MedAI at SemEval-2021 Task 5: Start-to-end Tagging Framework for Toxic Spans Detection

Zhen Wang[1], Hongjie Fan[2]*, Junfei Liu[3]
School of Software and Microelectronics, Peking University
The Department of Science and Technology, China University of Political Science and Law
National Engineering Research Center for Software Engineering, Peking University
{wang.zh, liujunfei}@pku.edu.cn , hjfan@cupl.edu.cn

Abstract

This paper describes the system submitted to SemEval 2021 Task 5: Toxic Spans Detection. The task concerns evaluating systems that detect the spans that make a text toxic when detecting such spans are possible. To address the possibly multi-span detection problem, we develop a start-to-end tagging framework on the top of RoBERTa based language model. Besides, we design a custom loss function which take distance into account. In comparison to other participating teams, our system has achieved 69.03% F1 score, which is slight lower (-1.8 and -1.73) than the top 1 (70.83%) and top 2 (70.77%), respectively.

1 Introduction

In recent years, social networks and microblogging sites' popularity have increased, attracting more users. With a huge user base, social media will continue to publish a large amount of user-generated content. As the use of social media increased, other undesirable phenomena and behaviors emerged. Social media users often abuse this freedom to spread abusive or hateful posts or comments. In many cases, the user-generated content is offensive or proactive, and users may have to deal with threats such as cyberattacks or cyberbullying, and other undesirable (Warner and Hirschberg 2012). Therefore, the issue of detecting and possibly limiting the spread of toxic post has become increasingly important.

Although several toxicity or abusive language detection datasets (Wulczyn et al. 2016; Borkan et al. 2019) and models (Borkan et al. 2019; Pavlopoulos et al. 2017; Zampieri et al. 2019) have been released, most of them classify whole comments or documents, and do not identify the spans that make a text toxic. But highlighting such toxic spans can assist human moderators (e.g., news portals moderators) who often deal with lengthy comments, and who prefer attribution instead of just a system-generated unexplained toxicity score per post. The evaluation of systems that could accurately locate toxic spans within a text is thus a crucial step towards successful semi-automated moderation.

For this reason, SemEval 2021 set up the task Toxic Spans Detection to detect and extract the spans that make a text toxic, when detecting such spans is possible (Pavlopoulos et al. 2021). To address the possibly multi-span extraction problem, we develop a start-to-end tagging framework with custom distance loss, which can tag the start and end position of a toxic span. Based on this scheme, we can effectively deal with the multi-span extraction problem.

The rest of the paper is organized as follows: Section 2 provides system overview. Section 3 describes our approach in detail. Our experiment is discussed in Section 4. We conclude our work in Section 5.

2 System Overview

2.1 Preprocessing and Word Embedding

The training dataset contains 3 columns:

ID - Contains a unique number to identify each training example.

Spans - Contains a list of indexes that indicates the position of toxic spans.

Text - Contains the text that need to detect and extract the toxic spans.

Note that the spans are not given in text, We transformed the indexes to text first. Besides, we append the "negative" word to the end of each post serving as the indicator. We use word embeddings as input to the model. Word embedding is a distributed vector representation of words (Mikolov et al. 2013), capturing the syntactic and semantic in-

* Corresponding author.

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 258–262
Bangkok, Thailand (online), August 5–6, 2021.

formation of words. Effective word embedding can get better performance. After comparison, we use the RoBERTa-based pre-training language models as sentence encoder for word embedding.

2.2 Sequence Tagging

Sequence tagging as a general methods can be used in wide applications, such as named entity recognition, relation extraction, machine reading comprehension and so on.

The tagging scheme can be divided into BIO (Zheng et al. 2017), BIOES (Huang et al. 2015) and others, in which B denotes the first token of an output span, I denotes subsequent tokens in a span, O denotes tokens that are not part of an output span, E denotes the last token of an output span and S denotes token that is an output span.

3 Model Description

Our model has two steps as follows: 1. Concatenate the "negative" word at end of each post. 2. Obtain the word embedding of each token in the post to form the final representation and predict the start and end probabilities for each token as output.

Figure 1 shows the general structure of the system. More details for the systems components are shown in the following subsections.

3.1 Embedding Layer

As input sequence X of length T is composed of word tokens: $X = \{x_1, \ldots, x_T\}$. Each token x_t is replaced with the corresponding vocabulary index $V(t)$. The embedding layer transforms the token into vector $e_t \in R^d$ which is selected from the embedding matrix E according to the index, where d is the dimensionality of the embedding space.

In order to indicate the model extract toxic or negative spans, we append the word embedding vector of "negative" to the end of each post. We take the mean of last two hidden layer's weight as word embedding. The example of sentence constructed is also shown in Figure 1.

3.2 Tagging scheme

Although the classical BIOES tag based model can obtain competitive result, we think the training dataset is not big enough to learn so many tags. So different the above methods, we apply the start-to-end tagging scheme that predicting start and end probabilities for each token. The different target sequence used by several loss function are as shown in Figure 1.

3.3 Loss Function

3.3.1 Classical Cross-Entropy Loss

At the beginning, we use the classical binary cross-entropy loss, which creates a criterion that measures the Binary Cross Entropy between the target and the output. The loss can be described as:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top$$

$$l_n = -\left[y_n \cdot \log x_n + (1 - y_n) \cdot \log\left(1 - x_n\right)\right]$$

3.3.2 Label Smoothing Loss

Consider that, we are using roBERTa(Liu et al. 2019) as encoder, which is a large pre-trained language model. and may cause the over-fitting problem. To prevent this, we apply the label smoothing(Szegedy et al. 2015) method and change the '0' in the target sequence to small value 0.025. The computation method is same with cross-entropy loss.

3.3.3 Kullback-Leibler Divergence Loss

Besides the handcrafted label smoothing loss, we also tried the KLDivLoss, which is a useful distance measure for continuous distributions and is often useful when performing direct regression over the space of (discretely sampled) continuous output distributions.

The target sequence is the same with the above binary cross-entropy loss. The loss can be described as:

$$l(x, y) = L = \{l_1, \ldots, l_N\}$$

$$l_n = y_n \cdot (\log y_n - x_n)$$

where the index *N* spans all dimensions of input and *L* has the same shape as input.

3.3.4 Custom Distance Loss

We notice that the cross-entropy loss pay equal weight to each position's loss, no matter how far the distance between it and the target. To penalize more on the distant false prediction, we propose a custom distance loss, which use an auxiliary sequence that generated by insert equal interval from 0 to 1 center on the '1' target. And use the mean dot product to compute the distance loss.

4 Evaluation

4.1 Data

The shared task provides trail, training and testing datasets to be used by all participants. The statistics

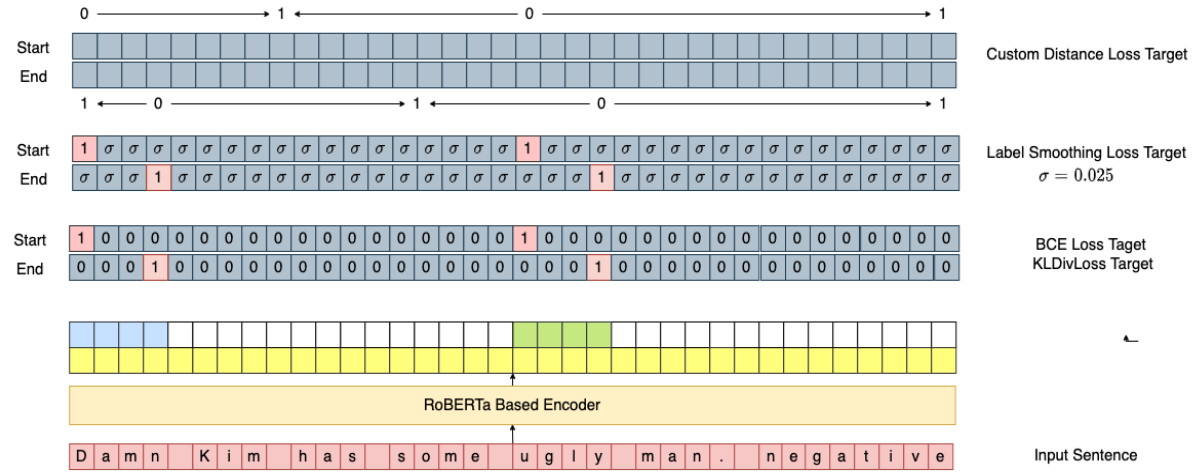

Figure 1: Start-to-end Tagging Framework

of trail, training and testing dataset can be shown in Table 1.

	Trail	Train	Test
Without span	43	485	394
With span	647	7454	1606
Total	690	7939	2000

Table 1: Datasets for SemEval-2021 Task 5

In this task, we apply the 5-fold cross-validation method and only use the official training data set for training and validating.

4.2 Evaluation Measure

To evaluate the responses of a system, we employ the F1 score, as in Martino et al. 2019. Let system $\mathrm{A_i}$ return a set $\mathrm{S^t_{A_i}}$ of character offsets, for parts of the post found to be toxic. Let G^t be the character offsets of the ground truth annotations of t. We compute the F1 score of system $\mathrm{A_i}$ with respect to the ground truth G for post t as follows, where $||$ denotes set cardinality.

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i,G) \cdot R^t(A_i,G)}{P^t(A_i,G) + R^t(A_i,G)}$$

$$P^t(A_i, G) = \frac{\left|S^t_{A_i} \cap S^t_G\right|}{\left|S^t_{A_i}\right|}$$

$$R^t(A_i, G) = \frac{\left|S^t_{A_i} \cap S^t_G\right|}{\left|S^t_G\right|}$$

If $\mathrm{S^t_G}$ is empty for some post t (no gold spans are given for t), we set $\mathrm{F1^t\,(A_i, G) = 1}$ if $\mathrm{S^t_{A_i}}$ is also empty, and $\mathrm{F1^t\,(A_i, G) = 0}$ otherwise. We finally average $\mathrm{F1^t\,(A_i, G)}$ over all the posts t of an evaluation dataset T to obtain a single score for $\mathrm{A_i}$.

4.3 Experiments

The model is implemented using Pytorch (Paszke et al. 2019). We experiment with RoBERTa (Liu et al. 2019) based pre-trained language model as encoder, including RoBERTa-base-squad2(Deepset), twitter-RoBERTa-base-sentiment (Barbieri et al. 2020) and DistillRoBERTa-base (Sanh et al. 2019) And we take the average of last two hidden layers's weights as embedding. Our model is trained with AdamW (Loshchilov and Hutter 2017) optimizer with initial learning rate 0.00003 and weight decay coefficient 0.012. The max sequence length is 512 and dropout (Srivastava et al. 2014) rate is 0.5 to prevent our model from over fitting. And the threshold is set to 0.5. The final submission which scores 69.03 is equipped with both the bi-nary cross-entropy loss, custom distance loss and voting ensemble mechanism.

4.4 Results and Analysis

In order to evaluate the effect of the custom loss function, we compare our approach with its variant.

Variant 1: The variant only use the cross-entropy loss.

Variant 2: The variant only use label smoothing loss.

Variant 3: The variant only use Kullback-Leibler divergence loss.

We take DistillRoBERTa-base as encoder for all of the above experiments. Table 2 show that the variant 2 model has the lowest score, which

Model	F1 score
Variant 1	0.6662
Variant 2	0.6347
Variant 3	0.6618
Our Model	**0.6754**

Table 2: Performance of Our System and Its Variants

may cased by the enormous '0' label. Besides, the model with binary cross-entropy loss and custom distance loss obtains the best result. Thus we decide to use the model as our final ensembling element.

4.5 Voting Ensemble

As mentioned above, we tried several RoBERTa besed pre-trained language model as encoder. In this section, we will discuss the performance difference between them.

Encoder	F1 score
DistillRoBERTa-base	0.6754
RoBERTa-base-squad2	**0.6793**
twitter-RoBERTa-base-sentiment	0.6742

Table 3: Performance of Different Encoder

As shown in Table 3, we can find that the overall score's difference is slight. But when we take a closer look at the performance, the result on single example is different. And the RoBERTa-base-squad2 encoder achieved best result, which may caused by the training method.

Text: "good side of trump? are you kidding me? trump has no good side all bad, he is divisive, a racist and bigot, pathological liar, scammer, tax cheat, sexual pervert,"

Golden Spans: ['pathological liar', 'scammer', 'sexual pervert']

DistillRoBERTa-base: ['racist and bigot', 'sexual pervert']

RoBERTa-base-squad2: ['racist', 'scammer', 'sexual pervert']

twitter-RoBERTa-base-sentiment: ['racist and bigot, pathological liar, scammer', 'sexual pervert']

As shown above, Complementing and correcting each other may improve the overall performance due to the difference. This is exactly what ensemble learning is good at. Ensembling of several models is widely used method to improve the performance of the overall system by combining predictions of several models, such as as for they provide complementary information.

Considering this, we decide to apply the model ensemble methods, particularly the vote mechanism was applied. In which, if the number of occurrences of one index is bigger than 3 in the all above model's predictions, the index will be add to the final result, otherwise it will be exclude. The ensemble result obtains 69.03% F1 score on the test data set without any rule correction or dictionary based post process. Our model ranks in the top 10 among nearly 100 participating teams with slight lower (-1.8 and -1.73) than the top 1 (70.83%) and top 2(70.77%), respectively.

5 Conclusion and future work

In this paper, we propose a start-to-end tagging framework with custom distance loss function for SemEval-2021 Task 5. The performance of our model which is equipped with distance loss and voting mechanism better than its variants. But the distance loss target is assigned manually, which may have low generalization ability to different data set and task. We will try to improve its performance and apply this tagging scheme to other task in future work.

References

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561.

Deepset. Farm. https://github.com/deepset-ai/FARM.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

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. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news articles. *CoRR*, abs/1910.02517.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

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

John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper attention to abusive user content moderation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, Copenhagen, Denmark. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Montréal, Canada. Association for Computational Linguistics.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2016. Ex machina: Personal attacks seen at scale. *CoRR*, abs/1610.08914.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. 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, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.