WLV-RIT at HASOC-Dravidian-CodeMix-FIRE2020: Offensive Language Identification in Code-switched YouTube Comments

Tharindu Ranasinghe[a], Sarthak Gupte[b], Marcos Zampieri[b] and Ifeoma Nwogu[b]

[a]University of Wolverhampton, UK
[b]Rochester Institute of Technology, USA

Abstract

This paper describes the WLV-RIT entry to the Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC) shared task 2020. The HASOC 2020 organizers provided participants with annotated datasets containing social media posts of code-mixed in Dravidian languages (Malayalam-English and Tamil-English). We participated in task 1: Offensive comment identification in Code-mixed Malayalam Youtube comments. In our methodology, we take advantage of available English data by applying cross-lingual contextual word embeddings and transfer learning to make predictions to Malayalam data. We further improve the results using various fine tuning strategies. Our system achieved 0.89 weighted average F1 score for the test set and it ranked 5^{th} place out of 12 participants.

Keywords

offensive language identification, hate speech, text classification, code-switching

1. Introduction

Offensive content is pervasive in social media putting users of various platforms at risk [1]. The pervasiveness of such content motivated the development of several systems capable of identifying offensive posts in a number of languages [2, 3]. Once identified, these posts can be then set aside for human moderation or deleted from online platforms mitigating risks to their users [4].

Recent studies have addressed many types of offensive content such as online abuse [5, 6], aggression [7], cyberbullying [8, 9], and hate speech [10, 11]. International workshops and competitions such as HatEval 2019 [12], OffensEval 2019 and 2020 [13, 14], co-located with SemEval, have been organized in the last two years attracting a large number of participants. Most high performing system in these competitions used neural networks and contextual word embeddings such as BERT [15].

In this paper we describe the WLV-RIT entry to the the HASOC 2020 shared task which featured Malayalam-English code-switched data. Building on the experience of recent high

FIRE 2020: Forum for Information Retrieval Evaluation, December 16-20, 2020, Hyderabad, India
T.D.RanasingheHettiarachchige@wlv.ac.uk (T. Ranasinghe); sg7179@rit.edu (S. Gupte); marcos.zampieri@rit.edu (M. Zampieri); ion@cs.rit.edu (I. Nwogu)

performing models submitted to the OffensEval competitions, we use a transformer-based architecture described in detail in Section 3. The HASOC 2020 code-switching dataset is a particularly challenging one for state-of-the-art offensive language detection systems and we make use of transfer learning techniques that have been recently applied to project predictions from English to resource-poorer languages with great success [16].

2. Task description and Datasets

The goal of this task is to identify the offensive language of the code-mixed dateset of comments/posts in Dravidian Languages (Tamil-English and Malayalam-English) collected from social media. Each comment/post is annotated with an offensive language label at the comment/post level. The dataset has been collected from YouTube comments [17]. We participated in task 1 which is a message-level label classification task; given a YouTube comment in Code-mixed (Mixture of Native and Roman Script) Tamil and Malayalam, systems have to classify whether a post is offensive or not-offensive. To the best of our knowledge, this is the first dataset to be released for offensive language detection in Dravidian Code-Mixed text [17].

In addition to the dataset provided by the organisers we also used an English Offensive Language Identification Dataset (OLID) [18] used in the SemEval-2019 Task 6 (OffensEval) [13] for transfer learning experiments which are describing in Section 3. OLID is arguably one of the most popular offensive language datasets. It contains manually annotated tweets with the following three-level taxonomy and labels:

A: Offensive language identification - offensive vs. non-offensive;

B: Categorization of offensive language - targeted insult or thread vs. untargeted profanity;

C: Offensive language target identification - individual vs. group vs. other.

We adopted the transfer learning strategy similar to previous recent work [16]. We believe that the flexibility provided by the hierarchical annotation model of OLID allows us to map OLID level A (offensive vs. non-offensive) to labels in the HASOC Malayalam-English dataset.

3. Methods

The methodology applied in this work is divided in two parts. Subsection 3.1 describes traditional machine learning applied to this task and in Subsection 3.2 we describe the transformer models used.

The motivation behind our methodology is the recent success that the transformers had in wide range of NLP tasks like language generation [15], sequence classification [19, 20], word similarity [21], named entity recognition [22], question and answering [23] etc. The main idea of the methodology is that we train a classification model with several transformer models inorder to identify offensive texts. However, the transformer models are known to be resource intensive requiring fairly large datasets [15], therefore, we also experimented with several traditional machine learning models

3.1. Traditional Machine Learning Methods

In the first part of the methodology, we used traditional machine learning models. We experimented with three models; Multinomial Naive Bayes [24], Support Vector Machines [25], and Random Forest [26]. The models take an input vector created using Bag-of-words and outputs a label, either offensive or non-offensive. The models for Multinomial Naive Bayes, SVM and Random Forest were implemented using the Scikit-learn [27].

Data Preprocessing We performed three preprocessing techniques; removing punctuations, removing emojis and lemmatising the English words. This was done with the use of the NLTK (Natural Language Toolkit) library [28] in Python.

Hyper Parameter Optimisation Optimisation of hyper parameters was performed on SVM and random forest only. For SVM, the hyper parameters fine-tuned were alpha, random state and max iteration, where alpha represents regularisation, random state is used for shuffling of the data and max iteration denotes number of passes through the training data which is also known as epochs. Optimal values achieved were alpha=0.001, random state=5, max iteration=15. For random forest, only one hyper parameter was used which is n-estimator that denotes number of decision trees created. Optimal value achieved for number of trees was 500.

3.2. Transformers Models

As the second part of the methodology, we used Transformer models. Transformer architectures have been trained on general tasks like language modelling and then can be fine-tuned for classification tasks [29]. They take an input of a sequence and outputs the representation of the sequence. The sequence has one or two segments that the first token of the sequence is always [CLS] which contains the special classification embedding and another special token [SEP] is used for separating segments. For text classification tasks, Transformer models take the final hidden state $\mathbf{h}$ of the first token [CLS] as the representation of the whole sequence [29]. A simple softmax classifier is added to the top of the transformer model to predict the probability of a class as shown in Equation 1 where W is the task-specific parameter matrix. The architecture diagram of the classification is shown in Figure 1

$$p(c|\mathbf{h}) = softmax(W\mathbf{h}) \tag{1}$$

Transformers We experimented two pretrained transformer models; BERT [15] and XLM-ROBERTA [30] We used the HuggingFace's implementation of the transformer models [31] and the pre-trained models available in the HuggingFace model repository.[1] These models were used mainly considering their support to Malayalam language. For BERT we used the BERT multilingual model (BERT-M) and for XLM-ROBERTA (XLM-R) we used the XLM-R-Large model. Both models support 104 languages including Malayalam. The interesting fact about XLM-R is that it is very compatible in monolingual benchmarks while achieving best results in cross-lingual benchmarks at the same time [30].

[1]HuggingFace model repository - https://huggingface.co/models

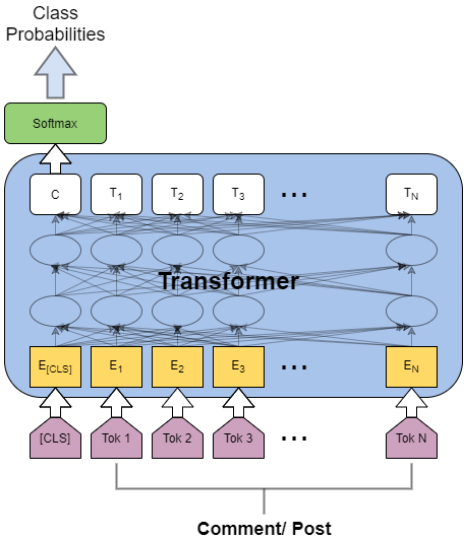

Figure 1: Transformer Text Classification Architecture

Transfer Learning The main idea of the transfer learning strategy is that we train a classification model on a resource rich language, typically English, using a transformer model and perform transfer learning on a less resource language. We trained the classification model on the first level of OLID [32] and then we save the weights of the transformer model as well as the softmax layer. We use this saved weights from English to initialise the weights when we are training the classification model for Malayalam. This strategy has improved the performance of different languages with less resources for offensive language identification such as Hindi, Bengali etc [16]. Therefore we experimented with this strategy to see whether it improves the results for Malayalam too. According to the recent research, cross-lingual transformers have slight edge when using this transfer-learning strategy [16].

Data Preprocessing The data preprocessing for this task was kept fairly minimal to make it portable for other languages too. We only followed one data preprocessing technique; converting emojis to text. Emojis are found to play a key role in expressing emotions in the context of social media [33]. But, we cannot assure the existence of embeddings for emojis in pretrained models. Therefore as a preprocessing step, we converted emojis to text. For this conversion we used the Python libraries *demoji* [2] and *emoji* [3]. *demoji* returns a normal descriptive text and *emoji* returns a specifically formatted text. For an example, the conversion of ☺ is 'slightly smiling face' using *demoji* and ':slightly_smiling_face:' using *emoji*. Considering that *demoji* returns a normal text, we used *demoji* to convert the emojis to text.

Fine-tuning To improve the models, we experimented different fine-tuning strategies: majority class self-ensemble, average self-ensemble, language modelling, which are described below. These fine tuning strategies have shown promising results in recent shared tasks [34].

[2]demoji repository - https://github.com/bsolomon1124/demojis
[3]emoji repository - https://github.com/carpedm20/emoji

1. **Self-Ensemble (SE)** - Self-ensemble is found as a technique which results better performance than the performance of a single model [35]. In this approach, same model architecture is trained or fine-tuned with different random seeds or train-validation splits. Then the output of each model is aggregated to generate the final results. As the aggregation methods, we analysed majority-class and average in this research. The number of models used with self-ensemble will be denoted by N.
    - *Majority-class SE (MSE)* - As the majority class, we computed the mode of the classes predicted by each model. Given a data instance, following the softmax layer, a model predicts probabilities for each class and the class with highest probability is taken as the model predicted class.
    - *Average SE (ASE)* - In average SE, final probability of class c is calculated as the average of probabilities predicted by each model as in Equation 2 where h is the final hidden state of the [CLS] token. Then the class with highest probability is selected as the final class.

$$p_{ASE}(c|h) = \frac{\sum_{k=1}^{N} p_k(c|h)}{N} \tag{2}$$

2. **Language Modelling (LM)** - As language modelling, we retrained the transformer model on task dataset before fine-tuning it for the downstream task; text classification. This training is took place according with the model's initial trained objective. Following this technique model understanding on the task data can be improved.

Implementation We used a Nvidia Tesla K80 GPU to train the models. We mainly fine tuned the learning rate and number of epochs of the classification model manually to obtain the best results for the validation set. We obtained $1e^{-}5$ as the best value for learning rate and 3 as the best value for number of epochs for all the languages. Training for English language took around 1 hour while training for Malayalam took around 30 minutes.

4. Results and Evaluation

In this section, we report the experiments we conducted and their results. As informed by the task organisers, we used Weighted Average F1 score to measure the model performance. We also report Precision, Recall and F1 score for each class label as well the Macro F1 score in the results tables. Results in Tables 1 - 5 are computed on validation dataset. Finally, in Section 4.1 we report the results provided by organisers to our models, for the test set.

Table 1 shows the results we gained with traditional machine learning algorithms. Out of the three traditional machine learning algorithms Random Forest performed best, providing us with 0.93 weighted average F1 score. In the experiments we did with transformers, initially we focused on the impact of transfer learning when used with different transformer models and the obtained results are summarised in Table 2. According to the results XLM-R with transfer learning outperformed other models. Also we could notice that transfer learning improved both models; BERT and XLM-R.

	Non Hate Offensive			Hate Offensive			Weighted Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
Random Forest	0.93	0.99	0.96	0.92	0.68	0.78	0.93	0.93	**0.93**	0.87
Linear SVM	0.93	0.98	0.96	0.88	0.68	0.77	0.92	0.93	0.92	0.86
Mult. Naive Bayes	0.90	0.98	0.94	0.88	0.53	0.66	0.90	0.90	0.89	0.80

Table 1
Results for offensive language detection with traditional ML models. For each model, Precision (P), Recall (R), and F1 are reported on all classes, and weighted averages. Macro-F1 is also listed.

	Non Hate Offensive			Hate Offensive			Weighted Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
XLM-R (TL)	0.91	0.96	0.94	0.77	0.59	0.67	0.89	0.89	**0.89**	0.80
BERT-m (TL)	0.90	0.95	0.93	0.76	0.57	0.65	0.88	0.88	0.87	0.78
XLM-R	0.89	0.98	0.93	0.79	0.40	0.53	0.88	0.88	0.86	0.74
BERT-m	0.88	0.97	0.92	0.78	0.38	0.51	0.86	0.87	0.85	0.72

Table 2
Results for offensive language detection with default settings on Transformers. For each model, Precision (P), Recall (R), and F1 are reported on all classes, and weighted averages. Macro-F1 is also listed. *TL* indicated the *Transfer Learning* experiments

The self ensemble methods were experimented using all the transformer models and obtained results are summarised in Tables 3 and 4. In most experiments, ASE has given a higher F1 than MSE and it improved the results over the default settings. With that fine tuning strategy too XLM-R with transfer learning outperformed all the other models.

The language modeling fine tuning strategy were experimented using all the transformer models and obtained results are summarised in Table 5. These experimented were done on top of ASE fine tuning strategy since it provided better results than the default settings. Results show that language modeling clearly improved the results. In fact, the best result from our experiments were shown when XLM-R model with transfer learning fine tuned with ASE and language modeling.

	Non Hate Offensive			Hate Offensive			Weighted Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
XLM-R (TL)	0.91	0.96	0.94	0.77	0.59	0.67	0.89	0.89	**0.89**	0.80
BERT-m (TL)	0.90	0.95	0.93	0.76	0.57	0.65	0.88	0.88	0.87	0.78
XLM-R	0.89	0.98	0.93	0.79	0.42	0.55	0.88	0.88	0.86	0.76
BERT-m	0.88	0.97	0.92	0.78	0.40	0.53	0.87	0.87	0.85	0.74

Table 3
Results for offensive language detection with MSE on Transformers. For each model, Precision (P), Recall (R), and F1 are reported on all classes, and weighted averages. Macro-F1 is also listed. *TL* indicated the *Transfer Learning* experiments

	Non Hate Offensive			Hate Offensive			Weighted Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
XLM-R (TL)	0.92	0.97	0.95	0.78	0.60	0.68	0.90	0.90	**0.90**	0.81
BERT-m (TL)	0.91	0.96	0.94	0.77	0.58	0.66	0.89	0.89	0.88	0.79
XLM-R	0.90	0.99	0.94	0.80	0.43	0.56	0.89	0.89	0.87	0.77
BERT-m	0.89	0.98	0.93	0.79	0.41	0.54	0.88	0.88	0.86	0.75

Table 4
Results for offensive language detection with ASE on Transformers. For each model, Precision (P), Recall (R), and F1 are reported on all classes, and weighted averages. Macro-F1 is also listed. *TL* indicated the *Transfer Learning* experiments

	Non Hate Offensive			Hate Offensive			Weighted Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
XLM-R (TL)	0.94	0.99	0.97	0.82	0.63	0.69	0.93	0.93	**0.93**	0.85
BERT-m (TL)	0.92	0.97	0.95	0.78	0.59	0.67	0.91	0.91	0.91	0.82
XLM-R	0.91	0.99	0.95	0.81	0.44	0.56	0.90	0.90	0.88	0.78
BERT-m	0.890	0.99	0.94	0.80	0.42	0.55	0.89	0.89	0.87	0.76

Table 5
Results for offensive language detection with ASE and Language Modeling on Transformers. For each model, Precision (P), Recall (R), and F1 are reported on all classes, and weighted averages. Macro-F1 is also listed. *TL* indicated the *Transfer Learning* experiments

4.1. Submission Results

Considering the evaluation results on the validation set, we selected the fine-tuned XLM-R(TL) model with ASE + language modeling as our official submission to the HASOC task. According to the results provided by the organisers, our best model has scored 0.89 weighted average F1 score on the test set and ranked 5^{th} out of 12 participants.

5. Analysis

In addition to the experiments described in this paper, we carried out a qualitative analysis on the dataset to find interesting patterns and observations. In the training data out of 3,200 tweets only 567 were labelled offensive and the remaining 2,633 were labelled as not-offensive. The use of English words were minimal although there are many tweets which are in Malayalam language but written in Roman script. When analysing the tweets labelled as offensive, we observed that there are many tweets in the dataset which are actually not-offensive but labelled as offensive. Free English translations of some examples include:

(1) Spent 4 years proclaiming to be a Royal Mech.
(2) There are 25k dislikes from Ikka (Mammooty) fans, you are free to unlike and cry.
(3) Nice, looks like a TV drama series from SuryaTV (a Malayalam channel).
(4) Have you no shame defaming a reputed hospital?

We observed that between 20% and 25% of the tweets which are labelled as offensive are similar to the example shown above which has certainly impacted the performance of the models.

6. Conclusion

In this paper we have presented the system submitted by the WLV-RIT team to the HASOC 2020 - Offensive Language Identification - Dravidian Code Mix Task 1 at FIRE 2020. Following a recent study [16], we have shown that the XLM-R with transfer learning is the most successful transformer model from several transformer models we experimented. It should be noted that the traditional machine learning models comes very close to the performance of the transformer models. We have shown that the best traditional machine learning algorithm we experimented; Random Forest outperforms the majority of our transformer model based experiments. This can be due to properties of the dataset or due to the fact that a low-resource language like Malayalam is under represented in multilingual pre-trained models. With several fine tuning strategies, XLM-R with transfer learning provides the best result for the validation set. Finally, our approach achieved 5^{th} place in the leaderboard for the test set.

Acknowledgements

We would like to thank the HASOC organizers for running this interesting shared task and for replying promptly to all our inquiries. We further thank the anonymous reviewers for their insightful feedback.

References

[1] S. Bauman, R. B. Toomey, J. L. Walker, Associations among bullying, cyberbullying, and suicide in high school students, Journal of adolescence 36 (2013) 341–350.
[2] R. Kumar, A. K. Ojha, S. Malmasi, M. Zampieri, Evaluating aggression identification in social media, in: Proceedings of TRAC, 2020.
[3] Z. Pitenis, M. Zampieri, T. Ranasinghe, Offensive Language Identification in Greek, in: Proceedings of LREC, 2020.
[4] J. Risch, R. Krestel, Delete or not Delete? Semi-Automatic Comment Moderation for the Newsroom, in: Proceedings of TRAC, 2018.
[5] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, Y. Chang, Abusive language detection in online user content, in: Proceedings of WWW, 2016.
[6] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, N. Kourtellis, Large scale Crowdsourcing and Characterization of Twitter Abusive Behavior, in: Proceedings of ICWSM, 2018.
[7] R. Kumar, A. K. Ojha, S. Malmasi, M. Zampieri, Benchmarking Aggression Identification in Social Media, in: Proceedings of TRAC, 2018.
[8] C. Chelmis, D.-S. Zois, M. Yao, Mining patterns of cyberbullying on twitter, in: Proceedings of ICDMW, 2017.
[9] M. Yao, C. Chelmis, D.-S. Zois, Cyberbullying detection on instagram with optimal online feature selection, in: Proceedings of ASONAM, 2018.

[10] S. Malmasi, M. Zampieri, Detecting Hate Speech in Social Media, in: Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP), 2017, pp. 467–472.
[11] B. Mathew, A. Illendula, P. Saha, S. Sarkar, P. Goyal, A. Mukherjee, Temporal effects of unmoderated hate speech in gab, arXiv preprint arXiv:1909.10966 (2019).
[12] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. R. Pardo, P. Rosso, M. Sanguinetti, Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter, in: Proceedings of SemEval, 2019.
[13] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval), in: Proceedings of SemEval, 2019.
[14] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, Ç. Çöltekin, Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020), in: Proceedings of SemEval, 2020.
[15] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of NAACL, 2019.
[16] T. Ranasinghe, M. Zampieri, Multilingual offensive language identification with cross-lingual embeddings, in: Proceedings of EMNLP, 2020.
[17] B. R. Chakravarthi, M. A. Kumar, J. P. McCrae, P. B, S. KP, T. Mandl, Overview of the track on "HASOC-Offensive Language Identification- DravidianCodeMix", in: Proceedings of FIRE, 2020.
[18] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Predicting the Type and Target of Offensive Posts in Social Media, in: Proceedings of NAACL, 2019.
[19] T. Ranasinghe, H. Hettiarachchi, BRUMS at SemEval-2020 Task 12 : Transformer based Multilingual Offensive Language Identification in Social Media, in: Proceedings of SemEval, 2020.
[20] T. Ranasinghe, M. Zampieri, H. Hettiarachchi, BRUMS at HASOC 2019: Deep Learning Models for Multilingual Hate Speech and Offensive Language Identification, In Proceedings of FIRE (2019).
[21] H. Hettiarachchi, T. Ranasinghe, Brums at semeval-2020 task 3: Contextualised embeddings for predicting the (graded) effect of context in word similarity, in: Proceedings of SemEval, 2020.
[22] C. Liang, Y. Yu, H. Jiang, S. Er, R. Wang, T. Zhao, C. Zhang, Bond: Bert-assisted open-domain named entity recognition with distant supervision, in: Proceedings of SIGKDD, 2020.
[23] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, J. Lin, End-to-end open-domain question answering with BERTserini, in: Proceedings of NAACL, 2019.
[24] A. M. Kibriya, E. Frank, B. Pfahringer, G. Holmes, Multinomial naive bayes for text categorization revisited, in: Australasian Joint Conference on Artificial Intelligence, Springer, 2004, pp. 488–499.
[25] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.
[26] A. Liaw, M. Wiener, et al., Classification and regression by randomforest, R news 2 (2002) 18–22.
[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel,

P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.
[28] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, " O'Reilly Media, Inc.", 2009.
[29] C. Sun, X. Qiu, Y. Xu, X. Huang, How to fine-tune bert for text classification?, in: M. Sun, X. Huang, H. Ji, Z. Liu, Y. Liu (Eds.), Chinese Computational Linguistics, Springer International Publishing, 2019, pp. 194–206.
[30] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, arXiv preprint arXiv:1911.02116 (2019).
[31] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Brew, Huggingface's transformers: State-of-the-art natural language processing, ArXiv abs/1910.03771 (2019).
[32] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Predicting the type and target of offensive posts in social media, in: Proceedings of NAACL, 2019.
[33] H. Hettiarachchi, T. Ranasinghe, Emoji powered capsule network to detect type and target of offensive posts in social media, in: Proceedings of RANLP, 2019.
[34] H. Hettiarachchi, T. Ranasinghe, Infominer at wnut-2020 task 2: Transformer-based covid-19 informative tweet extraction, in: Proceedings of W-NUT, 2020.
[35] Y. Xu, X. Qiu, L. Zhou, X. Huang, Improving bert fine-tuning via self-ensemble and self-distillation, arXiv preprint arXiv:2002.10345 (2020).