Transfer Learning with Augmented Vocabulary for Tweet Classification

(Grand Challenge)

Vineet Kumar, Karthikeya Racharla
Indian Institute of Technology
Kharagpur, WB, India, 721302
vineet.kumar@iitkgp.ac.in
karthikeya.r@iitkgp.ac.in

Debapriyo Majumdar
Indian Statistical Institute
CVPR Unit, 203 B.T. Road,
Kolkata, WB, India, 700108
debapriyo@isical.ac.in

***Abstract*—In this paper, we describe our experiments and insights gathered in the process of participating in the BigMM Grand Challenge 2020, where the task was to classify a set of tweets pertaining to the #MeToo movement into five linguistic aspects. We analyzed the data set and experimented with several approaches, including classical machine learning models and state of the art deep learning architectures. We achieved our best results by applying transfer learning on a pre-trained ULMFiT model. Our best performing approach had ranked first on the leader-board when the grand challenge finished.**

***Keywords*-MeToo Movement, Tweet Classification, Transfer Learning, Augmented Vocabulary, ULMFiT**

I. Introduction

The #MeToo movement is a social movement against sexual harassment and abuse, where people publicize allegations of sexual crimes committed by powerful men. The movement garnered meaningful conversations in social space around sexual abuse and harassment. With the popularity of several social channels, automatic detection of targeted sentiment spread over social media has become very important. Some examples of such applications were presented in [1], [2] and [3]. The MeToo Grand Challenge (BMGC), aimed at encouraging research towards deeper understanding of multiple facets of the #MeToo movement, was organized as a part of IEEE BigMM 2020.

The dataset released for the challenge [4, 5, 6] contained a total of 9,973 tweets with hashtag MeToo and were manually annotated into 5 linguistic aspects, namely – Relevance, Hate Speech, Sarcasm, Dialogue Acts and Stance towards #MeToo movement. The task was to develop multi-task frameworks aimed at predicting the labels corresponding to a given tweet. One may consider these aspects as tasks and can develop specialized models tailored for individual tasks. The evaluation metric for the challenge was the **A**rea **u**nder the ROC **C**urve (AUC), where the ROC curve (receiver operating characteristic curve) plots the true-positive rate against the false-positive rate at different classification thresholds.

This paper describes the experiments conducted by us and the insights we gained as we worked on the task. We experimented with classical machine learning models, tree-based approaches, as well as deep learning methods and appropriately combined them. Our combined model achieved the best AUC of 0.56365 for the task and ranked first [7] in the competition.

II. The Dataset

Due to Twitter legal requirements, the dataset [6] shared by challenge organizers only had tweet-ids and their corresponding labels, using which participants had to download those tweets directly. Technically the process is known as tweet hydration.

The dataset was provided into two sets, namely the training set with 7,978 tweets and test set with 1,995 tweets. The test dataset was released later in the course of the challenge. Upon submission of the results, the leaderboard showed the evaluation on only 70% of the test set (as a development set), allowing the participants to tune their approaches further, but the final results were disclosed after the challenge was over.

Tweet Hydration: We used Hydrator [8], open-source software to hydrate our tweets. Due to the hostile nature of tweets, we were able to hydrate only 6,869 tweets (rest 14% tweets were deleted) from the train set and 1,737 tweets (rest 13% tweets were deleted) from the test set.

The Hydrator has enabled us to also extract other relevant information related to specifics of a tweet, such as – favorites count, retweet count, flag signifying whether tweet is sensitive, and information whether the person's Twitter profile is verified or not.

Derived Labels: We created 2 flags based on the existence of labels, namely *None_all_classes*, a flag which is 1 when all other labels are zeroes and *None_wo_support*, flagged 1 when all other labels except *Support* are zeros. Interestingly, in the overall training data, there is only a single tweet having *Support* as 1, rest all classes as 0. Label pairs (*Support*, *Oppose*) and (*Directed_Hate*, *Generalized_Hate*) are mutually exclusive, whereas labels *Text_Only_Informative* and *Image_Only_Informative* that represent the relevance of text and image with tweets are not exclusive. Among the 6,869 train data tweets that we were able to hydrate, 26% of tweets have both image and text as informative, which means that when both image and text are taken together, they are related and informative. 20% tweets state none of these two labels as informative. 73% of tweets say that only text is informative, and 33% state that only image is informative.

We observed severe class imbalance with most of the labels. The distribution of positive samples per class is shown in the Table I. To further check the interaction effects between the labels, we grouped the existing labels into 4 derived labels: *Stance* – to indicate if at least any one of *Support* or *Oppose* is flagged 1 with the corresponding tweet. Similarly – *Hate* to indicate the association of either *Directed Hate* or *Generalized Hate*, *Dialogue acts - Allegation, Refutation, Justification* that were specific to the MeToo movement and *Relevance* implying the relevance of text and image labels on tweets. Over 90% of tweets did not have any dialogue acts associated with them.

Label	Positive Samples	% Positive
Text_Only_Informative	4,989	72.63
Image_Only_Informative	2,271	33.06
Directed_Hate	257	3.74
Generalized_Hate	194	2.82
Sarcasm	146	2.13
Allegation	365	5.31
Justification	230	3.35
Refutation	143	2.08
Support	2,189	31.87
Oppose	505	7.35

Table I
NUMBER OF POSITIVE LABELLED TWEETS

III. PREPROCESSING

Many of the images associated with the tweets contain text. We initially planned to convert the images to text using OCR and use the output in our experiments, but due to time constraints, we could not. Hence we restricted ourselves to only the textual part of the tweets.

A. Data Cleaning

We cleaned the tweets through the following steps:

1) All tweets were converted to lower-case.
2) Removed punctuations, extra spaces, extra newlines and irrelevant characters.
3) Replaced URLs in the tweet with the word 'URL', #hashtag with hashtag and @ handle with the special word 'USER_MENTION'.
4) Removed 'RT' character (retweet).

We did not remove stop-words because their removal hampers the performance of sentiment classification as empirically investigated by Saif et al. [9].

B. Converting Texts to Features

Vectorization refers to the process of transforming the collection of texts in a corpus to a numerical representation of feature vectors. All the terms in a corpus, may not be of equal importance – that's when we use original term-weighting based vectorization approaches like TF–IDF. Using the `TfidfVectorizer()` method, we converted the raw tweets into representable TF-IDF matrix forms. This is equivalent to using CountVectorizer, computing the word counts systematically, followed by TfidfTransformer, to compute Inverse Document Frequency (IDF) to obtain TF-IDF scores. We also created an indexed dictionary of all the tokens in our tweet corpus, then vectorized each token by turning it into sequences of integers using indices of the token dictionary. The coefficients corresponding to each token were extracted using these approaches. Obviously, all the tweets are not supposed to have the same lengths; hence we padded these tweets with zeroes, setting the maximum length. Keras provides the in-built `Tokenizer()`[1] class for these demonstrations.

IV. OUR APPROACHES AND EXPERIMENTS

Due to the overlap between tweets corresponding to different labels, we considered the task a combination of several different sub-tasks, one for each label. In other words, given a tweet, for each label, our models determine whether the tweet belongs to the label or not. In the initial phase of the challenge, we could use only the training set and used a part of it for validation. We tried with some classical and recent tree-based models and then explored various modern deep learning methods. Our codes are hosted at Github[2].

A. Classical ML Models

We experimented with Logistic Regression, Multinomial Naïve Bayes and Support Vector Machine (SVM) initially. However, these methods did not produce encouraging results, although SVM is known to learn well irrespective of dimensionality of the feature space.

Having tried tree-based models such as RandomForest ensemble and stacked boosting classifiers such as XGBoost [10] and LightGBM [11], we observed some improvement in the AUC score using XGBoost, probably because of its ability to address class imbalance while training. The default parameter `scale_pos_weights` controls high class-imbalance by leveraging the balance between positive and negative instances. Albeit there was a risk of over-fitting and the inability to generalize, a higher number of false-positives were obtained in the validation data. Through this approach, we were able to get mean column-wise AUC on the leader-board close to 0.5182. Table III shows the results from some of these classical models. We performed these experiments using scikit-learn [12].

[1]https://keras.io/api/preprocessing/text/#tokenizer-class
[2]https://github.com/vntkumar8/transfer-learning-metoo

B. Bi-LSTM

The Long Short Term Memory (LSTM) [13] has been one of the most popular and successful recurrent neural network approach in tasks related to text data due to its ability in preserving long-term dependency in texts. Bidirectional LSTMs [14] extend the traditional LSTMs and are generally proven to work better with sequential classification problems [15]. We implemented BiLSTM using the Bidirectional layer wrapper supported via Keras [16]. We made use of the pre-trained glove-twitter embeddings [17] which were trained over 2B uncased tweets, containing 27 billion tokens and 1.2M vocab. For our BiLSTM model, the weights from our glove embeddings were loaded. We initially set the 'trainable' parameter to 'False' as a measure to check if the model improves while using the same word embeddings. As a regularization step, we introduced Dropout [18] layer that works with some input probability by dropping random nodes in the network layer, hence improving the generalization while learning. We also introduced Conv1D filters with filter length 5 for representing the context from neighboring words of the embeddings. Overall, the AUC was not improving substantially using BiLSTM - only *Support* and *Oppose* classes performed relatively well. For other highly imbalanced classes, the results were poor.

C. Transfer learning with BERT

In recent years, transfer learning approaches with pre-trained BERT [19] models have achieved tremendous success in NLP tasks. With the inspiration, we implemented BERT using PyTorch Pretrained BERT[3]. We used `bert-base-uncased` 12 transformers version for our model from huggingface library. We used BERT tokenizer to add special tokens at the start & end of each sentence; Pad & truncate each sentence to a specific length (192 in our case) and explicitly differentiate real tokens from padding tokens with the "attention mask". The attention mask makes it explicit that which tokens are actual words and which are padding. We used `BertForSequenceClassification` class for our purpose. For training, we used 2×10^{-5} as our learning rate, 10^{-8} as eps value, and fine-tuned with AdamW optimizer for three epochs. Training finished quickly with 67% overall accuracy on the validation set for 20% split from training data itself. However, when we calculated AUC and subsequently analyzed errors, we found that BERT can not perform well - most of the words in the vocabulary of our dataset were unrecognised by BERT. We tried to solve the high class-imbalance by minority class oversampling, but that did not improve the results. We have presented the results from BERT and BERT with minority class random oversampling (BERT + ROS) in Table IV.

[3]https://huggingface.co/transformers

D. Transfer learning with ULMFiT

The textual content in the tweets is vastly different from the text most pre-trained models are trained on. Most of the tweets had at least one user mention and one URL; apart from that, few words were poorly typed with multiple typographic mistakes, slangs, and abbreviations. To counter these problems, Howard and Ruder [20] proposed ULMFiT, which uses the AWD-LSTM [21], that can be applied to NLP tasks with dramatic improvement in accuracy and efficiency.

The fundamental idea behind ULMFiT to tackle the problem of insufficient training examples is Inductive Transfer Learning[22]. After pre-training a general language model, we fine-tuned ULMFiT through the following two steps:

1) Target task Language Model fine-tuning: According to Yosinski et al. [23] different layer captures different types of information; thus, they should be fine-tuned differently as well. Discriminative fine-tuning allows tuning each layer with different learning rates.

$$\theta_t^l = \theta_{t-1}^l - \eta^l \nabla_{\theta^l} J(\theta)$$

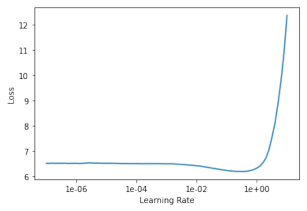
   where $\nabla_{\theta^l} J(\theta)$ is gradient w.r.t. objective function, η^l is learning rate for $l-$th layer.
2) Target task model fine-tuning: Once we have learned the target LM, we can fine-tune the classifier with gradual unfreezing to perform task-specific class prediction. The *Gradual Unfreezing* recommends first to unfreeze the last layer and fine-tune all the unfrozen layers for one epoch. Then unfreeze the next lower frozen layer and repeat, till we fine-tune all layers until convergence at the last iteration.

1) Experiment & Methodology for Training: We used ULMFiT for *stance detection* in our task. Stance detection in our context means detecting whether a tweet is in favour or against the #MeToo movement. We created a new label Stance, and it has three possible levels based on the following rule-set.

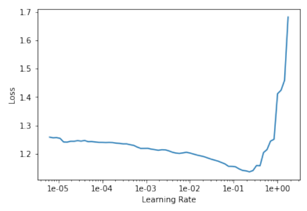
Support	$\bigwedge$	Oppose	Stance
0		0	None
1		0	Favour
0		1	Against

We downloaded the ULMFiT language model, which is pre-trained on the wikitext-103 corpus and contains 103 million words from Wikipedia, from the fast.ai database hosted on AWS[4]. We followed the approach described in Section IV-D. We used `fastai.text` tokenization technique, which tokenizes the target vocabulary efficiently and divides it into appropriate batches. One important thing to note here is that we don't need to pre-process tweets at all. Just converting the tweets to plain ASCII serves the purpose, and no pre-processing (like removal of URL, hashtags) is

[4]https://s3.amazonaws.com/fast-ai-modelzoo/wt103-fwd.tgz



(a) LR for LM Fine-tuning



(b) LR for Target Task

Figure 1. Determining Learning Rate

essential. We used `language_model_learner`[5] class to specify the data language model and pre-trained model. After that, we followed all the steps of optimum learning rate by gradual unfreezing. However, the results achieved from this model were not much helpful; we achieved AUC of 0.51 for *Oppose* and 0.52 for *Support*, respectively.

2) Augmented Vocabulary: Despite fine-tuning both the language model and the classifier, we could not achieve satisfactory performance using ULMFiT, possibly because of our small set of training examples. Moreover, the language syntax of tweets is quite different from the Wikipedia text. To overcome this issue, we tried fine-tuning the Language model itself on a large twitter corpus.

Our training data has only 6,869 tweets; hence we need more tweets data. To aid our efforts, we included two more such corpora, namely Kaggle Twitter Sentiment 140[6] and SEM Eval Stance Detection Dataset[7]. After including these two data sets with our tweet train data, we got almost 201K tweets. We tokenized this large text corpus, divided it into appropriate batches using `TextLMDataBunch`. We fed this dataset into our LM (language model), which is essentially AWD-LSTM. The fast.ai guideline recommends using learning rate that is an order of magnitude below the point at which the loss starts to diverge. The learning rate for our language model is shown in Figure 1(a). For our purpose 1×10^{-3} is an appropriate choice of LR, as from 1×10^{-2} onwards, the loss starts to diverge.

To train the language model during fine-tuning, we chose to unfreeze all the layers and then perform training for five epochs due to our computational constraints. The gradual unfreezing helps in avoiding "catastrophic forgetting", as explained in [20]. This step took significantly more time (1 hour in our case using Tesla K80 GPUs). After fine-tuning, we saved the encoder and the vocabulary for future use.

Next, we fine-tuned our classifier for stance detection, our target task. Just as before, we plot the learning rate curve in Figure 1(b) and determined that the optimal learning rate should be between 1×10^{-3} and 1×10^{-4}. We chose 1×10^{-3}, beyond which the loss diverges. While training, we chose to unfreeze our layers gradually.

[5]https://docs.fast.ai/text.learner.html#language_model_learner
[6]https://www.kaggle.com/kazanova/sentiment140
[7]https://www.saifmohammad.com/WebPages/StanceDataset.htm

epoch	**train_loss**	**valid_loss**	**accuracy**	**time** (in sec)
0	4.592715	4.344318	0.250712	637
1	4.383563	4.187261	0.265000	637
2	4.192356	4.110231	0.270790	638
3	4.029832	4.088607	0.274656	638
4	3.928093	4.095024	0.274406	637

Table II
UNFREEZING THE ENTIRE MODEL

Testing our classifier on the validation set produced good results, as shown in Table IV under column name (ULMFiT+AV : where AV stands for Augmented Vocabulary).

V. RESULTS

We chose our final model for each class based on the AUC score obtained over our validation set. For example, our best submission for *Image_Only_Informative* is taken from XGBoost, as it has the highest AUC, among other competing models. Best AUC Score for each label is shown in **bold** in Table III and IV.

Labels/Models	MNB	SVM	RF	LGBM	XGB
Text Only Informative	0.506	0.517	0.505	0.500	0.510
Image Only Informative	0.499	0.509	0.501	0.500	**0.539**
Directed Hate	0.500	0.510	0.503	0.491	0.503
Generalized Hate	0.500	0.524	0.508	0.513	0.524
Sarcasm	0.500	0.496	0.499	0.485	**0.535**
Allegation	0.500	0.515	**0.515**	0.491	0.472
Justification	0.500	0.504	0.508	0.495	**0.505**
Refutation	0.500	0.546	0.565	0.510	**0.578**
Support	0.496	0.496	0.499	0.504	0.511
Oppose	0.500	0.514	0.511	0.498	0.523

Table III
LABEL WISE AUC SCORES FROM VARIOUS ML MODELS

We tried deep learning based methods on 4 labels. The AUC-ROC score for each label is shown in Table: IV.

Labels/Models	BiLSTM + Glove	BERT	BERT + ROS	ULMFiT	ULMFiT + AV
Support	0.50	0.49	0.48	0.52	**0.58**
Oppose	0.49	0.49	0.51	0.51	**0.73**
Directed Hate	**0.52**	0.51	0.51	0.50	0.50
Generalized Hate	**0.51**	0.51	0.50	0.51	0.49

Table IV
LABEL WISE AUC SCORES FOR VARIOUS DL MODELS

VI. CONCLUSION AND FUTURE WORK

Although we achieved the best AUC score in the grand challenge, several improvements are possible as future work. As we pointed out in Section III, we have not made use of the images in the tweets. The images containing text can be used after an OCR conversion. More sophisticated approaches may include the use or invention of advanced architectures that work on image and text together.

References

[1] R. Mishra, P. Prakhar Sinha, R. Sawhney, D. Mahata, P. Mathur, and R. Ratn Shah, "SNAP-BATNET: Cascading author profiling and social network graphs for suicide ideation detection on social media," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, (Minneapolis, Minnesota), pp. 147–156, Association for Computational Linguistics, June 2019.

[2] P. P. Sinha, R. Mishra, R. Sawhney, D. Mahata, R. R. Shah, and H. Liu, "#suicidal - a multipronged approach to identify and explore suicidal ideation in twitter," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, (New York, NY, USA), p. 941–950, Association for Computing Machinery, 2019.

[3] M. Agarwal, M. Leekha, R. Sawhney, and R. R. Shah, "Crisis-dias: Towards multimodal damage analysis-deployment, challenges and assessment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 346–353, 2020.

[4] A. Ghosh Chowdhury, R. Sawhney, R. R. Shah, and D. Mahata, "#YouToo? detection of personal recollections of sexual harassment on social media," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2527–2537, Association for Computational Linguistics, July 2019.

[5] A. Ghosh Chowdhury, R. Sawhney, P. Mathur, D. Mahata, and R. Ratn Shah, "Speak up, fight back! detection of social media disclosures of sexual harassment," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 136–146, Association for Computational Linguistics, June 2019.

[6] A. Gautam, P. Mathur, R. Gosangi, D. Mahata, R. Sawhney, and R. R. Shah, "#metooma: Multi-aspect annotations of tweets related to the metoo movement," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, pp. 209–216, May 2020.

[7] *IEEE BigMM Grand Data Challenge*, (accessed August 7, 2020). : Kaggle Leader Board : https://www.kaggle.com/c/ieee-bigmm-data-challenge.

[8] Hydrator, *Documenting the Now*, (accessed August 6, 2020). https://github.com/docnow/hydrator.

[9] H. Saif, M. Fernandez, Y. He, and H. Alani, "On stopwords, filtering and data sparsity for sentiment analysis of Twitter," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, (Reykjavik, Iceland), pp. 810–817, European Language Resources Association (ELRA), 2014.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, pp. 3146–3154, 2017.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[15] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[16] F. Chollet *et al.*, "Keras," 2015. https://keras.io.

[17] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

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

[20] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[21] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," *arXiv preprint arXiv:1708.02182*, 2017.

[22] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.