NTU_NLP at SemEval-2020 Task 12: Identifying Offensive Tweets Using Hierarchical Multi-Task Learning Approach

Po-Chun Chen[1], Hen-Hsen Huang[2], Hsin-Hsi Chen[1]
[1]Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
[2]Department of Computer Science,
National Chengchi University, Taipei, Taiwan
pcchen@nlg.csie.ntu.edu.tw, hhhuang@nccu.edu.tw,
hhchen@ntu.edu.tw

Abstract

This paper presents our hierarchical multi-task learning (HMTL) and multi-task learning (MTL) approaches for improving the text encoder in Sub-tasks A, B, and C of Multilingual Offensive Language Identification in Social Media (SemEval-2020 Task 12). We show that using the MTL approach can greatly improve the performance of complex problems, i.e. Sub-tasks B and C. Coupled with a hierarchical approach, the performances are further improved. Overall, our best model, HMTL outperforms the baseline model by 3% and 2% of Macro F-score in Sub-tasks B and C of OffensEval 2020, respectively.

1 Introduction

Multilingual Offensive Language Identification in Social Media (OffensEval 2020) hosted by (Zampieri et al., 2020) is a popular competition attracting many teams. The task is based on Offensive Language Identification Dataset (OLID) 1.0 and a new dataset SOLID. OffensEval 2020 consists of three subtasks. In Sub-task A, the goal is to identify if a given tweet is offensive or non-offensive. In Sub-task B, the focus is to identify if the offensive content in a tweet is targeted or untargeted. In Sub-task C, systems have to detect the type of a target in an offensive tweet.

The purpose of this paper is to investigate the multi-task learning (MTL) approach to the OffensEval tasks. We propose the MTL model, which is based on the popular text encoder BERT (Devlin et al., 2018) with the MTL architecture. To leverage the hierarchical nature of the three subtasks in OffensEval 2020, we further propose the HMTL model, which is based on the BERT with a hierarchical MTL architecture.

This paper is organized as follows. We summarize the related work in Section 2. Section 3 introduces the dataset, the preprocessing procedure, and the architectures of our models. Our results in OffensEval 2019 and OffensEval 2020 are reported and discussed in Section 4. Finally, Section 5 concludes this paper.

2 Related Work

The overview paper of OffensEval 2019 (Zampieri et al., 2019b) shows that 70% of the teams that participated last year use the deep learning models, and 8% of them use the BERT model in Sub-task A. Many of the top-ranked teams use the BERT model. The NULI team (Liu et al., 2019), which wins the first place in Sub-task A, proposes a BERT-based model and an LSTM-based model. Their work shows that the pre-trained BERT model has a good competition result in the task. The vradivchev_anikolov team (Radivchev and Nikolov, 2019) wins the first place and the second place in Sub-tasks C and A, respectively. They explore several models, among which BERT has the highest performance. The UM-

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 2105–2110
Barcelona, Spain (Online), December 12, 2020.

IU@LING team (Zhu et al., 2019) wins the third place in Sub-task A. They use the PyTorch framework to build the BERT model in Sub-task A, and use the SVM model in Sub-tasks B and C.

Multi-task learning (MTL) is one of widely-used strategies for improving neural network models. Ruder (2017) introduces the two most commonly used methods of MTL in deep learning, namely hard parameter sharing and soft parameter sharing approaches. Sanh et al. (2019) propose a hierarchical multi-task (HMTL) model, by combining low-level simple tasks with high-level complex tasks to train. The model achieves the state-of-the-art results in named entity recognition, entity mention detection, and relation extraction. In this paper, we consider the hierarchical nature of the three tasks and introduces the HMTL framework to deal with the problems.

3 Data and Methodology

3.1 Data

The OLID 1.0 dataset used in the competition last year is available as a part of the OffensEval 2019 Shared Task Zampieri et al. (2019a). It is collected from Twitter API by searching for some keywords. According to the hierarchical annotation, the main tasks of this competition are divided into three sub-tasks of different levels shown as follows:

Sub-task A: Offensive language identification.

Sub-task B: Automatic categorization of offense.

Sub-task C: Offense target identification.

Sub-task A is a binary classification task, where an instance will be labeled as Not Offensive (NOT) or Offensive (OFF). For an OFF instance, Sub-task B further categorizes it into two types: Targeted Insult (TIN) and Untargeted (UNT). Sub-task C focuses on the subtypes of TIN, where an TIN instance will be further categorized into three subtypes: Individual (IND), Group (GRP), and Other (OTH). The dataset contains 14,100 tweets, 13,240 tweets in the training set, and 860 tweets in the test set. Figure 1 shows the detailed information of the dataset.

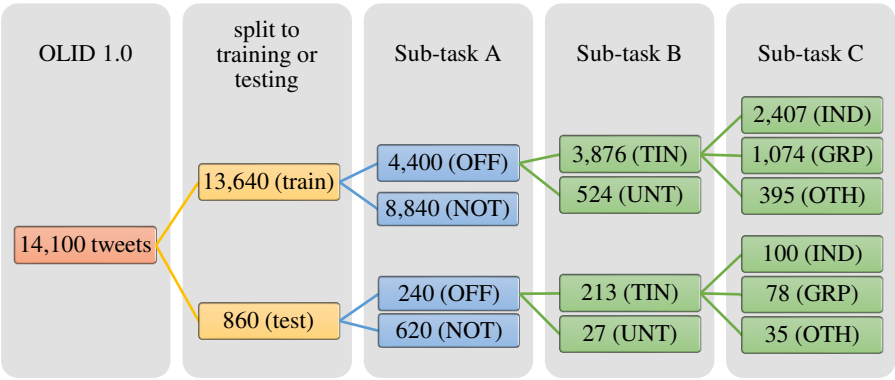

Figure 1: The hierarchy of the subtasks and the size of each part of dataset.

In the competition of this year, Rosenthal et al. (2020) provide a new dataset called SOLID. The new dataset employs the same schema as the original dataset but adopts a different annotation method. It is composed of weakly supervised data only. The dataset provides μ_{conf} and σ_{conf} for each instance, where μ_{conf} is the average of the confidences predicted by several supervised models, and σ_{conf} is the standard deviation of confidences computed from μ_{conf}. Then, μ_{conf} is the confidence scores of OFF in Subtask A and UNT in Sub-task B. In Sub-task C, each category has its own μ_{conf}. In Sub-task A, we treat the instances with a μ_{conf} greater than 50% as OFF, and the rest of instances as NOT. In Sub-task B, we treat the instances with a μ_{conf} greater than 50% as UNT, and the rest of instances as TIN. In Sub-task C, we select the category with the highest μ_{conf} as the category of the instance.

From Figure 1, we can indicate two issues with OLID 1.0 data. Firstly, it has the class imbalance problem, which will cause the classifier overfitting the more common classes, resulting in relatively low performances for less common classes (Minority Class Examples). Secondly, the number of certain categories is too small. For example, only 524 instances of UNT and 395 instances of OTH are available.

We solve these two problems by screening SOLID data. We extract the instances with high μ_{conf} and class-balanced data from SOLID to address these two issues.

For a hierarchical dataset such as OLID, we propose a method to ensure the balance of each subtask category and make the low-level subtasks have as much training data as possible. We propose the data selection method as follows.

Data Selection Method: We find the category with the least data in the lowest-level task and select all the data in that category. Supposing the quantity of these data is n, we select n instances from each of other categories in the same subtask to make each class balanced. Ideally, the amount of data in the upper layer is the sum of the number of his subtasks. According to this method, we can conclude that the number of SOLID required for each category is shown in Figure 2. Because each instance in SOLID has a different confidence, we sort instances according to μ_{conf} from high to low to obtain the required amount.

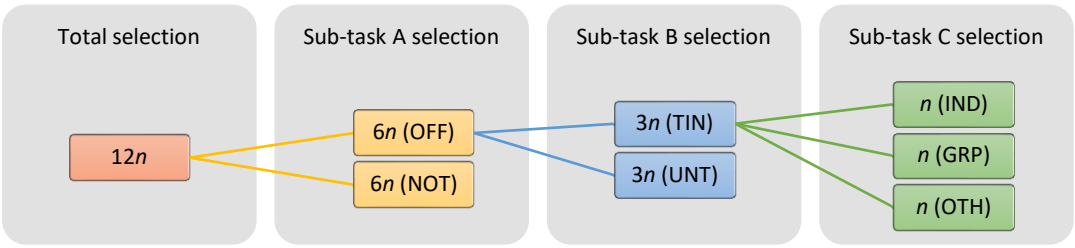

Figure 2: The amount of data required for each category in our data selection method.

Sub-task C:

The number of OTH category is the least, so we choose them all. The selection threshold of IND and GRP categories are the μ_{conf} greater than 0.79 and 0.52, respectively. Finally, there are 11,836, 12060, and 11,494 instances in the IND, GRP, and OTH categories, respectively.

Sub-task B:

We select 35,390 data in the sub-task C stage. According to the μ_{conf} of Sub-task B, 4,234 and 31,156 data belong to the UNT and TIN categories, respectively. Since UNT is far less than TIN, we choose more data with the μ_{conf} of UNT greater than 0.53 to balance the categories. It is worth mentioning that although all IND, GRP, and OTH should belong to the TIN category, the tweets of Sub-task B and Sub-task C are almost the same. Therefore, tweets likely to be the UNT category will also have IND, GRP, and OTH scores. Finally, there are 31,156 instances in the TIN category and 31,023 instances in the UNT category.

Sub-task A:

In sub-tasks B and C of the SOLID dataset, all the μ_{conf} of OFF of tweets is greater than 0.5. There are 62,179 instances in Sub-task B that belong to the OFF category. We choose the μ_{conf} of OFF less than 0.121 as NOT data to balance the categories. Finally, there are 62,179 instances in the OFF category and 61,115 instances in the NOT category.

Finally:

The selected data is added to the OLID dataset as our training instances. The number of each category is as Table 1.

Label	# of instances
OFF	66,579
NOT	69,955
Total	136,534

Sub-task A

Label	# of instances
TIN	35,032
UNT	31,547
Total	66,579

Sub-task B

Label	# of instances
IND	14,243
GRP	13,134
OTH	11,889
Total	39,266

Sub-task C

Table 1: the size of each part of our training dataset

3.2 Preprocessing

First, we convert emoji to English by an emoji project.[1] In the tweets, emojis may have some offensive semantics. Taking an OLID instance as an example: "*@USER Because you are* 💩". Second, we merge all continuously and repeated "@user" to a single "@user" symbol to reduce the redundancy, since the training set has too many consecutive and repeated "@user". Take one of the OLID data as an example: "*@USER @USER @USER @USER @USER @USER @USER @USER @USER @USER @USER Do you know what's going to happen now? I ' m going to have to lay in bed and cry while I listen to Kelly Clarkson. Thanks free speech antifa.*" Finally, we use BERT's tokenizer to tokenize tweets.

3.3 Models

We train three models, including simple BERT model, hierarchical multi-task learning (HMTL) model, and multi-task learning (MTL) model. All models have the same parameter setting and BERT version. For training, a batch size 32, max-sequence-length 64, max-epoch-number 15, and Adam Optimizer learning rate 2e-7 are used. The training was carried out on an NVIDIA V100 GPU.

BERT: We train a BERT model for each different task. For all models, we use BERT provided by google-research, where the version is BERT-Large-Cased (Whole Word Masking). This BERT version has 24-layer, 1024-hidden, 16-heads, and 340M parameters.

Multi-task learning approach (MTL): Sub-task B and Sub-task C have much less data than Sub-task A. We plan to use the multi-task learning method to share Sub-task A's knowledge with other sub-tasks to improve the prediction performance. We choose a hard parameter sharing approach to construct the MTL model. Entire BERT hidden layers are shared among all tasks. The sum of the loss of each Sub-task is the loss of MTL, the formula of loss function as follows.

$$L_{MTL} = \sum_{i \in \{Sub-task\ A,\ \ Sub-task\ B,\ \ Sub-task\ C\}} L_i \qquad (1)$$

Hierarchical multi-task learning approach (HMTL): We design an architecture called HMTL that extends from MTL. Hierarchical models trained on the dataset with the hierarchical scheme may improve performance, since subcategories, TIN and UNT of Sub-task B, may be able to learn hidden information from the parent category, OFF of Sub-task A. Similarly, subcategories, IND, GRP and OTH of Sub-task B, may be able to learn hidden information from the parent category, TIN of Sub-task B.

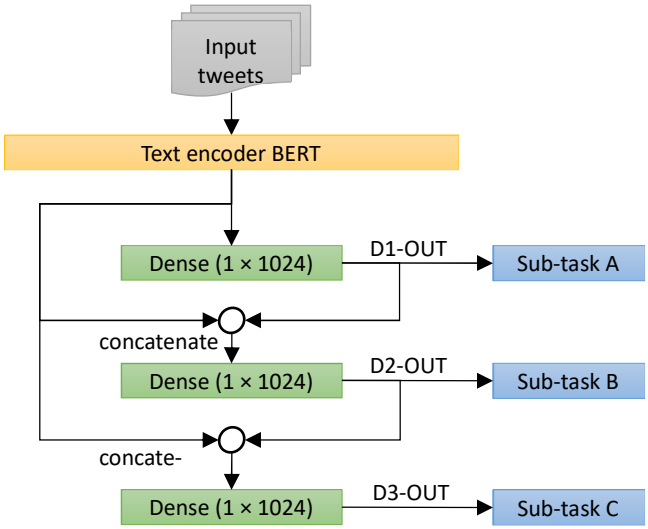

Figure 3: Architecture diagram of hierarchical multi-task approach

[1] https://github.com/carpedm20/emoji

We add three dense layers and modify the architecture as follows. The input of the first dense layer is the output of BERT, and its output (D1-OUT) is directly connected to the output layer of Sub-task A. The input of the second dense layer is the output of BERT concatenated with D1-OUT, and its output (D2-OUT) is directly connected to the output layer of Sub-task B. The input of the third dense layer is the output of BERT concatenated with D2-OUT, and its output is directly connected to the output layer of Sub-task C. We use the same loss function in the HMTL model as the MTL model. The architecture diagram of the model is shown in Figure 3.

4 Results

We show the results of our model in the OffensEval 2019 and 2020 testsets in Table 2 and Table 3, respectively. Both tables list the macro F-scores of the three models for each subtask. All the models we propose are much better than the majority baseline.

Table 2 lists the results on the test set of OffensEval 2019. The HMTL model has the best performance in Sub-task B, achieving a macro F-score of 0.7417. The MTL model has the best performance in Sub-tasks A and C, achieving a macro F-score of 0.8030 and 0.6223, respectively. Although the performance of the HMTL model is a little worse than that of the MTL model in Sub-tasks A and C, the difference is very small. Both HMTL and MTL are superior to BERT in every subtask, especially in Sub-task B and C.

In Table 3, we list the results on the test set of OffensEval 2020. The MTL model has the best performance in Sub-tasks A, achieving a macro F-score of 0.9120. The HMTL model has the best performance in Sub-tasks B and C, achieving a macro F-score of 0.7024 and 0.6577, respectively. Similarly, both HMTL and MTL are superior to BERT in every subtask, especially in Sub-task B and C.

Comparing the results of OffensEval 2019 and OffensEval 2020, our observations are as follows. Firstly, the performances of HMTL and MTL models in Sub-task A are similar. This may be because Sub-task A is relatively simple, the hierarchical approaches are not helpful. Secondly, Both HMTL and MTL outperform the vanilla BERT model in all sub-tasks of OffensEval 2019 and 2020, showing that the multi-task learning approach works very well in OffensEval. Thirdly, the performances of HMTL in Sub-tasks B and C is almost higher than those of MTL model except that it is a little worse in OffensEval 2019 Sub-task C. This shows that the hierarchical method is useful in more complex problems, such as Sub-tasks B and C. Lastly, in terms of the average F-scores of each model in each subtask, HMTL is better than MTL, and MTL is better than BERT.

Model	**Sub-task A**	**Sub-task B**	**Sub-task C**	**Avg.F-score**
Majority Baseline	0.4200	0.4700	0.2100	0.3667
HMTL	0.8019	**0.7417**	0.6189	**0.7208**
MTL	**0.8030**	0.7333	**0.6223**	0.7195
BERT	0.7878	0.7080	0.5950	0.6969

Table 2: Macro F-score of the proposed models in OffensEval 2019

Model	**Sub-task A**	**Sub-task B**	**Sub-task C**	**Avg.F-score**
Majority Baseline	0.4193	0.3741	0.5695	0.4543
HMTL	0.9115	**0.7024**	**0.6577**	**0.7572**
MTL	**0.9120**	0.7010	0.6550	0.7560
BERT	0.9098	0.6703	0.6355	0.7385

Table 3: Macro F-score of the proposed models in OffensEval 2020.

At the time of the competition, we used a relatively simple way to select training data. It consists of all the instances of OLID 1.0 and instances of Sub-task A in SOLID where their σ_{conf} s are less than 0.05.

Our submission result ranked 51st out of 85 participating teams in Sub-task A, achieving a macro F-score of 0.9068, ranked 3rd out of 43 participating teams in Sub-task B, achieving a macro F-score of 0.6906, and ranked 26th out of 39 participating teams in Sub-task C, achieving a macro F-score of 0.5695.

Table 3 shows that our proposed data selection method is very effective in improving performance. Compared with other teams, the rankings of tasks A, B, and C have been raised to 16th, 3rd, and 5th rank, respectively.

5 Conclusion

Our multi-task learning approaches, MTL and HMTL, can effectively use the implicit information of Sub-task A to improve the performance of Sub-task B and Sub-task C in the OffensEval 2019 and 2020. For both OffensEval evaluations, both MTL and HMTL models outperform the vanilla BERT model in all subtasks. We show that the multi-task learning approach works very well in more complex problems. Coupled with the hierarchical approach, performance can be further improved. Our MTL and HMTL approaches can be applied to other classification tasks with a hierarchical nature.

In the future, we will consider adding more complex structures after the BERT layer of the HMTL model. We hope that low-level tasks can gain more knowledge from high-level tasks to improve the overall performance.

Acknowledgements

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-106-2923-E-002-012-MY3, MOST 108-2218-E-009-051, MOST 108-2634-F-002-017, and MOST 109-2634-F-002-034, and by Academia Sinica, Taiwan, under grant AS-TP-107-M05.

Reference

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Ping Liu, Wen Li and Liang Zou. 2019. NULI at SemEval-2019 Task 6: Transfer Learning for Offensive Language Detection using Bidirectional Transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (*SemEval-2019*), pages 87–91, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Victor Radivchev and Alex Nikolov. 2019. Nikolov-Radivchev at SemEval-2019 Task 6: Offensive Tweet Classification with BERT and Ensembles. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (*SemEval-2019*), pages 691–695, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2020. A Large-Scale Weakly Supervised Dataset for Offensive Language Identification. *arXiv:2004.14454*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv:1706.05098*.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In Proceedings of *the Thirty-Third AAAI Conference on Artificial Intelligence* (*AAAI-19*), pages 6949–6956, Honolulu, Hawaii. AAAI Press.

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-HLT 2019*, pages 1415–1420, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

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 *Proceedings of the 13th International Workshop on Semantic Evaluation* (*SemEval 2019*), pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

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 the 14th International Workshop on Semantic Evaluation* (*SemEval 2020*), Barcelona, Spain.

Jian Zhu, Zuoyu Tian, and Sandra Kübler. 2019. UMIU@ LING at SemEval-2019 Task 6: Identifying offensive tweets using BERT and SVMs. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (*SemEval 2019*), pages 788–795, Minneapolis, Minnesota, USA. Association for Computational Linguistics.