KAFK at SemEval-2020 Task 12: Checkpoint Ensemble of Transformers for Hate Speech Classification

Kaushik Amar Das ◇, **Arup Baruah** ◇, **Ferdous Ahmed Barbhuiya** ◇, **Kuntal Dey** ♡†
◇IIIT Guwahati, India
♡Accenture Technology Labs, Bangalore
{kaushikamardas,arup.baruah}@gmail.com,
ferdous@iiitg.ac.in, kuntal.dey@accenture.com

Abstract

This paper presents the approach of Team KAFK for the English edition of SemEval-2020 Task 12. We use checkpoint ensembling to create ensembles of BERT-based transformers and show that it can improve the performance of classification systems. We explore attention mask dropout to mitigate for the poor constructs of social media texts. Our classifiers scored macro-f1 of 0.909, 0.551 and 0.616 for subtasks A, B and C respectively. The code is publicly released online.

1 Introduction

The research community has put much effort over the last few years in developing automated detection methods to combat hate speech in social media (Schmidt and Wiegand, 2017). But the complex nature of this phenomenon shows that it has no single solution. The second edition of the OffensEval Workshop (Zampieri et al., 2020), titled OffensEval-2020, is organized with the goal of promoting further research in this domain.

The OffensEval-2020 workshop features shared subtasks in five different languages. We participated in the English language which consists of three subtasks (A, B and C). Each subtask is a breakdown of the taxonomy of offensive content. *Subtask A* (Offensive language identification) is the classification of a post as offensive [OFF] or not offensive [NOT]. *Subtask B* (Automatic categorization of offence types) determines whether a post containing an insult or a threat is targeted towards an individual, a group, or other [TIN] or simply contains non-targeted profanity and swearing [UNT]. And *Subtask C* (Offense target identification) is the identification of the target of an offensive post. The targets being individual [IND], group [GRP] and other [OTH].

2 Related Work

Zampieri et al. (2019b) organized OffensEval-2019 where the participants were provided with a dataset of 14,200 annotated tweets (Zampieri et al., 2019a) to perform fine-grained classification of hate speech. HatEval (Basile et al., 2019) is another workshop on the detection of hateful content, specifically, hate speech against women and immigrants in Twitter posts. In this workshop the participants used various approaches such as SVM, sentence and word level embeddings, LSTMs (Hochreiter and Schmidhuber, 1997), Logistic Regression, etc., to build their classification systems, for example, (Baruah et al., 2019; Indurthi et al., 2019). HASOC 2019 (Mandl et al., 2019) was organized with a similar goal, but with a specific interest in Indo-European languages, namely Hindi, English and German. The top-performing systems in HASOC mostly made use of neural network models such as BERT (Devlin et al., 2019), Convolutional Neural Networks (CNN), LSTMs and multilingual embeddings. Other workshops like TRAC (Kumar et al., 2018; Kumar et al., 2020) focused on the identification of aggression in social media text. These workshops featured tasks that consisted of building classifiers that could discriminate between Overtly Aggressive, Covertly Aggressive, and Non-aggressive texts.

† This work was done when the author was affiliated with IBM Research India, New Delhi.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 2023–2029
Barcelona, Spain (Online), December 12, 2020.

3 Data

The dataset for the English edition of SemEval Task-12 is compiled by Rosenthal et al. (2020). It is a large scale dataset of loosely labelled text samples, i.e, each sample is associated with an *Average Confidence* measure and it's *Standard Deviation*. *Average Confidence* is the average of the confidences with which numerous supervised models predicted an instance as belonging to the positive class for a subtask. The positive class is OFF for subtask A, and UNT for subtask B. The dataset of subtask C has no positive class, instead, the average confidence of each class is given. In our study, we assign an instance to a class if its average confidence for that class is greater than or equal to 0.50. The label counts for each subtask is given in Figure 1.

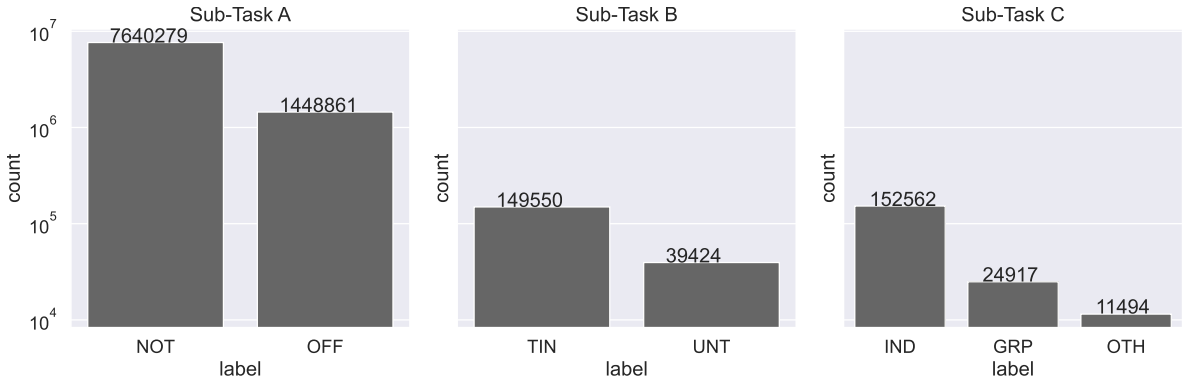


Figure 1: SemEval-2020 Dataset Sample Counts

4 Methodology

4.1 Checkpoint Ensembling

Model Ensembling refers to the method of constructing a single classifier from a collection of different classifiers (Dietterich, 2000). However, creating ensembles of large Deep Neural Networks (DNNs), like the ones used in this study, is an expensive task. It is generally not possible to train multiple models with limited time and GPU resources. Hence, based on the work by Chen et al. (2017), we used a simple *Checkpoint Ensembling* method for creating the transformer-based ensemble classifiers used in this study.

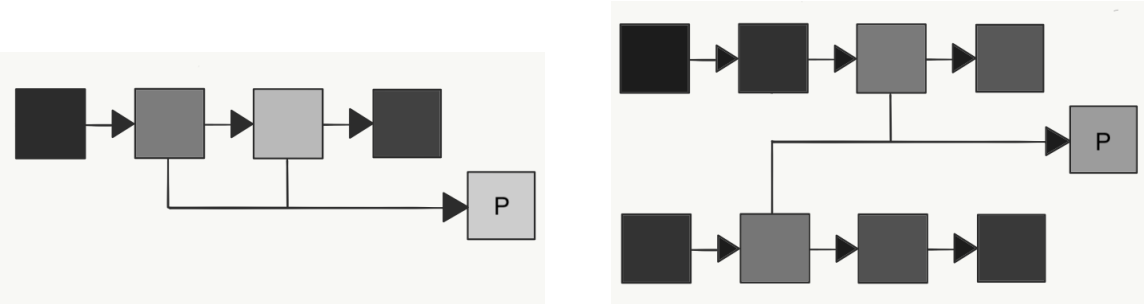
In *Checkpoint Ensembling*, a copy of the model is saved at each checkpoint. These copies are later combined in some fashion to make the classification (see Figure 2a). In contrast, a traditional ensemble usually combines different models (see Figure 2b). In our checkpoint ensembling approach, we save the dev set predictions and weights of the DNN models at each epoch and apply Algorithm 1 to determine which models to use for the ensemble. Algorithm 1 uses the dev set predictions to create a list of models to use in the ensemble. Algorithm 1 picks a model if using its predictions improves the metric (macro-f1 for OffensEval), otherwise not. Algorithm 1 is called twice with $reverse$ set to $True$ and then $False$. If the ensemble doesn't improve the metric, we can simply choose the best model found during training. After determining the models, we apply Algorithm 2 to get the final predictions. Algorithm 2 simply adds the predictions of the classifying layer of the chosen models and uses $argmax$ along each row to get the final prediction.

4.2 Classifiers

This section describes the classifiers built for each of the subtasks of OffensEval-2020. All of the classifiers described below follow the basic transfer learning procedure as shown in Figure 3. The classifiers and their training routines are written using PyTorch[1] (Paszke et al., 2019). The data splits are made such that the percentage of samples for each class is the same in each split. The random seed is set to 42 wherever applicable. The code has been made public for reference [2].

[1] `https://pytorch.org/`

[2] `https://github.com/cozek/OffensEval2020-code`



(a) Checkpoint Ensemble (b) Traditional Ensemble

Figure 2: Ensembles. The square boxes are models at each step of a training process. Lighter shade means better performance. P is the final prediction made by the ensemble.

Algorithm 1 Naive Checkpoint Ensemble

$A \leftarrow$ True labels, $reverse \leftarrow boolean$
$P \leftarrow$ Model predictions at each epoch
$N \leftarrow$ Number of samples, $C \leftarrow$ Number of classes
function ENSEMBLE($P, A, N, C, reverse$)
    $models \leftarrow \{\}, val \leftarrow 0$
    $Z[N][C] \leftarrow$ Zero Matrix
    $\epsilon \leftarrow len(P)$ ▷ Number of Epochs
    if $reverse$ **then**
        $range \leftarrow \epsilon$ to 0
    else
        $range \leftarrow 0$ to ϵ
    end if
    for $(e \leftarrow range)$ **do**
        $temp \leftarrow Z$
        $temp \leftarrow temp + P[e]$
        if $metric(A, temp) > val$ **then**
            $Z \leftarrow Z + P$
            $models \leftarrow models \cup e$
            $val \leftarrow metric(A, temp)$
        else
            continue
        end if
    end for
    return $models, val$
end function

Algorithm 2 Make Prediction

$m \leftarrow$ model ids chosen for ensemble
$E[N][C] \leftarrow$ Zero Matrix
for i in m **do**
    Load model with weights at epoch i
    $p \leftarrow model.predict(samples)$
    $E \leftarrow E + p$
end for
$preds \leftarrow$ Index of max element in each row of E

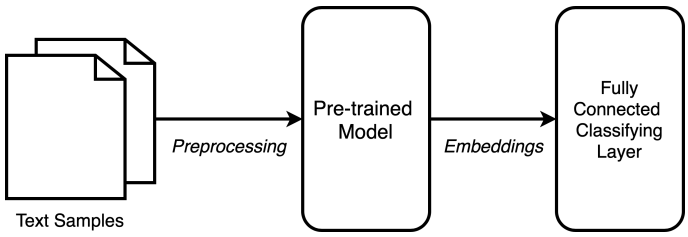


Figure 3: Transfer Learning Model

4.2.1 Subtask A: Ensembled GPT-2

GPT-2 is a large transformer-based (Vaswani et al., 2017) language-model developed by Radford et al. (2019). Its is trained on 40GB of internet text and has many capabilities, the main being able to generate synthetic text. The text generated of such high quality that it can easily be mistaken for being human-written.

Due to GPT-2's extremely large-size (1.5 billion parameters), it requires a lot of time and resources to train. So, we used DistilGPT-2 instead. Distil* (Sanh et al., 2019) is a class of compressed transformer-based models that has faster training and inference time while being small in size. These models are meant to enable the use of large high-performance models in a production environment. The authors of distil* show that it retains up to 97% performance of the original models. Also, Distil* enabled us to work on the massive datasets of OffensEval-2020 with high-performance base models with modest computing resources. This model is used for building the classifier for subtask A.

We used 70% of the data for training and 13633 samples for validation. We used a small subset for validation as we did not have enough time to obtain inferences on the entire remaining 30% ($\approx 2.7 \times 10^6$ samples). For training the classifier, we first converted each text sample into a sentence-matrix by extracting $786 - dimensional$ word embeddings from pre-trained DistilGPT-2. These were then fed into a dense layer having c units which makes the classification. Here c is the number of classes. We fine-tuned the entire model using a cross-entropy loss-function with a small learning rate of $1e-4$ for 5 epochs and applied *Checkpoint Ensembling* as described in Subsection 4.1. For optimizing the model, we used Ranger which is a combination of two optimizers, RAdam (Liu et al., 2019a) wrapped with LookAhead (Zhang et al., 2019). We set the (k, α) parameters of the optimizer to (5,0.5). The batch-size and maximum-sequence length were set to 399 and 64 respectively. Checkpoint ensembling improved the dev set macro-f1 score from 0.9656 to 0.9663.

4.2.2 Subtask B: Ensembled RoBERTa

Liu et al. (2019b) identified the short-comings of BERT (Devlin et al., 2019) and introduced RoBERTa, a robustly optimized version of BERT. We used it's pre-trained version for subtask B. We coupled *Checkpoint Ensembling* with the pre-trained *Roberta Sequence Classifier* by Wolf et al. (2019). The classifier was trained for 20 epochs with early stopping patience set to 4. We used 85% of the data as train set and the rest as dev set. The maximum sequence length was set to 245 and batch size used was 128. The other parameters and hyper-parameters were kept the same as that of subtask A. Checkpoint ensembling improved macro-f1 on the dev set from 0.8881 to 0.8907.

4.2.3 Subtask C: Ensembled DistilRoBERTa with Attention Mask Dropout

This classifier was built using the pre-trained distilled version of *Roberta Sequence Classifier* by Wolf et al. (2019), coupled with a slightly modified *Checkpoint Ensembling*. Instead of directly using the dev set predictions, we first sorted it in decreasing order of their macro-f1 scores. We also applied a drop out to the attention masks. Attention masks specify which of the tokens of the sentence the model should attend to. With probability p, we randomly dropped $d\%$ of attention masks of the tokens in the sentence. We set p to 0.50 and d to 30%. We used attention mask dropout in an attempt to mitigate the poor grammar which is often encountered in social media texts. Similar to the original dropout technique (Hinton et al., 2012), it was only used during training. The maximum sequence length was set to 245. The pre-trained model

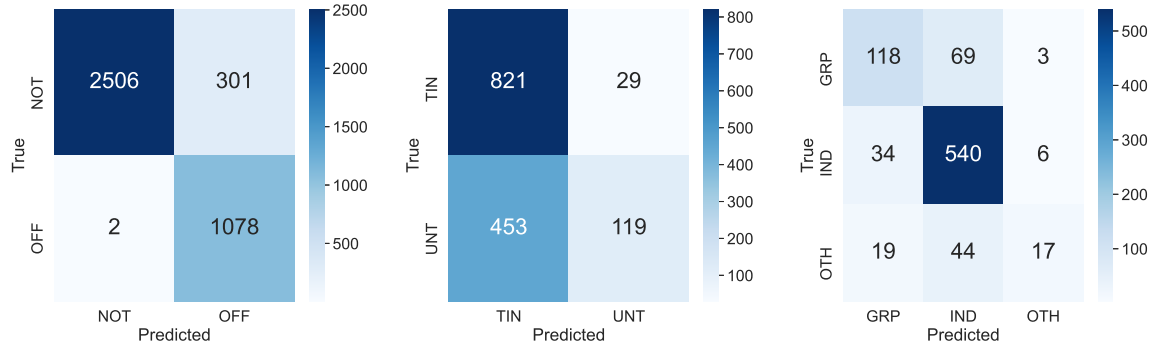
was fine-tuned using a cross-entropy loss function for 20 epochs with early stopping patience set to 4. We used a batch-size of 120 and learning rate of $1e-04$ with Ranger optimizer. Like in previous classifiers, we set the (k, α) parameters of the optimizer to (5, 0.5). Here 95% of the labelled data was used for training and the rest as dev set. Ensembling improved the dev set macro-f1 from 0.8148 to 0.8281.

5 Results and Error Analysis

	Model	*Macro F1*			*Rank*
		Dev Set	*Test Set*	*Best*	
Subtask A	Ensembled DistilGPT-2	0.9663	0.90989	0.92226	25
Subtask B	Ensembled RoBERTa	0.8907	0.55181	0.74618	31
Subtask C	Ensembled DistilRoBERTa	0.8281	0.6168	0.7145	15

Table 1: Results

The dev and test set results for each of the classifiers are given in Table 1. The test set contained 3887, 1422, 850 text samples for subtasks A, B and C respectively. Ensembled DistilGPT-2 was able to cross the 0.90 mark. Perhaps using the full dataset instead of just 70% might have resulted in a better score. This shows the need for techniques like model quantization to deal with massive datasets. Ensembled RoBERTa and Ensembled DistilRoBERTa performed quite poorly. They clearly overfit the data as the difference between the dev set and test set f1 scores is quite large. Later experimentation revealed that attention mask dropout in subtask C hurt the performance of the model. Without it, the dev set macro-f1 was 0.8275 and sorted checkpoint ensembling improved the score to 0.8336.



(a) Subtask A Test Set Confmat (b) Subtask B Test Set Confmat (c) Subtask C Test Set Confmat

Figure 4: Confusion Matrices

Ensembled DistilGPT-2 for Subtask A was able to correctly classify every offensive sample, except for two which were predicted as not offensive. Those two samples being • *"@USER @USER Dehumanize? He barely has a reflection of human"* and • *"@USER @USER 'Respect the result' is a mendacious soundbite trotted out by charlatans daily"*. This was perhaps due to the complex use of language, use of rare words such as *mendacious*, *trotted* and absence of typical profane words. As seen from the confusion matrix in Figure 4a, this classifier had a bias towards the offensive class.

Ensembled Roberta for Subtask B performed poorly. It failed to distinguish properly between targeted and untargeted samples. Untargeted samples such as • *"@USER Yeah my deck was insane that run, but normally I suck lol"* were predicted as targeted. We found that samples which contained the *"@USER"* token were mostly mistaken as targeted. The confusion matrix is given in Figure 4b.

Ensembled DistilRoberta for Subtask C misclassified most of the samples of 'other' class as targeting individuals, as apparent from its confusion matrix in Figure 4c. The same effect can be seen in the group class. Also, we found that this classifier made many mistakes where emojis are used. For example • *" I'm the only ☝ to get the job done ..... ion kno a nigga dat can cover for me"* is misclassified as belonging to the individual class rather than group targeted. In this example, the ' ☝ ' emoji is being used to denote "one". The model was unable to get the context from the emojis.

6 Discussion and Conclusion

In this work, we built three different transformer-based classifiers for the three subtasks of OffensEval 2020. We created ensembles using Checkpoint Ensembling. Although checkpoint ensembling improved the performance of the classifiers, the improvements were quite small. But, considering how cheap and simple the method is, it can't be dismissed completely. We found that attention mask dropout did not work as expected. We feel that more tuning of the p and d hyper-parameters might have been necessary to get it to work properly. In future work, we would like to explore and evaluate more sophisticated ensembling methods such as the Meta Classifier by Malmasi and Zampieri (2018).

References

Arup Baruah, Ferdous A. Barbhuiya, and Kuntal Dey. 2019. Abaruah at semeval-2019 task 5 : Bi-directional lstm for hate speech detection. In *SemEval@NAACL-HLT*.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA, June. Association for Computational Linguistics.

Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. Checkpoint ensembles: Ensemble methods from a single training process.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Vijayasaradhi Indurthi, Bakhtiyar Syed, Manish Shrivastava, Nikhil Chakravartula, Manish Gupta, and Vasudeva Varma. 2019. Fermi at semeval-2019 task 5: Using sentence embeddings to identify hate speech against immigrants and women in twitter. In *SemEval@NAACL-HLT*.

Ritesh Kumar, Atul Kr. Ojha, Marcos Zampieri, and Shervin Malmasi, editors. 2018. *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Ritesh Kumar, Atul Kr. Ojha, Bornini Lahiri, Marcos Zampieri, Shervin Malmasi, Vanessa Murdock, and Daniel Kadar, editors. 2020. *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, Marseille, France, May. European Language Resources Association (ELRA).

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019a. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:187 – 202.

Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, FIRE '19, page 14–17, New York, NY, USA. Association for Computing Machinery.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2020. A Large-Scale Semi-Supervised Dataset for Offensive Language Identification. In *arxiv*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *SocialNLP@EACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *SemEval@NAACL-HLT*.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*.

Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. 2019. Lookahead optimizer: k steps forward, 1 step back.