, Alon, and Yahav (2021) refer to this as *static attention* and demonstrate how static attention prevents GAT from approximating even very simple functions. To remedy this, Brody, Alon, and Yahav (2021) suggest a simple reordering of the operations in GAT resulting in what they refer to as GATv2. The reordering separates out the application of $\boldsymbol{a}$ and $\mathbf{W}$ in the scoring function so that they do not collapse into a single linear layer: $e(\mathbf{h}_i, \mathbf{h}_j) = \boldsymbol{a}^T LeakyReLU(\mathbf{W}[\mathbf{h}_i || \mathbf{h}_j])$.

Miao et al. (2022) use GAT to derive their user embeddings prior to performing attention fusion. Because GAT exhibits static attention, the user embeddings derived emphasize global community structure and de-emphasize local structures within the social graph. We hypothesize that substituting GAT with GATv2 will allow the user embeddings to better express the local structures of the social graph.

Attention fusion Attention mechanisms are used to fuse user and text embeddings derived from the graph attention network and the language model. First, the user embeddings corresponding to the current batch's text samples are selected and, then, appended to the text embeddings. The combined embeddings are positionally encoded and input into a multiheaded attention module using scaled dot-product attention as described by Vaswani et al. (2017) after which a residual connection and layer normalization are applied.

[5]See TwitterRoBERTa-base for Offensive Language Identification. https://huggingface.co/cardiffnlp/twitter-roberta-base-offensive

Training setup

Training and testing were performed via command line scripts. The scripts are largely based off of Miao et al. (2022)'s code repository.[6] with some modifications (e.g., adding and making available from the command line interface implementations of GATv2 and RoBERTa layers, and end-to-end models incorporating these layers; adding code for saving training metrics and evaluation scores to disk). Our modified fork can be found at https://github.com/guptaviha/GF-OLD. All models were trained and evaluated in Google Colab-hosted Jupyter notebooks running on a GPU instance (16 GB Tesla T4) for a maximum of 20 epochs with early stopping if the F1 score did not improve in the previous 4 epochs.

Experiments

We conducted a total of 19 experiments, including our attempts to replicate Miao et al. (2022)'s results, evaluation of isolated models before incorporation into the end-to-end model, and evaluations of different module combinations in the end-to-end model. In order to allow an apples-to-apples comparison between our different models, we use the same random seed value for the weight initialization of each experiment and maintain a consistent data split between each experiment. The latter follows the procedure used by Miao et al. (2022) and ensures a consistent distribution of offensive labels in the train and test data. We also similarly use the `ImbalancedDatasetSampler` provided in `PyTorch` to ensure that offensive labels are well-distributed across training batches.

The first experiment we undertook was an attempt to replicate the results of Miao et al. (2022) to use as a baseline for evaluating our modifications to the end-to-end model. However, we were only able to achieve

To evaluate the performance over various graph and language models, we started by attempting to reproduce the results of Miao2022 and establish a baseline joint model. We then systematically ran identical experiments on the following models

Each of our experiments was conducted using a fixed set of parameters: a batch size of 32, a GAT learning rate of 1e-2, a BERT learning rate of 5e-5, 3.5e-5, or 1e-5, a dropout rate of 0.5, a transformer attention dropout rate of 0.1, a transformer hidden dropout rate of 0.1, and a hidden vector size of 786. Each experiment ran over a total of 20 epochs with an early stopping patience of 5. Except for batch size and learning rate, all the parameters above were kept identical to Miao2022. Due to CUDA memory constraints, the batch size was reduced from 64 to 32. Following the square root rule (Krizhevsky, 2014), we also experimented with a series of BERT learning rates of 5e-5, 3.5e-5, and 1e-5.

[6]https://github.com/mzx4936/GF-OLD

Model	bs	lr	F1 score
GAT+BERT	64	5×10^{-5}	0.8893
GAT+RoBERTa	32	5×10^{-5}	0.8779
GAT+RoBERTa	32	3.5×10^{-5}	0.8898
GAT+TweetRoBERTa	32	5×10^{-5}	0.8934
GAT+TweetRoBERTa	32	3.5×10^{-5}	0.8925
GATv2+BERT	64	5×10^{-5}	0.8876
GATv2+RoBERTa	32	5×10^{-5}	0.8968
GATv2+RoBERTa	32	3.5×10^{-5}	0.8983
GATv2+RoBERTa	32	1×10^{-5}	0.8866
GATv2+TweetRoBERTa	32	3.5×10^{-5}	0.8949
GATv2+TweetRoBERTa	32	3.5×10^{-5}	0.8919
GATv2+TweetRoBERTa	32	1×10^{-5}	0.8985

Table 1: Selected combinations of different graph attention networks, language models and learning rates (lr), trained for 20 epochs with early stopping ($patience = 5$). Batch size (bs) was only modified in order for the given model to fit in GPU memory.

	Baseline GAT+BERT	Final GATv2 + TweetRoBERTa
Mean F1	0.8927	0.8936
Max F1	0.9018	0.012
Min F1	0.7132	0.8230
SD F1	0.0204	0.0114

Table 2: Selected combinations of different graph attention networks, language models and learning rates (lr), trained for 20 epochs with early stopping ($patience = 5$). Batch size (bs) was only modified in order for the given model to fit in GPU memory.

For experiments that performed well, we ran a series of 10 experimental runs of 20 epochs each, across different seeds to evaluate the best and average F1 scores.

DESCRIBE EXPERIMENTS RUN AND RATIONALE.

Results

For our final results, ...

Discussion

Our final model is able to achieve a test accuracy of SCORE and an F1 score of SCORE on Miao et al. (2022) Twitter dataset surpassing our baseline by XYZ and XYZ, respectively. There are several avenues for further exploration and possible improvement that we can identify. ERROR ANALYSIS. The first would be a more thorough comparison of models that incorporated hyperparameter tuning on a per-model basis, especially in regard to the application of different learning rates and regularization. The second is to revisit the construction of the social graph and the derivation of the user embeddings to evaluate how well they are capturing community structure. In particular, one could use non-parametric methods for community detection as a baseline against which to measure how effectively the graph attention layer as capturing community features.

References

Barbieri, F.; Camacho-Collados, J.; Neves, L.; and Espinosa-Anke, L. 2020. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification.

Brody, S.; Alon, U.; and Yahav, E. 2021. How Attentive are Graph Attention Networks?

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Liu, P.; Li, W.; and Zou, L. 2019. NULI at SemEval-2019 Task 6: Transfer Learning for Offensive Language Detection using Bidirectional Transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, 87–91. Minneapolis, Minnesota, USA: Association for Computational Linguistics.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.

Miao, Z.; Chen, X.; Wang, H.; Tang, R.; Yang, Z.; and Tang, W. 2022. Detecting Offensive Language on Social Networks: An End-to-end Detection Method based on Graph Attention Networks.

Mishra, P.; Yannakoudakis, H.; and Shutova, E. 2019. Tackling Online Abuse: A Survey of Automated Abuse Detection Methods.

Qian, J.; ElSherief, M.; Belding, E.; and Wang, W. Y. 2018. Leveraging Intra-User and Inter-User Representation Learning for Automated Hate Speech Detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 118–123. New Orleans, Louisiana: Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph Attention Networks.

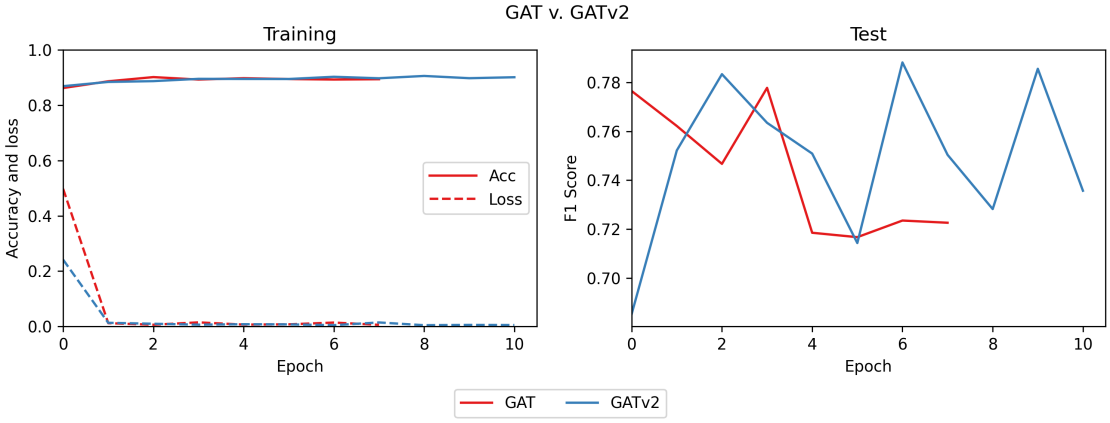

Figure 2: Comparison of model training accuracy, training loss and test F1 score for GAT and GATv2.

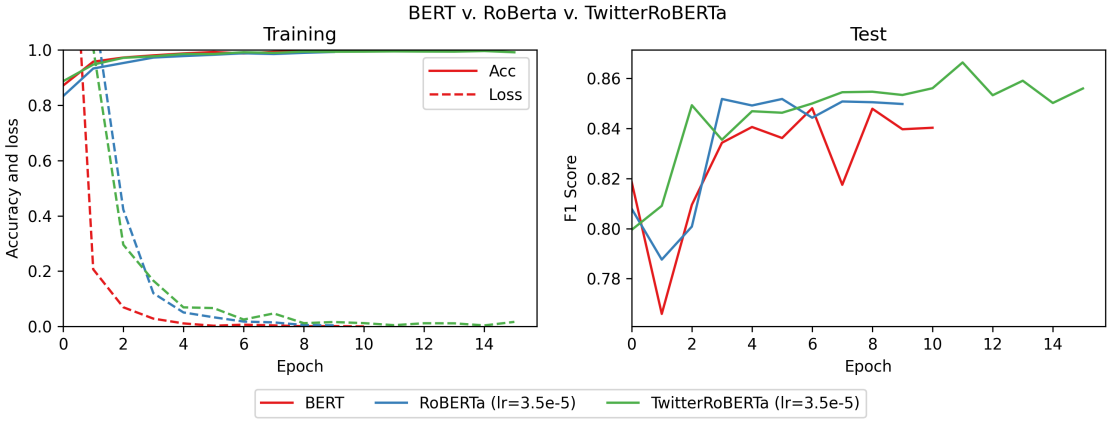

Figure 3: Comparison of model training accuracy, training loss and test F1 score for BERT, RoBERTa and TwitterRoBERTa.

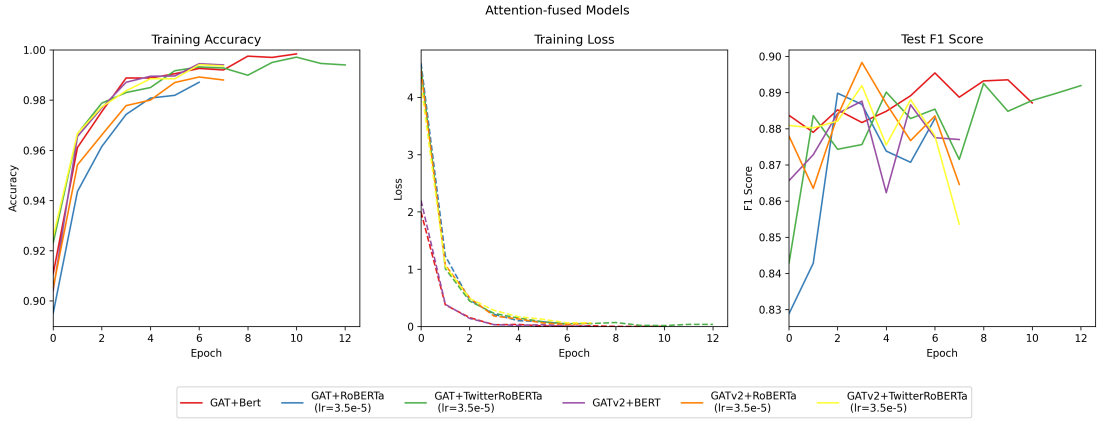

Figure 4: Comparison of model training accuracy, training loss and test F1 score for all attention-fused models.