A Comparative Study of Using Pre-trained Language Models for Toxic Comment Classification

Zhixue Zhao
zhixue.zhao@sheffield.ac.uk
University of Sheffield
Sheffield, UK

Ziqi Zhang
ziqi.zhang@sheffield.ac.uk
University of Sheffield
Sheffield, UK

Frank Hopfgartner
f.hopfgartner@sheffield.ac.uk
University of Sheffield
Sheffield, UK

ABSTRACT

As user-generated contents thrive, so does the spread of toxic comment. Therefore, detecting toxic comment becomes an active research area, and it is often handled as a text classification task. As recent popular methods for text classification tasks, pre-trained language model-based methods are at the forefront of natural language processing, achieving state-of-the-art performance on various NLP tasks. However, there is a paucity in studies using such methods on toxic comment classification. In this work, we study how to best make use of pre-trained language model-based methods for toxic comment classification and the performances of different pre-trained language models on these tasks. Our results show that, Out of the three most popular language models, i.e. BERT, RoBERTa, and XLM, BERT and RoBERTa generally outperform XLM on toxic comment classification. We also prove that using a basic linear downstream structure outperforms complex ones such as CNN and BiLSTM. What is more, we find that further fine-tuning a pre-trained language model with light hyper-parameter settings brings improvements to the downstream toxic comment classification task, especially when the task has a relatively small dataset.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Social and professional topics** → *User characteristics.*

KEYWORDS

toxic comment, hate speech, neural networks, language model, fine-tuning, pre-training, BERT, RoBERTa, XLM

ACM Reference Format:
Zhixue Zhao, Ziqi Zhang, and Frank Hopfgartner. 2021. A Comparative Study of Using Pre-trained Language Models for Toxic Comment Classification. In *Companion Proceedings of the Web Conference 2021 (WWW '21 Companion), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3442442.3452313

1 INTRODUCTION

Detection of toxic comments online has been a fast-growing research area [31, 39]. Toxic comment classification (TCC) is usually treated as a text classification problem, mainly dealt with by machine learning methods. [1, 9, 30]. The recent trend for text classification tasks uses language models (LMs) that are pre-trained on large unlabeled corpora. The basic idea is to extract the pre-trained neural network layers from the language model (LM) and add new neural network layers on top of them to tailor for the downstream task [18]. In other words, features from the pre-trained LM are transferred to new neural network layers to build a model for the target classification task.

Within the area of TCC research, only a few studies have used pre-trained LMs to TCC, and little work has been done to explore how to best make use of such LMs for TCC. Moreover, previous studies have used different LMs and different TCC datasets and tasks, which make their results difficult to compare. Further, existing widely-used pre-trained LMs are trained on formal languages from books and news rather than colloquial online speech where toxic comment can often be found. Therefore, it is unclear to what extent these existing pre-trained LMs offer real value for TCC tasks.

In this work, we explore strategies of utilising pre-trained LMs with the aim to understand their impact on the downstream TCC tasks, and to compare different LMs. In short, we look into the impact of downstream neural-network architectures, continued pre-training of LMs with a further exploration of light-weight training parameters and the performance differences between current widely-used pre-trained LMs.

Our results first confirm that using a simple linear neural network as the downstream structure atop of pre-trained LMs for TCC tasks works better than using sophisticated structures such as CNN and Bi-LSTM, which contain more parameters. Secondly, continued pre-training of a LM is beneficial to the downstream TCC, especially when the dataset is relatively small. The benefits are also noticeable even with low hyper-parameter settings that make the overall model much more efficient compared to those previously reported. It also makes our method more accessible to users with limited computational resources. Thirdly, the performance of different pre-trained LMs varies. BERT and RoBERTa generally outperform XLM on a wide range of TCC tasks, while XLM benefits more from continued pre-training of LMs.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 explains our approach to use LMs on TCC tasks. Section 4 presents our experiments and results. Section 5 concludes the work and discusses future work.

2 RELATED WORK

This section begins with an overview of how TCC tasks are defined in the literature. It will then go on to review the research on pre-trained LM-based methods for text classification. This is because

WWW '21 Companion, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8313-4/21/04.
https://doi.org/10.1145/3442442.3452313

TCC is essentially a text classification task, and limited research has been done on using pre-trained LM-based methods on TCC tasks, while it is becoming popular for text classification tasks. The last part of this section discusses existing studies on TCC tasks using pre-trained LMs and existing research gaps in this area.

2.1 Definition of toxic comment classification tasks

Toxic comment generally refers to different types of unhealthy and negative user-generated content, which includes hate speech, abusive language, cyberbullying, etc [4, 5, 22].

Toxic comment classification is a text classification task. In earlier studies, the majority of research handle TCC as a binary classification problem that separates one particular type of toxic comment from all the others [13]. Later studies frame toxic comment classification as multi-class classification tasks where a comment will be assigned into one of the multiple types of toxic comment, or multi-label tasks where a comment could be assigned into none or at least one (could be multiple) type of toxic comment. For an example of multi-class tasks, [32] and [40] categorize comments into three groups, where each comment belongs to "none" or "racism" or "sexism". Fewer studies on toxic comment classification consider multi-label problems. In their research, [37] studies toxic comment classification in both multi-class and multi-label fashion. In their multi-label task, one comment can be assigned with zero to six labels including "obscene", "threat", and so on. Multi-class and multi-label classifications make the learning harder by nature, and different types of toxic comments could have some intersectionality, hence increasing the complexity of the problem. However, they are closer to the real-world scenarios [12, 26, 37].

2.2 Pre-trained LM-based methods for text classification

Traditionally, text classification has used statistical machine learning methods such as Support Vector Machines, Naïve Bayes and Decision Trees [35]. Since 2010 research has shifted towards deep neural networks (DNN)-based models such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Bi-directional Long Short-Term Memory network (Bi-LSTM) and hybrid neural networks which combine different DNN configurations [10, 35, 43]. Since the introduction of transformer-based structures, the use of pre-trained LMs in downstream text classification has become the mainstream [11]. The basic process is to add task-specific layers for the downstream task atop of the pre-trained LM and then train the new model where only the task-specific layers are trained from scratch [11, 23, 27]. Commonly-used pre-trained LMs include BERT, RoBERTa, XLM, etc. [11, 23, 27]. These models were pre-trained on extraordinarily large corpora, such as those containing over 3 billion words for BERT [11].

Typically, the output layers (the last/bottom few layers following hidden layers) in a pre-trained LM are replaced with task-specific layers and then the new model will be trained in a supervised fashion on the target text classification. Two strategies are widely used for improving pre-trained LMs performance on downstream text classification tasks: the design of downstream neural network structure and the continued, in-domain pre-training of LMs [3, 7, 18, 24].

Downstream network architecture A basic downstream network architecture for classification tasks is a linear transformation layer [11]. For example, [29] and [28] use a basic downstream architecture on BERT for sentiment classification and hate speech detection. [7] extend the basic downstream architecture by adding an extra LSTM layer with self-attention between the linear classifier layer and the pre-trained LM. [3] have added two layers of Bi-LSTM on top of BERT. All these different networks, e.g., LSTM with attention, CNN, are added on top of the pre-trained LM and are then fed into a classifier layer. There are many other studies using a method of similar fashion. We do not cover them in details and point interested readers to the work of [15, 36]. However, current studies on downstream neural network architectures are mainly based on BERT, and not yet applied to the more recent LMs such as RoBERTa and XLM.

Continued pre-training of language models Some research explores continued pre-training of the LM using a large in-domain corpus and then transferring the newly fine-tuned LM to the target classification tasks. As a result, training of a classifier model is initialized with weights from a further fine-tuned LM. This approach is also referred as "continued pre-training in domain" [18].

SciBERT [3] and BioBERT [24] are two examples of "continued pre-training in domain", which both fine-tuned BERT using large domain corpora and require significant computing resources and time [18]. To address this, [18] propose task-adaptive pre-training (TAPT) which use unlabeled training data from downstream target task to further fine-tune RoBERTa. Hence, the data size for further fine-tuning LMs in domains has been largely reduced. However, in their proposed TAPT, significant computing resources are still required due to the high settings of hyper-parameters, particularly due to the batch size and the epoch.

Although TAPT is a method of continued pre-training of LMs in domain, in the remainder of this paper, we use "continued pre-training/fine-tuning in domain" to refer to continued fine-tuning the LM using a large unlabeled in-domain corpus that is not the same as the unlabeled training data from the target task. TAPT on the other hand, will be used to refer to the more specific type of continued pre-training in domain using the unlabeled training data from the target task.

2.3 Pre-trained LM-based methods for tasks toxic comment classification

Despite increasing research on TCC and pre-trained LM-based text classification, few studies are addressing the intersection of both. As a sub-task of text classification, TCC follows a similar trend shift from traditional machine learning methods to DNN-based methods, and to pre-trained LM-based methods. A few studies on TCC explored pre-trained LMs and strategies on how to transfer them efficiently. [28] transfer BERT to different downstream architectures on two hate speech classification tasks. In the shared task on aggressive content detection, TRAC-2, a few papers utilise pre-trained LMs but with a focus on improving the model performance on one specific TCC task rather than generalising their methods to multiple TCC tasks or pre-trained LM models [20]. Furthermore, Most

of them use BERT directly without complex downstream neural-network or continued fine-tuning [2, 16, 25, 34]. To the best of our knowledge, only one paper applies a straightforward RoBERTa (a linear layer atop), and no research has yet been conducted on utilizing XLM on TCC tasks [2]. It is unclear how other pre-trained LMs, such as RoBERTa and XLM, perform and how generalisable they are on a wide range of TCC tasks. Additionally, a few studies utilize BERT on TCC tasks but only one of them studied different transferring strategies, e.g. downstream architectures [2, 16, 25, 28, 34].

In summary, there remains scope for us to further exploit all benefits of pre-trained LM-based methods for TCC tasks. First, how do different pre-trained LMs perform on different downstream TCC tasks, especially multi-class and multi-label classification tasks which are harder than binary tasks? Second, how do different downstream neural network structures impact on the performance on TCC? Third, can continued pre-training in domain improve the performance on downstream TCC tasks and if so, can we further improve the efficiency of the training process while maintaining the performance?

3 METHODOLOGY

We focus on pre-trained LM-based methods for TCC and investigate two strategies of transferring pre-trained LMs to TCC tasks. First, we analyse the impact of the complexity of the downstream network architectures; Second, we study the impact of continued pre-training LMs following the TAPT method but using much lower hyper-parameters to further reduce the computational requirements. We refer to this as 'TAPT-light' [18]. Third, we compare the performance difference between pre-trained LMs.

In short, each model takes a pre-trained LM and add a downstream network architecture on top of it and then all parameters are jointly trained on a given supervised TCC task. The differences between models are based on: the pre-trained LM used (Section 3.1), the downstream architecture (Section 3.2), the continued fine-tuning method of the pre-trained LM (whether fine-tuned or how many epochs for the fine-tuning, Section 3.3). Additionally, we evaluate our strategies using a comprehensive set of TCC datasets, including ten different classes and both multi-class and multi-label classification tasks. Details of these will be given in Section 4.1.

3.1 Pre-trained language models

To investigate the performance of different pre-trained LMs, we choose three of the most cited pre-trained LMs to study the generalizability of our strategies applied to TCC tasks. These are BERT, RoBERTa and XLM, representing the current state-of-the-art in a wide range of tasks [11, 23, 27]. Another reason we select RoBERTa and XLM is that, as mentioned in Section 2.3, very few studies have used them on TCC but they have been shown to gain good performance on other related classification tasks when compared to BERT.

3.2 Downstream network architectures

In terms of the downstream layers, we compare three architectures: linear classification layer, CNN + linear classification layer, Bi-LSTM + linear classification layer [11, 28]. We choose these for the following reasons. First, Bi-LSTM and CNN are frequently used in text classification tasks [6, 19]. Second, the linear classifier is the simplest structure and allows us to compare with other complex downstream architectures and their effects on transfer learning performance on TCC tasks.

Figure 1 illustrates the three BERT models with different downstream neural network architectures, as an example of a pre-trained model set. RoBERTa and XLM follow the same methods, each one is tested on three different downstream neural network architectures.

Linear classification layer: In the linear layer, the output of the first token (i.e., the output of [CLS]) from the final hidden state of the LM is used in the final classification. The output of a default dimension of given LM is fed to a linear layer and transformed to a dimension equal to the number of labels. A dropout of 0.1 is applied before the linear transformation [11].

CNN + Linear classification layer: The second architecture is built on the first one but with convolutional layers between the linear layer and the LM. The outputs of each layer of the LM (excluding the language modelling head) are fed to a 3-layers convolutional network. This CNN configuration is the same as that proposed by [19] which includes three filter window sizes of 3, 4 and 5, respectively. Each size has 100 feature maps. The outputs from each convolutional layers are passed to a max-pooling layer and then concatenated together before feeding to the final linear classification layer, which is the same as mentioned above.

Bi-LSTM + Linear classification layer: This architecture is similar to the CNN-based structure but the three CNN layers are replaced with a bi-directional LSTM layer [17]. In short, the outputs of each layer of the LM are fed to a Bi-LSTM neural network and the final two hidden states of the output are then concatenated before feeding them to the final linear classification layer.

3.3 Continued pre-training in domain: TAPT-light

In this part, we explore the performance of continued pre-training in domain using minimum computing resources on TCC tasks. We build our methods on the work of TAPT and reduce the hyper-parameter settings of TAPT to further minimise the computational resources required, which we refer to as TAPT-light [18]. To be more specific, the batch size is reduced to 16, compared to 2048 in the original TAPT. Second, we experiment with different epoch values respectively: 1, 5, 10, 20, 50, 100, compared to 100 in the original TAPT.

In summary, for each TAPT-light model, the training process is divided into two stages. In the first stage, as highlighted in red in Figure 2, the pre-trained LM is fine-tuned on unlabelled training corpus which is created from the downstream task (Section 4.1). Models are trained for different epochs, respectively, for accessing the impact of epochs later. In the second stage, as highlighted in blue in Figure 2, the output layer of the LM is replaced with a linear classification layer and then the new classification model is trained in a supervised fashion on a TCC task, where all parameters are trained jointly. Only the linear classification layers are trained from scratch.

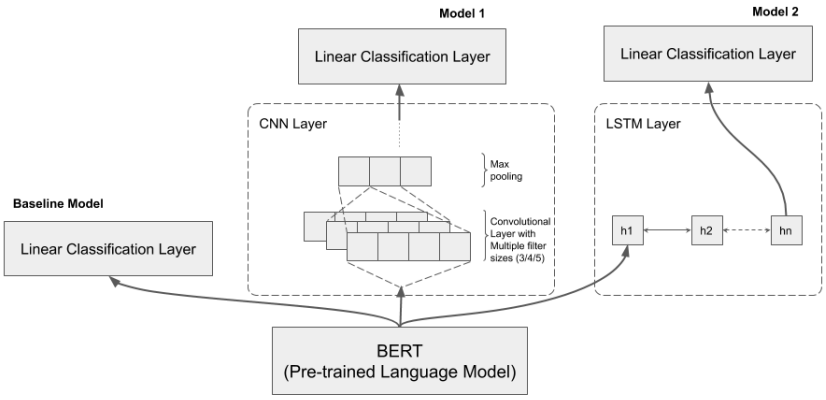

Figure 1: An example of utilizing BERT with different downstream architectures atop.

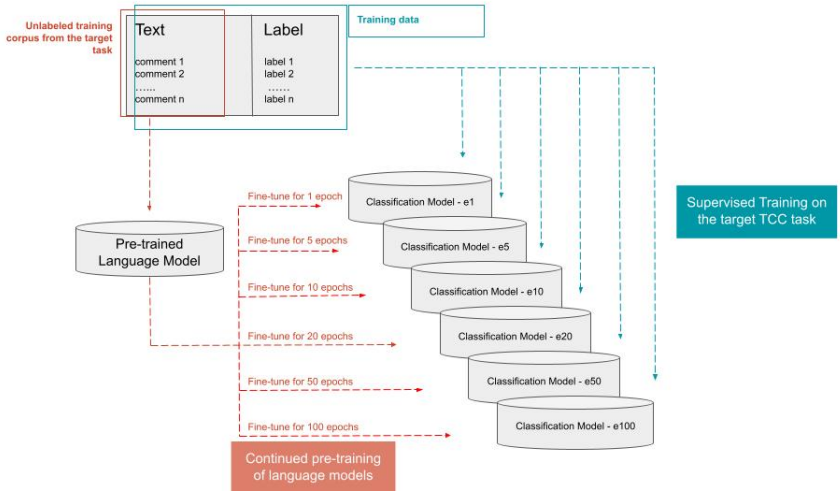

Figure 2: The training process of a TAPT-light model.

4 EXPERIMENTS

4.1 Datasets

Four TCC tasks are selected to assess the two strategies of transferring pre-trained LMs. We aim to cover different social media platforms, dataset sizes and classification types when selecting TCC tasks. The four selected datasets contain between range from 15,000 to 159,571 comments[1] and cover three different social media platforms with different classification types (multi-class and multi-label). Table 1 lists the details of the four datasets.

[1]The original version of Dataset Founta [14] included more than 80,000 tweets with tweets IDs published. We have successfully retrieved 50,425 valid tweets. The remaining missing tweets failed to be retrieved due to their deletion.

4.2 Baseline model

We create a baseline model for each pre-trained LM with the simplest components possible. Therefore, each baseline model takes a pre-trained LM without continued pre-training, and a linear classifier architecture is added on top of it, as shown by "Baseline Model" noted in Figure 1.

We select the version for each pre-trained LM that is of a parameter number ranging from 110 millions to 144 millions. This way, the three models are of a similar size. Plus, these sizes are smaller than most of their peer versions so that their training is faster. We use the following versions of BERT, RoBERTa and XLM: "bert-base-cased" (12 layers, 768 hidden dimensions, 12 heads, 110 million parameters); "roberta-base" (12 layers, 768 hidden dimensions, 12 heads, 125 million parameters); "xlm-mlm-enfr-1024" (6 layers, 1024 hidden dimensions, 8 heads, 144 million parameters).

Dataset	Data Source	Data Numbers	Labels	Classification Tasks
Kumar [21]	Facebook	14,998	non-aggressive (42%), overtly aggressive (35%), covertly aggressive (23%)	multi-class
Waseem[38]	Twitter	18,625	racism (11%), sexism (20%), both (69%), neither	multi-class
Founta [14]	Twitter	50,425	abusive (8%), hateful (3%), normal (73%), spam (16%)	multi-class
Wiki [8]	Wikipedia	159,571	toxic (10%), severe toxic (1%), obscene (5%), threat (0.3%), insult (5%), identity hate (1%)	multi-label

Table 1: Summary of the four toxic comment classification tasks.

4.3 Implementation

Pre-processing We constrain the maximum length of each input instance (i.e., a piece of comment) to be 100 tokens, following the standard practice found in literature [43]. In this way, we truncate the longer comment and pad the shorter messages with zero values.

Hyperparameters For all supervised learning in this work, i.e. the TCC tasks, we train the model for 3 epochs using a batch size of 16. For all unsupervised learning, i.e. the first stage of TAPT-light, we train the model for different epochs as stated in Section 3.3, using a batch size of 16. All training processes use Adam optimizer and a learning rate of 5e-5. These settings are recommended as "possible values to work well" in downstream task-specific training by [11].

Hardware and Implementation We used a single Tesla V100-PCIE-32GB GPU for all experiments. Compared with LM domain fine-tuning described in [3] (a single TPU v3 with 8 cores) and [18] (Google Cloud v3-8 TPU), our hardware requirements are less powerful. This makes the reproduction of our results easier.

Our implementation uses the HuggingFace transformers 3.0.0 library [41] and PyTorch 1.5.1. All multi-class classification tasks use Softmax as the final activation function and the multi-label classification tasks use Sigmoid instead.

4.4 Results: impact of downstream network architectures

Table 2 shows the results comparing the different downstream network architectures. "BERT", "RoBERTa" and "XLM" represent the baseline model (as shown by "Baseline Model" in Figure 1) for each pre-trained LM, while "BERT-CNN", as an example, represents the pre-trained BERT LM with a CNN structure on top as the downstream network architecture.

We first compare macro F1 of each model as it highlights how well a model handles minority classes on a unbalance dataset, compared to micro F1 [42]. Most of the TCC tasks typically handle highly unbalanced data, and as a result of that, model performance on minority classes (i.e. toxic comments) is often overshadowed by the majority classes when only looking at micro F1 [42]. As shown in Table 2, the macro F1 results indicate that the baseline models consistently perform much better than their "complex" counterparts where CNN or Bi-LSTM is used as the downstream network architecture.

In terms of micro F1, the baseline models obtained the highest F1 across all datasets, with the exception of XLM on the Founta dataset. The biggest difference compared to the CNN and Bi-LSTM downstream network architectures were noticed on the smaller datasets, i.e., Kumar and Waseem, and the multi-label classification task (Wiki). For example, the micro F1 of RoBERTa-CNN is reduced by 0.0465 and 0.0354, compared to the baseline model on the multi-label classification task (Wiki) and the smallest dataset (Kumar) respectively, and reduced by 0.0133 on Founta, whose dataset is multi-class and three times bigger than Kumar. Similar patterns are observed on results of BERT groups and XLM groups.

Comparing CNN and Bi-LSTM as the downstream structure for the pre-trained LM, there is no consistent pattern indicating which one is better.

The results above suggest that transferring a pre-trained LM to complex downstream network architectures does not offer any benefits over a simple linear structure in the context of TCC tasks. One possible reason for this is that the complex DNN might dilute or cannot interpret well the general representations that LMs learned from pre-training on extremely large corpora.

4.5 Results: impact of continued pre-training in domain

In summary, we have tested in-domain fine-tuning on four TCC tasks using three public pre-trained LMs for six different epoch settings. In total, 72 models have been reported in Table 3.

Firstly, as shown in table 2, continued pre-training on the target task often leads to further performance gain compared to the baseline models, with exceptions noted on the Founta dataset and Wiki dataset when using BERT and RoBERTa. Particularly on the smaller datasets, Kumar and Waseem, the performance improvements gained from TAPT-lights are noticeable. For example, the macro F1 on Waseem has increased by 0.1994, 0.2046 and 0.1427 for BERT, RoBERTa and XLM respectively, when further pre-training the LM for 50 epochs using the unlabeled downstream data. The higher increase of than micro F1 also suggests that continued pre-training particularly benefits small classes in the datasets, and these are often classes of the toxic comment rather than the non-toxic comment. While for Wiki, the biggest dataset among the four, the baseline models of BERT and RoBERTa achieved a strong performance and continued pre-training in domain demonstrates a detrimental impact on its classification performance. The only exception is XLM. The three continued pre-trained LMs give no noticeable advantages or disadvantages over the Founta task.

Another finding is that there is no consistent pattern in terms of which epoch setting is the best across the four tasks. However, for multi-class classification tasks (i.e. Kumar, Waseem and Founta), models fine-tuned for 10 or 20 epochs achieve the highest F1 scores or comparable results to their counterparts that use a higher epoch value. For the multi-label classification task (i.e. Wiki), as mentioned above, continued fine-tuning in domain does not show noticeable benefits on this task. The dataset of this task is at least three times bigger than the other three.

Model	**Kumar** (15k)		**Waseem** (18.6k)		**Founta** (50.4k)		**Wiki** (159.6k)	
	micro F1	**macro F1**	**micro F1**	**macro F1**	**micro F1**	**macro F1**	**micro F1**	**macro F1**
BERT	**0.5904**	**0.5805**	**0.8738**	**0.6196**	**0.7957**	**0.6071**	**0.7816**	**0.6372**
BERT-CNN	0.5796	0.5665	0.8480	0.5971	0.7888	0.5929	0.7388	0.4800
BERT-BiLSTM	0.5750	0.5611	0.8469	0.5921	0.7929	0.5959	0.7504	0.5945
RoBERTa	**0.5667**	**0.5441**	**0.8673**	**0.6159**	**0.8025**	**0.6315**	**0.7822**	**0.6510**
RoBERTa-CNN	0.5313	0.4819	0.8491	0.5902	0.7892	0.5367	0.7357	0.4658
RoBERTa-BiLSTM	0.5450	0.5013	0.8426	0.5828	0.7769	0.5563	0.7405	0.5431
XLM	**0.5646**	**0.5503**	**0.8464**	**0.5957**	0.7989	**0.6149**	**0.7594**	**0.5122**
XLM-CNN	0.5525	0.5345	0.8362	0.5860	**0.8023**	0.6015	0.7233	0.4642
XLM-BiLSTM	0.5600	0.5409	0.8346	0.5794	0.7987	0.5751	0.7519	0.4690

Table 2: Comparing F1 (micro and macro) obtained by transferring different language models (BERT, RoBERTa, XLM) to different downstream network architectures. The best performance for each pre-trained LM on each task is boldfaced. The dataset sizes are in the brackets after the datasets and "k" means a thousand or thousands.

Model	**Kumar** (15k)		**Waseem** (18.6k)		**Founta** (50.4k)		**Wiki** (159.6k)	
	micro F1	**macro F1**	**micro F1**	**macro F1**	**micro F1**	**macro F1**	**micro F1**	**macro F1**
BERT	0.5904	0.5805	0.8738	0.6196	0.7957	0.6071	**0.7816**	0.6372
BERT-e1	0.6050	0.5938	0.8737	0.6196	**0.7981**	0.6096	*0.7422*	*0.5462*
BERT-e5	0.6004	0.5899	0.8781	0.6198	0.7965	**0.6185**	*0.7475*	*0.5648*
BERT-e10	0.6029	0.5894	**0.8824**	0.7538	0.7957	0.6076	*0.7804*	**0.6539**
BERT-e20	0.6058	0.5926	0.8743	0.8176	0.7959	0.6079	*0.7404*	*0.6206*
BERT-e50	**0.6092**	**0.5972**	*0.8727*	**0.8190**	*0.7914*	*0.6025*	*0.7426*	*0.5581*
BERT-e100	0.6046	0.5898	0.8765	0.7479	*0.7906*	*0.6046*	*0.7395*	*0.5196*
RoBERTa	0.5667	0.5441	0.8673	0.6159	0.8025	0.6315	**0.7822**	**0.6510**
RoBERTa-e1	0.6075	0.5924	0.8690	0.6160	**0.8106**	0.6382	*0.7393*	*0.5067*
RoBERTa-e5	0.6075	0.5961	**0.8808**	0.6277	*0.8019*	**0.6391**	*0.7606*	*0.5650*
RoBERTa-e10	0.6146	**0.6051**	0.8711	0.6174	0.8060	0.6368	*0.7530*	*0.5606*
RoBERTa-e20	0.6063	0.5922	0.8711	*0.6134*	*0.8011*	*0.6275*	*0.7736*	*0.5875*
RoBERTa-e50	0.6146	0.6045	0.8759	**0.8205**	*0.7997*	*0.6163*	*0.7711*	*0.6503*
RoBERTa-e100	**0.6167**	0.6043	0.8743	0.7439	0.8027	*0.6242*	*0.7683*	*0.6209*
XLM	0.5646	0.5503	0.8464	0.5957	0.7989	0.6149	0.7594	0.5122
XLM-e1	0.5754	0.5566	0.8593	0.6063	**0.8044**	**0.6248**	*0.7548*	0.6247
XLM-e5	0.5883	0.5762	0.8593	0.6057	*0.7987*	0.6184	0.7600	0.5837
XLM-e10	0.6000	0.5896	0.8636	0.7364	0.8003	0.6170	0.7661	0.5952
XLM-e20	0.5988	0.5891	**0.8695**	**0.7826**	*0.7987*	0.6184	0.7712	0.6006
XLM-e50	0.6158	0.6091	0.8690	0.7384	0.7999	0.6178	0.7707	**0.6427**
XLM-e100	**0.6196**	**0.6097**	0.8727	0.7883	*0.7985*	0.6155	**0.7748**	0.6233

Table 3: Comparing F1 (micro and macro) of the baseline models and their continued domain fine-tuning language model counterparts. "BERT-e1" refers to fine-tuning the BERT language model for 1 epoch and then transfer it to the target TCC tasks. The best performance for each pre-trained LM on each task is boldfaced. The model whose result is worse than its baseline model is italicized.

To summarise, our results suggest that the continued fine-tuning in domain is beneficial to TCC tasks on relatively small datasets. This may be because a large dataset could have already supplied adequate information for the model to learn; thus, features from pre-trained LM offer less value to the learning. Additionally, for TCC tasks, more training epochs do not necessarily lead to performance benefits. Also, under these circumstances, a small batch size, e.g. 16, is workable. One possible explanation is that a large

number of epochs could have led to catastrophic forgetting that useful features gained from the massive LM pre-training could be erased during the later fine-tuning and thus lead to a decrease of model performance [33]. Although we did not directly compare against TAPT, we believe our findings are still encouraging for future researchers since a identical reproduction of TAPT is very resource demanding that is not accessible to most people in realistic situations. Our results suggest that even with very low hyperparameter settings, continued fine-tuning in-domain still contributes to TCC task especially when the training dataset of the TCC is relatively small and unbalanced.

4.6 Results: impact of different pre-trained language models

As shown in Table 2, BERT generally outperforms RoBERTa and XLM when an identical downstream structure is applied. This pattern is clearer when comparing their macro F1 and particularly on small datasets like Kumar. For example, BERT-CNN gains a macro F1 of 0.5665 on Kumar, noticeably higher than RoBERTa (0.4819) and XLM (0.5345).

With continued pre-training in domain (results in Table 3), RoBERTa generally outperforms BERT and XLM, especially when comparing their best TAPT-light models and baseline models on each TCC tasks with regard to macro F1. Moreover, TAPT-light benefits XLM more than RoBERTa and BERT, particularly on TCC tasks with with relatively small datasets. This can be observed as that all TAPT-light XLM models outperform its XLM baseline in terms of macro F1.

One possible reason behind the above patterns is that XLM handles long sentences or documents (4000 tokens composed of sentences) during its pre-training, while TCC usually tackles short sentences and texts [23]. TAPT-light uses in-domain data to fine-tune LMs, and therefore, could have helped XLM to 'learn' short texts. On the other hand, BERT and RoBERTa were pre-trained on corpora whose maximum sequence length is 512 tokens which is similar to many TCC datasets [11, 27]. This could have explained why they generally outperform XLM and that they benefit less from TAPT-light.

4.7 Limitations

Being limited to computational resources, this study lacks a set of experiments with identical parameter settings to TAPT which might give further information on how TAPT-light compares to the original TAPT on TCC tasks. Also, due to the unavailability of the original datasets[2] used in TAPT, we were unable to directly compare our TAPT-light against TAPT on identical datasets. Another unexplored factor is that each LM and sophisticated architectures such as CNN and Bi-LSTM could have its own favourable hyper-parameter settings. However, those hyper-parameter settings are unified in our study according to the common settings in previous studies but not further refined. Also, to what extent those hyper-parameter settings impact on the performance is unknown.

[2]Here we refer to the two datasets in the REVIEW domain which is a relatively similar domain to TCC.

5 CONCLUSION

This paper studied transferring pre-trained LMs to TCC tasks. We focused on the impact of different pre-trained LMs, downstream network structures, and using the dataset of the downstream task to continue fine-tuning the LM before transferring them to downstream TCC tasks, as well as the impact of lowering some of their training hyper-parameters.

Our results suggest that in the context of TCC tasks, BERT and RoBERTa generally perform better than XLM. Second, using a simple downstream network architecture is a better choice over a complex one, such as CNN and Bi-LSTM. Third, we also find that our approach TAPT-light shows advantages to downstream TCC tasks when the dataset size of the downstream tasks is relatively small. What is more, with TAPT-light, all other variables being equal, a low epoch setting will suffice to obtain the best achievable results of a model in most cases. This potentially makes the TAPT-based methods more accessible and easier to train.

Our future work will explore several directions. For example, we will explore these studies in a different aspect of downstream tasks to see if our findings generalise well on them; we will develop and test systematic methods of unfreezing and freezing certain layers' parameters in the pre-trained LM.

REFERENCES

[1] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 759–760.

[2] Arup Baruah, Kaushik Das, Ferdous Barbhuiya, and Kuntal Dey. 2020. Aggression identification in english, hindi and bangla text using bert, roberta and svm. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*. 76–82.

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).

[4] Pete Burnap, Omer F Rana, Nick Avis, Matthew Williams, William Housley, Adam Edwards, Jeffrey Morgan, and Luke Sloan. 2015. Detecting tension in online communities with computational Twitter analysis. *Technological Forecasting and Social Change* 95 (2015), 96–108.

[5] Vikas S Chavan and SS Shylaja. 2015. Machine learning approach for detection of cyber-aggressive comments by peers on social media network. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2354–2358.

[6] Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4 (2016), 357–370.

[7] Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained language models. *arXiv preprint arXiv:1902.10547* (2019).

[8] ConversationAI. 2017. *Toxic Comment Classification Challenge: Identify and classify toxic online comments.* https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[9] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*.

[10] Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*. 86–95.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[12] Yuchun Fang, Zhengyan Ma, Zhaoxiang Zhang, Xu-Yao Zhang, Xiang Bai, et al. 2017. Dynamic Multi-Task Learning with Convolutional Neural Network.. In *IJCAI*. 1668–1674.

[13] Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 85.

[14] Antigoni Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of

twitter abusive behavior. In *Twelfth International AAAI Conference on Web and Social Media*.

[15] Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. 2019. Target-dependent sentiment classification with BERT. *IEEE Access* 7 (2019), 154290–154299.

[16] Denis Gordeev and Olga Lykova. 2020. BERT of all trades, master of some. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*. 93–98.

[17] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks* 18, 5-6 (2005), 602–610.

[18] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. *arXiv preprint arXiv:2004.10964* (2020).

[19] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[20] Ritesh Kumar, Atul Kr Ojha, Bornini Lahiri, Marcos Zampieri, Shervin Malmasi, Vanessa Murdock, and Daniel Kadar. 2020. Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*.

[21] Ritesh Kumar, Aishwarya N. Reganti, Akshit Bhatia, and Tushar Maheshwari. 2018. Aggression-annotated Corpus of Hindi-English Code-mixed Data. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (Eds.). European Language Resources Association (ELRA), Miyazaki, Japan.

[22] Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *Twenty-seventh AAAI conference on artificial intelligence*.

[23] Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291* (2019).

[24] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.

[25] Han Liu, Peter Burnap, Wafa Alorainy, and Matthew Williams. 2020. Scmhl5 at trac-2 shared task on aggression identification: Bert based ensemble learning approach. (2020).

[26] Han Liu, Pete Burnap, Wafa Alorainy, and Matthew L Williams. 2019. Fuzzy Multi-task Learning for Hate Speech Type Identification. In *The World Wide Web Conference*. ACM, 3006–3012.

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[28] Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. 2019. A BERT-based transfer learning approach for hate speech detection in online social media. In *International Conference on Complex Networks and Their Applications*. Springer, 928–940.

[29] Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, Vol. 1. IEEE, 1–5.

[30] Vinita Nahar, Sanad Al-Maskari, Xue Li, and Chaoyi Pang. 2014. Semi-supervised learning for cyberbullying detection in social networks. In *Australasian Database Conference*. Springer, 160–171.

[31] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 145–153.

[32] Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206* (2017).

[33] Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. Dissertation. NATIONAL UNIVERSITY OF IRELAND, GALWAY.

[34] Niloofar Safi Samghabadi, Parth Patwa, PYKL Srinivas, Prerana Mukherjee, Amitava Das, and Thamar Solorio. 2020. Aggression and misogyny detection using BERT: A multi-task approach. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*. 126–131.

[35] Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. 1–10.

[36] Matthew Tang, Priyanka Gandhi, Md Ahsanul Kabir, Christopher Zou, Jordyn Blakey, and Xiao Luo. 2019. Progress notes classification and keyword extraction using attention-based deep learning models with BERT. *arXiv preprint arXiv:1910.05786* (2019).

[37] Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. *arXiv preprint arXiv:1809.07572* (2018).

[38] Zeerak Waseem. 2016. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*. Association for Computational Linguistics, Austin, Texas, 138–142. http://aclweb.org/anthology/W16-5618

[39] Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. *arXiv preprint arXiv:1705.09899* (2017).

[40] Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*. 88–93.

[41] 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 (2019).

[42] Ziqi Zhang and Lei Luo. 2019. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web* 10, 5 (2019), 925–945.

[43] Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference*. Springer, 745–760.