Information Engineering Express
International Institute of Applied Informatics
2026, Vol. 12, No. 1, IEE848

Offensive Language Detection on Social Media Using Three Language Models and Three Datasets

Zhenming Li [*], Kazutaka Shimada [*]

Abstract

There are more and more offensive posts on Social Media nowadays. Those posts are harmful and should be treated seriously. The most efficient way to detect offensive posts is to fine-tune a Large Language Model (LLM) on an offensive language dataset. In our research, we focus on maximizing the capacity of LLMs on offensive language detection tasks on Social Media. We select three LLMs with different attributes (DeepMoji, Bert, and HateBert) and three offensive language datasets (OLID, Curious Cat, and Ask FM). We mainly discuss achieving the best performance by configuring the LLMs and datasets. Experimental results show that simply fine-tuning an LLM with larger data can not always achieve the best performance. The combination of LLMs was effective, especially the combination of DeepMoji and HateBert.

Keywords: offensive language detection, Large Language Models, Social Media

1 Introduction

Nowadays, Social Media is getting more and more important in our daily lives. On Facebook or Instagram, people share their daily lives and interesting ideas with others. Social Media makes our lives more colorful and meaningful. However, some people intensively spread offensive comments toward others and society on Social Media. It has become a conspicuous problem. Hence, we need to pay attention to this phenomenon. One approach to detecting offensive Social Media content is to utilize NLP (Natural Language Processing) techniques. Large Language Models (LLMs) are important for this task because of their mighty capacity and enormous knowledge.

One approach to applying an LLM to downstream tasks such as offensive language detection is fine-tuning. However, the characteristics of each LLM are different, e.g., the size, structure, and so on. We select three LLMs with different structures and attributes to discuss this question: DeepMoji, Bert$_{Base}$, and Hate-Bert.

Another question is the size and characteristics of datasets for fine-tuning. In general, a larger dataset is preferred for the fine-tuning. However, the combination of datasets does not always generate better performance. We compare three datasets from different resources and the combinations to discuss this question: OLID, Curious Cat, and Ask FM. We compare

[*] Kyushu Institute of Technology, Fukuoka, Japan

how the dataset is applied to the LLMs as the combination approach. In this paper, we introduce "Knowledge Extension" to combine two datasets. In the knowledge extension, we compute a similarity measure between two datasets and then create a new dataset for fine-tuning.

The contributions of this paper are as follows:

- We compare three LLMs and three datasets from several aspects. As one knowledge from the experimental result, HateBert, which was pre-trained about the task, was the most effective among three LLMs. In other words, compatibility between LLM and the downstream task is extremely important.
- We show the effectiveness of introducing similarity of datasets in fine-tuning an LLM. Our knowledge extension approach with the similarity-based dataset often contributes to the improvement of accuracy.

In Section 2, we introduce related work based on LLMs to offensive language detection. Then, we we introduce the details of three basic LLMs and three datasets in Section 3 and 4. Next, we introduce the knowledge extension approach for the data combination in Section 5. We evaluate several combinations of LLMs and datasets and discuss the results in Section 6. Finally, we conclude our work in Section 7.

2 Related Work

With the tremendous traffic of incoming posts, manually filtering all the posts is already impossible. Utilizing NLP techniques is an efficient way, and it attracts huge attention from the researchers. Traditional machine learning methods have proved robust in offensive language detection. Aditya et al. [1] have evaluated logistic regression, naive bayes, and support vector machines using n-grams of TF-IDF as features on a dataset they generated. They reported that their base result was 92.6 on the F1- score by using N-gram:(1,2) + TFIDF on a naive bayes classifier. Fatemah [2] has proposed an ensemble machine learning method on Arabic offensive language detection. He conducted the ensemble approaches such as bagging, Adaboost, and random forest, and reported that bagging performed the best F1-score, 88.0%.

Deep neural network (DNN) has proved efficient as well. Park et al. [3] have utilized character-level and word-level CNNs to classify comments in the dataset by combining DATA-TWITTER-W and DATA-TWITTER-WH datasets. They reported that combining the two levels of granularity using two input channels achieves the best results, improving upon a character n-grams-based LR baseline (weighted F1 from 81.4% to 82.7%). Badjatiy et al. [4] evaluated several DNN architectures such as LSTM, CNN, CNN + GRU, and LSTM + Attention. They reported that the best setting was the word-level (LSTM) model that started with randomly initialized word embedding. The best result was the weighted f1-score of 93% for the DATA-TWITTER-WH dataset.

Nowadays with the rise of LLM, there have been more and more works trying to apply LLMs to offensive language detection. Konthala Yasaswini et al.[5] simply fine-tune several Transfromer-based LLMs such as mBERT, XLM-RoBERTa, DistilmBERT and ALBERT in their work. They consider datasets in Tamil, Malayalam, and Kannada languages. They compared the performance of Transformerbased LLMs with CNN and LSTM and found that Transformer-based LLMs performed better than CNN or LSTM.

Mozarari et al. [6] have proposed a Few-Shot Meta-Learning Framework. In their framework, they consider meta-learning in a multilingual setting. They selected six tasks concerning offensive language detection in six languages: English, Arabic, Danish, Turkish, Greek, and Persian. During the Meta-Training, one of the six languages is selected as "Target", and the others are "Support" so that the model can learn "Support" tasks and improve the "Target" task. They used XLM-R as the base learner and apply three meta-learning frameworks: Non-episodic, MAML, and Proto-MAML. They reported that meta-learning models perform better than transfer-learning-based models.

Roy et al. [7] have proposed an ensemble method considering several Language Models dealing with code-mixed offensive language. They applied ensemble learning on LLMs such as mBERT, NN-based models such as LSTM, and ML classifiers such as SVM and RF. They reported that transformer-based models like mBERT always perform better than ML and NN-based models. In their studies, means of utilizing LLMs are thoroughly discussed, however, suitability of LLM toward the certain downstream task is not discussed.

Casula and Tonelli [8] have proposed a generation-based data augmentation method for offensive language detection using several LLMs. They utilized some templates to generate auxiliary data from GPT-2 and test the augmented data on the BERT-base-uncased and RoBERTa-base. They reported the effectiveness of their proposed data augmentation method using GPT-2.

Our approach is also based on LLMs. We investigate the combinations of three LLMs with different attributes. In addition, we focus on the similarity of datasets to improve the accuracy in the offensive language identification task.

3 Large Language Model (LLM)

In our research, we utilize three pre-trained language models: DeepMoji, Bert$_{Base}$, and HateBert. We explain the size and structure of each model. In addition, we describe the relation with our task.

3.1 DeepMoji

The details of DeepMoji are as follow:

Structure and Size: DeepMoji was released by Bjarke Felbo et al [9]. It consists of an embedding layer, two BiLSTM layers, one token-level self-attension [10] layer, and one pooling layer. The structure of DeepMoji is shown in Figure 1. The total number of parameters in DeepMoji is 30 million.

DeepMoji was pre-trained from a corpus consisting of 1.2 billion tweets with a single emoji. It is pre-trained by predicting the emoji correctly in the original tweet.

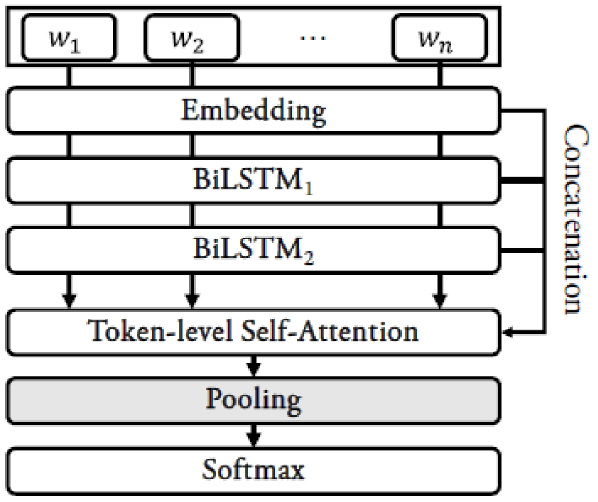

Figure 1: The structure of DeepMoji. In this paper, we use the pooling layer for the classification, as we will explain later.

Relation with our task: Since DeepMoji was trained to predict emojis, it is powerful in dealing with sentiment analysis tasks. Offensive language is highly related to the sentimental exhibition, such as anger, jealousy, and dislike. Consequently, although DeepMoji was not trained especially for offensive language detection, we can take advantage of its capacity to handle sentimental analysis.

3.2 Bert$_{Base}$

The details of Bert$_{Base}$ are as follows:

Structure and Size: Bert was released by Devlin et al [11]. There are many variants of the Bert model. In this paper, we use the bert-base-uncased (Bert$_{Base}$) model. Bert$_{Base}$ consists of an Embedding layer and 12 Transformer Encoders. The total number of parameters in Bert$_{Base}$ is 110 million.
Bert$_{Base}$ was pre-trained on Wikipedia and BookCorpus corpora. The training process is based on two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP).

Relation with our task: Bert$_{Base}$ is a robust language model on many benchmark tasks. Due to its MLM pre-taining, it is possible to generate contextual language representation for input sentences. Although Bert$_{Base}$ is not specially trained for offensive language detection, we still can use the language representation generated from Bert$_{Base}$ in our task.

3.3 HateBert

The details of HateBert are as follows:

Structure and Size: HateBert was released by Tommaso Caselli et al [12]. It is the same structure as Bert$_{Base}$. Therefore, the total number of parameters in HateBert is also 110 million.
HateBert is not pre-trained from scratch. It is just an extended version of the Bert$_{Base}$

model with a corpus. The authors collected offensive comments from Reddit Channels: the RAL-E corpus. Then, similar to MLM in Bert$_{Base}$, they fine-tuned the model with the corpus.

Relation with our task: As mentioned above, HateBert is specially trained for offensive language detection. Therefore, it can generate contextual language representation effectively for offensive language.

4 Datasets

In our research, we utilize three annotated datasets: OLID, Curious Cat, and Ask FM. We explain basic information and the statistics of the datasets.

4.1 OLID

The details of OLID are as follow:

Basic Information: OLID is the abbreviation for Offensive Language Identification Dataset [13]. The data were collected from Twitter, and all the tweets in the dataset are in English. Although there are three schemas for annotation of the dataset, we use one of them. The annotation is “Offensive” or “Non-Offensive” for each tweet. We remove most non-ASCII characters, except emoji, as a pre-processing. Hashtags, URLs, and Retweets in the tweet are converted to @hashtag, @URL, and @retweet, respectively. Informal spellings of words are regularized: tryyyyyyyy! -> try!

Statistics: OLID consists of 14100 annotated tweets and it is divided into a training partition of 13,240 tweets and a testing partition of 860 tweets by the authors. The training partition is divided into 80% for training and 20% for development. In the total of 14100 tweets, there are 4640 offensive tweets and 9460 Non-Offensive tweets. The followings are some examples in the dataset:

- @USER He is a moron [Offensive]
- @USER Well she is better looking than the lying chicks lawyer [Offensive]
- @USER Good luck on your future endeavors...you are surely going to be missed [Non-Offensive]

4.2 Curious Cat

The details of Curious Cat are as follow:

Basic Information: Curious Cat is an offensive language detection dataset [14]. The data were the posts collected from a website, Curious Cat, and annotated manually into binary labels: Offensive or Non-Offensive. The dataset contains posts written in English, and all the posts that are not in English were removed. In contrast with OLID, there is no pre-processing: removal of non-ASCII characters, regularization of informal spellings, and so on.

Statistics: Curious Cat consists of 4946 annotated posts and it is divided into three parts by authors: training, development, and test. In this paper, we follow the default setting of the partition of datasets. In the total 4946 posts, there are 780 offensive posts and 4184 Non-offensive posts. The followings are some examples in the dataset:

- are u from liverrrpoooooll [Non-Offensive]
- fuck u glee stans [Offensive]
- FAGGOT FAGGOT FAGGOT FAGGOT FAGGOT FAGGOT FAGGOT FAGGOT FAGGOT FAGGOT FAGGOT FAGGOT [Offensive]

4.3 Ask FM

The details of Ask FM are as follow:

Basic Information: Ask FM is an offensive language detection dataset [15]. The same as Curious Cat, Ask FM is collected from a website, Ask FM, and annotated manually into binary labels: Offensive or Non-Offensive. The dataset contains posts written in English, and all the posts that are not in English were removed. The same as Curious Cat, there is no pre-processing: removal of non-ASCII characters, regularization of informal spellings, and so on. In other words, the two datasets, Curious Cat and Ask FM, are extremely similar in formation and style.

Statistics: Ask FM consists of 11194 annotated posts and it is divided into three parts by authors: training, development, and test. We also follow the default setting of the partition of datasets. In the total of 11194 posts, there are 2023 offensive posts and 9171 Non-offensive posts. The followings are some examples in the dataset:

- for that ill get ava to kick you ass fucking meanie[U+200E] Mini Savage [Offensive]
- Kaitlin stfu and gtfo [Offensive]
- I wish I was as tall as you :((((([Non-Offensive]

5 Methodology

IIn this section, we describe how to fine-tune the model. The first approach is normal fine-tuning. For the fine-tuning, we compare two approaches: individual LLM and a combination of LLMs. The second approach is based on knowledge extension.

5.1 Fine-tuning

We prepare two types of fine-tuning approaches. In this fine-tuning context, the parameters of an LLM are updated based on the BinaryCrossEntropy value through normal back-propagation.

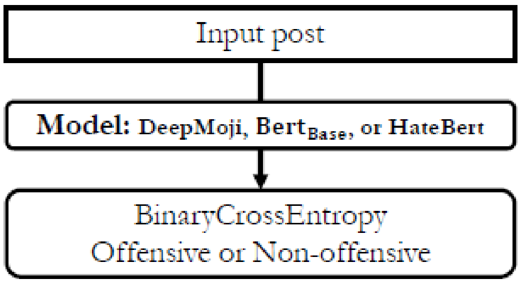
Input post
Model: DeepMoji, Bert$_{Base}$, or HateBert
BinaryCrossEntropy
Offensive or Non-offensive

Figure 2: Fine-tuning of each model.

Individual: To obtain a strong classifier, we need to apply a dataset to an LLM, namely fine-tuning. The simplest approach is one-model-to-one-dataset. Figure 2 shows this fine-tuning process. The model in the figure is assigned either DeepMoji, Bert$_{Base}$, or HateBert.

Combination: In general, for LLMs, the larger the number of parameters in LLM, the better the results tend to be generated. To consider the hypothesis, we combine LLMs in Section 3. Figure 3 shows the outline of the combination approach. Each model outputs the vector from an input post. For example, the CLS token is the vector for Bert$_{base}$ and HateBert. Table 1 shows the model name and the vector information. The combined model concatenates the output from each model, and it utilizes the dense layer for the classification task, namely offensive or non-offensive.

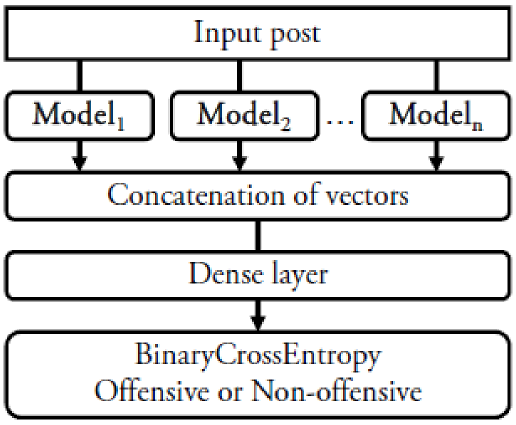

Figure 3: Combination of models: DeepMoji, Bert$_{Base}$, and HateBert.

Table 1: The vectors of each model for the concatenation process in the combined model.

Model	Vector	Size
DeepMoji	Pooling Layer	2304
BertBase	CLS Vector	768
HateBert	CLS Vector	768

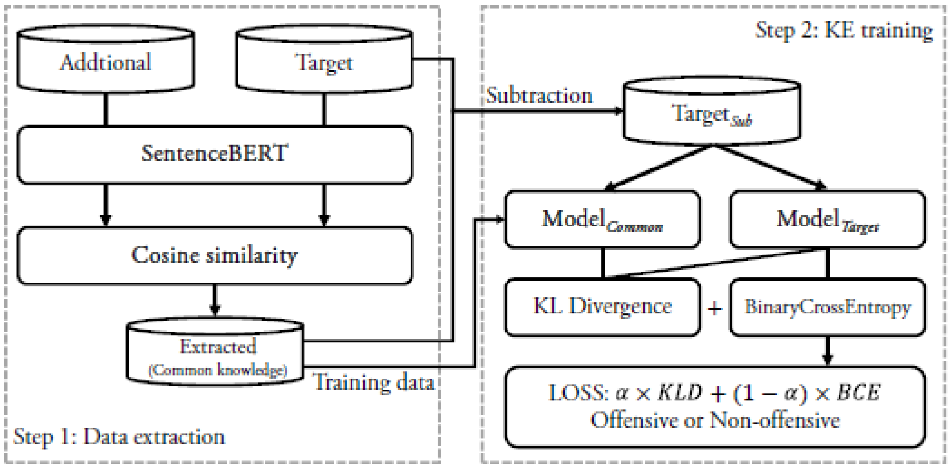

Figure 4: The outline of knowledge extension-based fine-tuning.

5.2 Knowledge Extension

In general, a larger dataset is preferred for the fine-tuning, as compared with a smaller one. However, the combination of datasets does not always generate better performance. Since the topic of the three datasets in this paper is offensive language identification, we assume that there are similar instances in the datasets, namely common knowledge. The similar instances contribute to the improvement of accuracy through the learning process.
In this paper, we propose a fine-tuning method based on knowledge extension.

The idea of knowledge extension comes from knowledge inheritance proposed by [16]. Knowledge inheritance is an idea designed to inherit knowledge from existing LLMs. It handles two LLMs; One LLM learns the knowledge from another LLM. As a result, the model becomes robust and mighty.

The outline of our approach, knowledge extension (KE), is shown in Figure 4. The method consists of two steps: data extraction and KE training. Step 1 is a process for extracting common knowledge from another (additional) dataset. In step 2, two models are learned for the KL divergence and the final loss by using the extracted data.

In the data extraction process, we select an additional dataset and determine the extraction rate, ***ER***, first. On the basis of ER, we decide the number of extracted instances from the two datasets by using the following equation.

$$NumExt = ER \times NumData \tag{1}$$

where *NumData* is the number of instances in the two datasets. Next, we generate embeddings of each instance by using SentenceBERT [17], one of the most famous sentence embedding models. Then, we compute the cosine similarity between all pairs in the two datasets. Finally, we extract the top-*NumExt* instances from the datasets. This process is carried out for both offensive instances and non-offensive instances.

Table 2: Hyper-Parameters.

hyper-parameters	status
Epochs	50
Batch Size	32
Max Sequence Length	100
Learning Rate	1e-5
EarlyStopping	15 epochs
ReduceLROnPlateau	5 epochs

In the KE training, we generate the dataset (Target$_{sub}$) by subtracting the extracted data from the original target dataset, namely the difference between them. Here we provide two models, following the knowledge inheritance. One is the model for learning common knowledge from the extracted data: Model$_{common}$. The other is the model for learning and predicting the label of each instance for the evaluation: Model$_{Target}$ We use the same LLM for the two models. In the learning process, we compute KL divergence between Model$_{common}$ and Model$_{Target}$. By using the KL divergence, Model$_{Target}$ can obtain the knowledge from Model$_{common}$. The final loss of this method is computed as

$$FinalLoss = \alpha \times KLD + (1 - \alpha) \times BCE \tag{2}$$

where $\boldsymbol{\alpha}$ is a weight parameter between two models. Finally, we predict each instance in the test data by using Model$_{target}$ fine-tuned from knowledge extension.

6 Experiment

6.1 Experiment Setting

The details of experiment settings for fine-tuning are as follow:

Hyper-Parameters: Hyper-parameters for the individual models and combined models are the same. The hyper-parameters are shown in Table 2. We introduce the Early-Stopping system to stop the training when there is no update on the development data anymore. We also use an automatic learning-rate altering system: ReduceLROnPlateau, to make the fine-tuning as comprehensive as possible.

Model combination: There are four choices for combined models: (DeepMoji + Bert$_{Base}$), (DeepMoji + HateBert), (Bert$_{Base}$ + HateBert), (DeepMoji + Bert$_{Base}$ + HateBert).

Metrics: For the offensive language identification task, it is important to determine whether offensive posts are correctly recognized by the model. On the other hand, datasets described in Section 4 were imbalanced; the number of non-offensive posts was larger than that of offensive posts. Therefore, we compute only the F1-score of offensive posts. It is the harmonic mean of the precision and recall.

Knowledge Extension: The hyper-parameters and metrics are the same as the setting above. The combinations of datasets, namely target and additional data, are six choices the data. Note that OLID as target and Curious Cat as additional are different with Curious Cat as target and OLID as additional. We set 0.3 to α in Equation (2), a parameter of Kullback Leibler Divergence and BinaryCrossEntropy.

6.2 Results and Analysis

In this section, we discuss the results from several aspects. There are three tables for the experiment results. Table 3 and Table 4 are results on normal Fine-tuning setting, while Table 5 shows the results of our proposed Knowledge Extension method. All these three tables show different details about the experiment, and we discuss the discoveries in our paper based on the three tables.

Discussion for Table 3: The results of individual and combined models are shown in Table 3. In the table, the first row denotes the name of the dataset, and the first column shows the name of LLM. The "AVE" denotes the average of three values of OLID, Curious Cat, and Ask FM. The "ALL" denotes the result from an LLM or LLMs fine-tuned from all three datasets. In other words, we combined three datasets as one dataset, and the model learned and evaluated the model with the dataset.

From the results about individual models, HateBert always outperformed DeepMoji and single Bert$_{Base}$. For example, for OLID, HateBert: 70.64 vs. Deepmoji: 65.92, Bert$_{Base}$: 67.63. The F1-scores of Bert$_{Base}$ for each dataset were lower than HateBert, although the structures of them are the same. As mentioned in Section 3, HateBert was designed especially for offensive language detection. Therefore, the pre-trained model for the downstream task obtained better performance through the datasets.

Then we observed that DeepMoji outperformed Bert$_{Base}$ for Curious Cat and AskFM. The reason is also the compatibility between DeepMoji and the two datasets. While DeepMoji was pre-trained on data from social media, Bert$_{Base}$ was pre-trained on data from Wikipedia and so on. The target data in this paper are data on social media. The models learned from similar resources are more effective as the basic models for fine-tuning.

From the results about combined models, we found that the combinations with HateBert contributed to the improvement of the F1-scores. For example, for OLID, B + H: 72.36 vs. Bert$_{Base}$: 67.63, D + H: 69.75 vs. DeepMoji: 65.92, D + B + H: 70.24 vs. D + B: 65.46. The best score for OLID was B+H (Bert$_{Base}$ and HateBert), the best scores for Curious Cat and Ask Fm were all combinations. This result shows the effectiveness of combinations of LLMs with different attributes: LSTM and emoji for DeepMoji, Transformers with written text corpora for Bert$_{Base}$, Pre-trained from offensive data for HateBert. Diversity of LLMs is one of the most important points for the classification model. However, we also observe cases with more models in combination but lower performance. For example, for OLID, B+H: 72.36 > D+B+H: 70.236. When we observe each element in the combination, we notice that the performance of each element has an influence on the combination. The single DeepMoji achieves 65.92 on OLID. It's the worst single model compared with Bert$_{Base}$: 67.63 and HateBert: 70.64. Therefore, when including DeepMoji into the combination, there is the possibility of deteriorating the total performance.

Table 3: Results of individual and combined models. The scores in boldface denotes the best score of each dataset. AVE denotes the average of columns of OLID, Curious Cat, and Ask FM. ALL is the F1-scores when three datasets were combined into one dataset. D, B, H denote DeepMoji, Bert$_{Base}$, and HateBert, respectively.

	OLID	Curious Cat	Ask FM	AVE	ALL
DeepMoji	65.92	63.87	58.97	62.92	61.50
Bert$_{Base}$	67.63	48.98	54.35	56.99	53.78
HateBert	70.64	70.11	59.80	66.85	56.58
D + B	65.46	48.87	53.02	55.78	52.13
D + H	69.75	71.62	60.37	67.25	55.48
B + H	72.36	66.79	59.04	66.06	54.74
D + B + H	70.24	72.62	61.06	67.97	56.70

Table 4: Evaluation with Ask FM. The values in boldface were the best score in each row. "Ask FM only" denotes that the model was learned from data in Ask FM. "with O" denotes that the model learned from data in Ask FM and OILD, and the model was evaluated with Ask FM.

	Ask FM only	with O	with C	with O + C
DeepMoji	58.97	59.00	61.26	58.45
BertBase	54.35	48.28	53.92	48.11
HateBert	59.80	50.40	61.49	48.50
D + B	53.02	48.14	54.10	45.14
D + H	60.37	46.34	62.46	47.69
B + H	59.04	48.08	64.23	47.20
D + B + H	61.06	48.16	61.11	48.03

Table 5: Results of knowledge extension. The values in boldface denote the best score of the column.

Target	OLID		Curious Cat		Ask FM	
Additional	Curious Cat	Ask FM	OLID	Ask FM	OLID	Curious Cat
Extraction Rate: 0.3						
DeepMoji	63.69	64.67	52.77	57.11	54.29	56.88
BertBase	58.49	62.37	54.66†	53.18†	46.51	48.56
HateBert	67.59	68.56	57.91	70.43	52.63	60.12†
Extraction Rate: 0.7						
DeepMoji	53.18	55.05	60.45	49.54	57.37	51.02
BertBase	60.54	63.16	52.21†	53.76†	43.85	52.69
HateBert	70.69†	69.58	53.58	72.45	45.47	61.88†

Next, we compare AVE and ALL. The situation of ALL denotes the largest dataset of them because it contains OLID, Cusrious Cat, and AskFM. In general, larger training data lead to the improvement of accuracy. On the other hand, in this experiment, all F1-scores of ALL were lower than those of AVE: e.g., 62.92 vs. 61.50 for DeepMoji.

Discussion for Table 4: To answer the quesion about why ALL can not yield better results than AVE in Table 3, we analyse Table 4. Table 4 shows detailed relations about data combinations and the F1-socres. It's the normal finetuning setting as introduced in

Section 6.1, but training with different dataset combinations that contain AskFM and testing with AskFM. The models with Curious Cat tended to improve the F1-scores. On the other hand, the models with OLID (including OLID + Curious Cat) led to the deterioration of the F1-scores. The reason is similarity between datasets. While Curious Cat and Ask Fm are Q&A style social media services, data from OILD were collected from Twitter. This result shows that there are suitable combinations of datasets.

Discussion for Table 5: Next, we discuss the knowledge extraction approach. The experimental result is shown in Table 5. In the table, the first row denotes the target dataset, the second row shows the additional dataset for extraction of common knowledge. We compared two types of ER in Equation (1). The values with † in the table denote better F1-scores as compared with the same setting without knowledge extension (see Table 3). For example, 70.24 for HateBert and OLID in Table 3 vs. 70.69 in Table 5. For Ask FM, HateBert with knowledge extension outperformed the best F1-score, D+B+H: 61.88 vs. 61.06. In addition, empirically speaking, fine-tuning an LLM with large size data is costly in terms of time and memory resources. Consequently, our knowledge extension approach is a more effective and resource-efficient solution than normal fine-tuning with large data.

7 Conclusion

In this paper, we compared three LLMs and three datasets. We investigated suitable combinations of the three models and the characteristics. The best single model was HateBert, which was pre-trained on the similar task. This result shows the importance of the attribute of LLMs for the target task. We also introduced a knowledge extension approach. It was based on similarity-based data extraction. For Ask FM, HateBert with knowledge extension outperformed the best F1-score from all combinations As an interesting knowledge from the experimental result, the size of the dataset is not always the essential factor since it deteriorated the performance (See AVE and ALL in Table 3). One important thing is to use a similar dataset for the target dataset for generating a strong classifier.

However, there are still some issues in our work. Currently, we evaluate our approach with English datasets. We are interested in discussing the extendability of the our approach and result in this paper with other languages, such as Japanese, German, and Italian. Moreover, deep error analysis is needed for the improvement of our approach.

Acknowledgement

This work was partly supported by JSPS KAKENHI Grant Number 20K12110. This work was also partly supported by JST, the establishment of university fellowships towards the creation of science technology innovation, Grant Number JPMJFS2133.

References

[1] Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based

approach. arXiv preprint arXiv:1809.08651, 2018.

[2] Fatemah Husain. Arabic offensive language detection using machine learning and ensemble machine learning approaches. arXiv preprint arXiv:2005.08946, 2020.

[3] Ji Ho Park and Pascale Fung. One-step and two-step classification for abusive language detection on Twitter. In Proceedings of the First Workshop on Abusive Language Online, pages 41–45, 2017.

[4] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In Proceedings of the 26th international conference on World Wide Web companion, pages 759–760, 2017.

[5] Konthala Yasaswini, Karthik Puranik, Adeep Hande, Ruba Priyadharshini, Sajeetha Thavareesan, and Bharathi Raja Chakravarthi. IIITT@DravidianLangTech-EACL2021: Transfer Learning for Offensive Language Detection in Dravidian Languages. In Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pages 187–194, 2021.

[6] Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. Cross-Lingual Few-Shot Hate Speech and Offensive Language Detection Using Meta Learning. IEEE Access, 10:14880–14896, 2022.

[7] Pradeep Kumar Roy, Snehaan Bhawal, and Chinnaudayar Navaneethakrishnan Subalalitha. Hate speech and offensive language detection in Dravidian languages using deep ensemble framework. Computer Speech & Language, 75:101386, 2022.

[8] Camilla Casula and Sara Tonelli. Generation-Based Data Augmentation for Offensive Language Detection: Is It Worth It? In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 3359–3377, 2023.

[9] Bjarke Felbo, Alan Mislove, Anders Sogaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1615–1625, 2017.

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

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, 2019.

[12] Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. Hate-BERT: Retraining BERT for Abusive Language Detection in English. In Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021), pages 17–25, 2021.

[13] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in

Social Media (OffensEval). In Proceedings of the 13th International Workshop on Semantic Evaluation, pages 75–86, 2019.

[14] Niloofar Safi Samghabadi, Afsheen Hatami, Mahsa Shafaei, Sudipta Kar, and Thamar Solorio. Attending the Emotions to Detect Online Abusive Language. In Proceedings of the Fourth Workshop on Online Abuse and Harms, pages 79–88. Association for Computational Linguistics, 2020.

[15] Niloofar Safi Samghabadi, Suraj Maharjan, Alan Sprague, Raquel Diaz-Sprague, and Thamar Solorio. Detecting Nastiness in Social Media. In Proceedings of the First Workshop on Abusive Language Online, pages 63–72, 2017.

[16] Yujia Qin, Yankai Lin, Jing Yi, Jiajie Zhang, Xu Han, Zhengyan Zhang, Yusheng Su, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Knowledge Inheritance for Pre-trained Language Models. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3921–3937, 2022.

[17] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084, 2019.