S-NLP at SemEval-2021 Task 5: An Analysis of Dual Networks for Sequence Tagging

Viet Anh Nguyen *, **Tam Minh Nguyen** *, **Huy Quang Dao** *, **Quang Huu Pham** *
AI Research Team
R&D Lab, Sun Asterisk Inc.
{nguyen.viet.anh-d, nguyen.minh.tam-b,
dao.quang.huy-b, pham.huu.quang}
@sun-asterisk.com

Abstract

The SemEval 2021 task 5: Toxic Spans Detection is a task of identifying considered-toxic spans in text, which provides a valuable, automatic tool for moderating online contents. This paper represents the second-place method for the task, an ensemble of two approaches. While one approach relies on combining different embedding methods to extract diverse semantic and syntactic representations of words in context; the other utilizes extra data with a slightly customized Self-training, a semi-supervised learning technique, for sequence tagging problems. Both of our architectures take advantage of a strong language model, which was fine-tuned on a toxic classification task. Although experimental evidence indicates higher effectiveness of the first approach than the second one, combining them leads to our best results of 70.77 F1-score on the test dataset.

1 Introduction

Social Network sites are an integral part of our society. These platforms are often designed to maximize user interaction without sufficient means to moderate such interactions. The amount of users being cyber-bullied by toxic comments has reached an alarming proportion (Chan et al., 2021). To efficiently maintain the health of online communities, an automatic online-content filtering tool needs to be developed. Numerous previous attempts to resolve this issue have focused on toxic comment classification (Georgakopoulos et al., 2018; Chu et al., 2017; Pham et al., 2020; Risch and Krestel, 2020). Although these classification models are capable of detecting toxic comments, their outputs are not interpretable (Mathew et al., 2020).

On the other hand, Toxic Spans Detection (Pavlopoulos et al., 2021) is a task of locating toxic segments in texts. With such a system, the moderators can easily highlight offensive words in comments, which is an essential and explainable assistance for automated comment rating. In this paper, we propose our two approaches to resolve the task. Our contributions are as follows:

- We investigate the effectiveness of our slightly customized Self-training (Wei et al., 2021) technique for a sequence tagging problem - Toxic Spans Detection.
- We explore the benefits of combining different word representations including Byte Pair Encoding (Sennrich et al., 2015), contextual character-level (Akbik et al., 2018), FastText (Bojanowski et al., 2016) and RoBERTa (Liu et al., 2019) word embeddings in order to utilize different syntactic and semantic information learned by these embedding methods.
- Taking advantage of a well-domain-adaptive pre-trained language model on a classification task (Unbiased-toxic-RoBERTa (Hanu and Unitary team, 2020)), we successfully integrate our two above-mentioned methods to achieve a high F1-score of 70.77 and rank 2nd at the Semeval 2021 Task 5: Toxic Spans Detection.
- Numerous exciting insights of the system's performance have been drawn with detailed error analysis.

2 Related Work

2.1 Word representation learning

Word2Vec (Mikolov et al., 2013) is among the earliest models for extracting continuous word representations. Although there have been numerous modern pre-trained text embeddings that outperformed Word2Vec in downstream tasks, it is still widely

* equal contribution

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 888–897
Bangkok, Thailand (online), August 5–6, 2021.

used due to its simplicity and effectiveness (Akbik et al., 2018). However, Word2Vec fails to handle rare or out-of-vocabulary words. To address this problem, FastText (Bojanowski et al., 2016) learn a word representation as sum of its character n-grams embeddings. On the other hand, (Sennrich et al., 2015) utilizes Byte Pair Encoding, an alternative approach for learning sub-word representations.

Recent pre-trained language models learn context-sensitive word representations by utilizing different pretext tasks namely autoregressive language modeling (Radford et al., 2019; Akbik et al., 2018), masked language modeling (Devlin et al., 2018) on a large amount of unlabeled data. Those methods have led to significant improvements in a wide range of downstream tasks, including Text Classification (Howard and Ruder, 2018), Question Answering (Devlin et al., 2018) and Named Entity Recognition (Akbik et al., 2019b).

Unbiased Toxic RoBERTa (Hanu and Unitary team, 2020) is a language model that utilizes general pre-trained RoBERTa (Liu et al., 2019) to continually pre-train a toxic comment classification task on Civil Comments Dataset [1]. This toxic-domain-adaptive language model can be successfully employed to Toxic Spans Detection task whose domain is a subset of Civil Comments.

2.2 Self-training

Self-training, a semi-supervised method, incorporates the prediction of teacher models on extra available in-domain unlabeled data into the training of a student model (Wei et al., 2021). Self-training has been recently successfully applied in both Computer Vision and Natural Language Processing tasks, including Image Classification (He et al., 2018), Object Detection (Xie et al., 2020), Machine Translation (He et al., 2020), ect. Despite its merits, issues such as the lack of in-domain unlabeled data (Du et al., 2020) and unreliable-pseudo labels (Pham et al., 2021) are the main obstacles for the success of Self-training.

For sequence-tagging problems, there are various methods of coping with noisy-pseudo labels. Unlike classification tasks, noisy self-labeled data can be easily eliminated by removing those which have low confidence scores; there is a lack of a comprehensive means to determine this score for a sequence-labeling data point. In several recent research, a deep reinforcement learning (Chen et al., 2018) and meta-learning (Wang et al., 2020) has been proposed to reduce "error propagation from noisy pseudo-labels" for sequence labeling tasks.

[1] https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification

3 Methodology

In this section, we describe our proposed framework in detail. Firstly, we develop a simple but strong baseline to discover the effectiveness of different backbone models. Consequently, we build, extend, and customize our two methods on top of the best backbone and baseline model.

3.1 Baseline

We consider this task as a word-level binary classification problem even though the label annotation of the dataset is at character-level. Therefore, we first align character annotations to word annotations. We utilize a straightforward architecture, with a pre-trained language model as the backbone and a simple classifier on top of it. Specifically, let denote $\mathbf{w} = \{w_1, w_2, ..., w_m\}$ and $\mathbf{y} = \{y_1, y_2, ..., y_m\}$ with w_i, y_i is the word and its label at position i respectively, and $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ with x_j is the j^{th} subword tokens. Notice here that m and n can be different because language models learns subword representations instead of word-embeddings. $\mathbf{h} = \{h_1, h_2, ..., h_n\}$ is the set of contextual embedding for all tokens in $\mathbf{x}$ (taken from the last layer's output of the backbone) and $\mathbf{p} = \{p_i, p_2, ..., p_m\}$ with p_i is the set of all subword positions of the word at position i. To obtain a word-level embedding, we took the sum of its corresponding subword embeddings.

$$e_i = \sum_{j \in p_i} h_j$$

Then probability distribution of the word at position i is formulized as follow.

$$p(w_i) = Softmax(W_c ReLU(W_h e_i + b_h) + b_c)$$

With W_c, b_c and W_h, b_h are the learnable weights and bias respectively. We optimize the model by minimize the Cross Entropy loss between the ground-truth and model predictions.

3.2 Method 1: Feature-based Learning

We customize and extend the baseline model by constructing a standard Named Entity Recognition

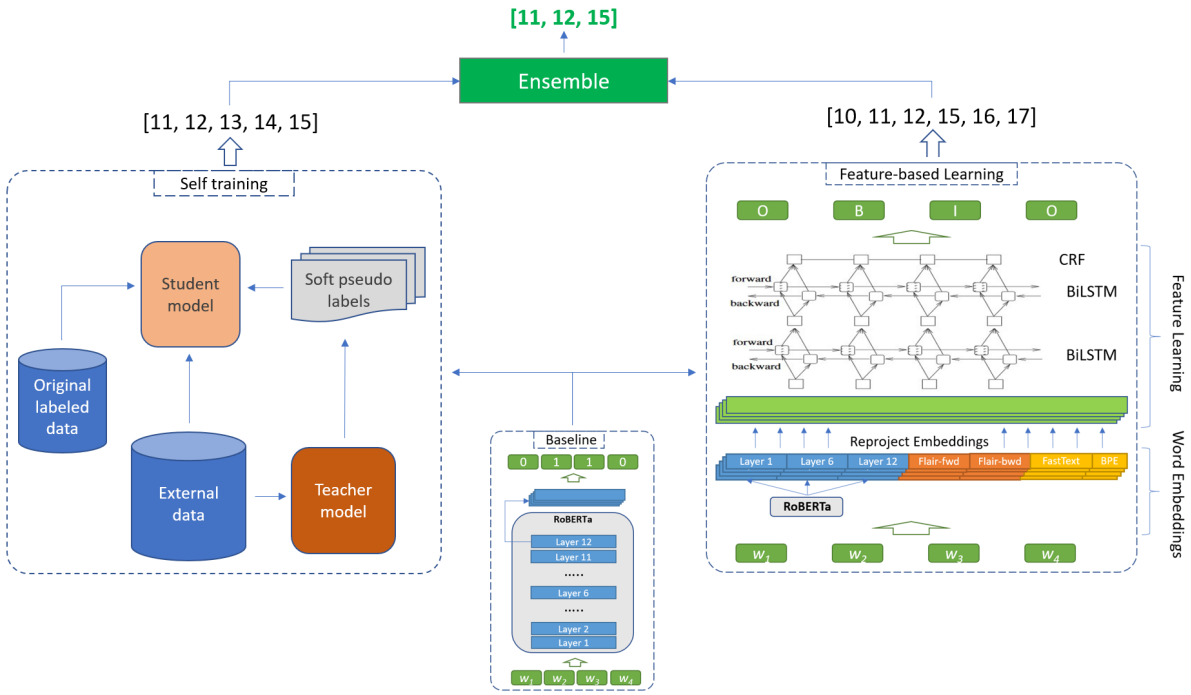

Figure 1: An illustration of our method. We start with a baseline, a simple sequence tagger utilizing Toxic RoBERTa as the backbone. In the Self-training branch, the teacher- the best-scored baseline, generates soft pseudo labels for the student to learn. On the other hand, the Feature-based Learning model concatenates the input vector with different embedding methods i.e. Flair, FastText and BPE, then trains the Named Entity Recognition task. Predicted character offsets (for each sentence) of two models are combined using Intersection Union (Ensemble Section) to obtain the final prediction.

model using Flair package (Akbik et al., 2019a) [2] in which each span is an entity encoded in IOB format. The model consists of two parts: input representation using diverse embeddings and a feature-based model.

3.2.1 Input Representation

In represent both syntactic and semantic information of a word, we combine embeddings extracted by different word embeddings methods. These representations strengthen advantages of each other while mutually easing their weaknesses.

These word embeddings and their usage in our works are as follows:

- **Flair**: Contextual Flair model works on character level. We fine-tune two models 'news-forward' and 'news-backward', on the Next Character Prediction task (Akbik et al., 2018), with 600K toxic texts from the Civil Comment Dataset to adapt them to toxic comment domain.

- **Toxic RoBERTa**: To utilize contextual embeddings from Toxic RoBERTa, besides features derived from the last layer as our baseline, we concatenate two more layers: the first one (layer 1) and the middle one (layer 6). This choice allows the feature learning to understand three levels of context-specificity (Ethayarajh, 2019). The final word representation is obtained by taking the sum of its subword embeddings.

- **FastText with Byte Pair Embedding**: It has been practically proven that combining contextual embeddings with static embeddings improves the performance of many NLP downstream tasks (Peters et al., 2018). We discard subword part, take only word vector part of a FastText model (pre-trained on Common Crawl dataset) for word representation and utilize an external English Byte Pair Embedding for out-of-vocabulary functionality. This combination performs as well as the original FastText while effectively reduces memory usage.

All of the above embeddings are concatenated to form a long vector for each word, which is digested by a feature learning model.

[2] https://github.com/flairNLP/flair

3.2.2 Feature Learning

The feature learning part is a sequence-to-sequence model that takes a sequence of word vectors and learns higher-level features and inferences tags. We use a linear layer to reproject the word embeddings onto a vector space with dimensions equal to the length of concatenated word embeddings. Two follow-up BiLSTM (Hochreiter and Schmidhuber, 1997) (Dyer et al., 2015) blocks are added to learn high-level semantic-syntactic dependencies of the sequence. Finally, a Conditional Random Fields (Sutton and McCallum, 2010) layer, placed on top of the BiLSTMs, makes tag prediction for each word.

3.3 Method 2: Self-training With In-domain Unlabeled Data

3.3.1 In-domain data retrieval

In-domain unlabeled data is one of the determining factors for Self-training. The Toxic Spans Detection task's labeled dataset is a subset of toxic-and-severe-toxic-labeled data in Civil Comment Dataset (Pavlopoulos et al., 2021). To retrieve additional data, we first selected posts classified as toxic by at least half of its toxicity annotators. After removing texts in both train and trial labeled datasets from the retrieved data, we randomly select a subset of 30,000 unseen texts for the task. The choice of extra datasets' size is heuristic and limited due to low-computing resources.

3.3.2 Data filtering and soft label

We slightly customized the pseudo-labels distillation process applied in classification tasks (He et al., 2018) for the sequence-tagging problem. Instead of evaluating and selecting each text in unlabeled data, we use the teacher model's post-softmax class probabilities to evaluate and select each word in a context. Specifically, if each word's confidence score is greater than a threshold, we keep the back-propagation process through that word; otherwise, we ignore it. Notice here that the probabilities mentioned above are also utilized as confidence scores and pseudo-labels for the student training.

3.3.3 Combine generated-labeled data with original-labeled data

The student model is trained on a combination of original-labeled and synthetic-labeled datasets. It has the same architecture as the teacher model except for increases in dropout rates of dropout layers and the hidden size in the model's head classifier. We chose the best checkpoint of the baseline model as teacher model.

3.3.4 Post-processing

For each continuous toxic-predicted span, we eliminate any existing punctuation at both its beginning and end. Additionally, to partially prevent our model from predicting common toxic comments' targets as toxic spans, we exclude any predicted span in our predefined list of targets (described in details in the Appendices section). This list is based on the identity-targets list of toxic comments in the Civil Comments Dataset.

3.4 Ensemble Learning

We combine our two approaches by taking intersection (Intersection Ensemble) or the union (Union Ensemble) of predicted character offsets generated by best model results, from each method, to obtain the final offsets for each sentence.

$$\mathbf{S_I} = \mathbf{S_1} \cap \mathbf{S_2} = \{x : x \in \mathbf{S_1} \text{ and } x \in \mathbf{S_2}\}$$
$$\mathbf{S_U} = \mathbf{S_1} \cup \mathbf{S_2} = \{x : x \in \mathbf{S_1} \text{ or } x \in \mathbf{S_2}\}$$

With $\mathbf{S_I}$, $\mathbf{S_U}$, $\mathbf{S_1}$,$\mathbf{S_2}$ are the intersection, union Feature-based Learning and Self-training offset predictions for one sentence of the ensemble model respectively.

Figure 1 illustrates our composed framework: the two approaches, built and extent on top of the baseline, are combined for the final predictions.

4 Experiments

4.1 Dataset

The original dataset contains 7939 annotated samples for training and 2000 unlabeled samples for testing. We use a small trial dataset, given by the task organizer which consists of 690 labeled samples, as our development set. We train our models on the training set, use the development set to find the best hyper-parameters, and finally make our submission on the private test set.

4.2 Experiment setup

This section focused on the hyper-parameters configurations of our two methods and is mentioned in the Appendices section.

4.3 System Configuration

Our experiments are conducted on a computer with Intel Core i7 9700K Turbo 4.9GHz, 32GB of RAM, GPU GeForce GTX 2080Ti, and 1TB SSD hard disk.

4.4 Evaluation Metric

The evaluation metric of our system is defined, by the task organizer (Pavlopoulos et al., 2021) , as follow:

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

$$P^t\left(A_i, G\right) = \frac{\left|S_{A_i}^t \cap S_G^t\right|}{\left|S_{A_i}^t\right|}$$

$$R^t\left(A_i, G\right) = \frac{\left|S_{A_i}^t \cap S_G^t\right|}{\left|S_G^t\right|}$$

$$\text{if } S_G^t = \{\phi\} \Rightarrow F_1^t\left(A_i, G\right) = \begin{cases} 1 \text{ if } S_{A_i}^t = \{\phi\} \\ 0 \text{ otherwise} \end{cases}$$

$$F_1^T\left(A_i, G\right) = \frac{1}{n} \sum_{t=1}^{n} F_1^t\left(A_i, G\right)$$

With:

- $\mathbf{S_{A_i}^t}$: character offsets of toxic post $\mathbf{t}$, output of system $\mathbf{A_i}$
- $\mathbf{G^t}$: ground truth character offsets of toxic post $\mathbf{t}$
- $\mathbf{F_1^t}\left(\mathbf{A_i}, \mathbf{G}\right)$: $\mathbf{F_1}$ score of system $\mathbf{A_i}$, with respect to ground truth $\mathbf{G^t}$ of post $\mathbf{t}$
- $\mathbf{F_1^T}\left(\mathbf{A_i}, \mathbf{G}\right)$: $\mathbf{F_1}$ score of system $\mathbf{A_i}$ on dataset $\mathbf{T}$
- $|.|$: set cardinality

4.5 Results

4.5.1 Baseline result

Table 1 indicates the performances of our baseline model with two different backbones, RoBERTa (Liu et al., 2019) and Unbiased Toxic RoBERTa (which is refered as Toxic RoBERTa for the rest of the paper) (Hanu and Unitary team, 2020). The toxic domain-adaptive pre-trained language model outperforms general RoBERTa by a large margin (up to 0.68), which sheds light on the necessity of adapting universal representations to task-specific domains.

Backbone	**Private test F1-score**
RoBERTa	68.62
Toxic RoBERTa	**69.30**

Table 1: Performances of the baseline model with two different backbones.

4.5.2 Feature-based Learning result

We froze Toxic RoBERTa backbone in all experiments of feature-based learning except the last one. This latest experiment compares the differences in model performances between tuning and not tuning Toxic RoBERTa.

Word Embeddings	**Private test score**
Toxic RoBERTa	69.89
FastText w/ BPE	67.89
Flair	67.92
Toxic RoBERTa + Flair	69.99
Toxic RoBERTa + FastText w/ BPE	69.95
Toxic RoBERTa + Flair + FastText w/ BPE	**70.26**
Toxic RoBERTa (fine-tuned) + Flair + FastText w/ BPE	67.37

Table 2: Results of different embedding combinations for method one

Table 2 shows the feature-based model's performance with different word embeddings and the gap in F1-score between feature-based and fine-tuning models. Our findings are as follows:

Toxic RoBERTa was the best feature extractor since using it achieved a competitive F1-score of 69.89. On the other hand, using only Flair results in a slightly better performance than FastText with BPE (67.92 and 67.89 respectively).

Adding more features (learned by Flair or FastText with BPE embeddings) to ones learned by Toxic RoBERTa improved F1-score (69.99 and 69.95 respectively). Ultimately, combining all the word-representations obtained the highest score at 70.26.

Fine-tuning RoBERTa dramatically decreased the performance (up to 3-4).

4.5.3 Self-training result

Table 3 presents the performance result of the 2nd method. Our choice of the teacher was the best-performed baseline model with 69.30 F1-score. Post-processing enhanced this performance, resulted in 69.44 F1-score. Self-training only leads to a better student with an improvement of 0.1 compared to the post-processed teacher model. We suspect that this unimpressive increase is due to the teacher model's confirmation bias and the unsolved issue of noisy-pseudo labels (Pham et al., 2021).

Backbone	Private test F1-score
Teacher w/o Post processing	69.30
Teacher w/ Post processing	69.44
Student	**69.54**

Table 3: Performances of the teacher model with and without post-processing and student model.

4.5.4 Ensemble learning result

Table 4 illustrates the effectiveness of our ensemble methods. Intersection Ensemble results in a significant improvements of our system prediction (0.51 and 1.23 compared to Feature-based Learning and Self-training respectively) while Union Ensemble leads to a substantial decrease of F1-score (-1.14 and -0.42 compared to method 1 and 2 respectively). This exciting finding indicates that Intersection Ensemble can rule out numerous falsely positive tokens of our two models whereas Union Ensemble worsen the performance by integrate these false positives.

Method	Private test F1-score
Feature-based Learning	70.26
Self-training	69.54
Union Ensemble	69.12
Intersection Ensemble	**70.77**

Table 4: Performance of ensemble models with different ensemble methods.

5 Error Analysis

Carefully analyzing errors made by our ensemble model on the test dataset has shed light on our system's weaknesses. The following terms are defined to aid the reading of this section:

Figure 2: Analysis of the detection of lengthy toxic spans.

- Toxic span: ground-truth toxic span.
- Long span: contains at least two words.
- True positive span: all words in the span are correctly predicted as toxic.
- False negative span: at least one word in the toxic span are predicted as non-toxic.

5.1 Unsuccessful detection of lengthy toxic spans

Among false negative spans in our system's predictions, 218 spans are short spans, which constitutes a small portion of the total short spans of the test set. In contrast, the other 158 spans are long spans while the total number of long spans of the test set is 200. This means our model had a bad performance on lengthy toxic spans with false negative rate up to 79%.

Figure 2 further investigates on long span failure. While 90% (38 spans/42 spans) of our correctly predicted long spans contains only two word, this numbers for false-negative spans decrease to 62% (98 spans/158 spans). Longer spans are harder to predict. Failing to detect spans with more than three words contributes approximately 38% to the model's false-negative.

Table 5 demonstrates several instances of the failure.

5.2 False negatives due to post-processing

Although experimentally showed its effectiveness, post-processing after Self-training inevitably excluded continuous predicted spans such as "NIGGERS" and "MUSLIMS" which are potentially a part of ground-truth toxic spans.

Ground-truth	Prediction
Bunch of cowards	cowards
burn them at the stake	burn them
big, fat, orange buffoon	None
you're a fecking write-off	None

Table 5: Several instances of errors in detecting long-spans

I wish **this moron would have been shot to death** by the US soldier instead of the other way around. I hope that 10 million is given to the soldier's family who actually deserve it.
Our **ignorant**, clueless legislators are working hard to ensure that Alaskans are as **ignorant** and clueless as the legislators themselves are. **Stupid** Alaskans are the only way these **moronic** legislators will continue to get re-elected.
Trash, and just not the regular **bigoted** flatulence, this crap you write is evil trash

Note: Underlines are the prediction of our models and bold text are our manually annotated toxic-spans.

Table 6: Several examples of our model predictions on no-span texts, which may have been mis-annotated

5.3 Failure due to mis-annotated spans

We notice our model predicted false positive tokens in 469 toxic comments and most of them (308 comments) are humanly annotated with no toxic spans. In our opinion, many of these texts are mis-annotated, which potentially lower the precision of our system.

Table 6 presents several examples of this issue. The underlines are our model predictions, while the bold text spans are our opinion of what toxic annotations should be for the given text. All these texts contain no toxic spans, according to the dataset's annotators.

6 Conclusion

In this paper, we proposed a system to resolve the SemEval task 5: Toxic Spans Detection. Our method utilized a pre-trained language model in toxic-domain and successfully combined two approaches Self-training and Feature-based Learning to achieve a high F1-score of 70.77. Finally, we provided insights into failure of the system and the task's potential falsely-negative annotations issue with careful error analysis.

Despite our success on the leader board, in future research, we determine to improve our model as follow:

- Investigate a solution for the noisy-pseudo label issue to enhance the performance of the Self-training method.
- Combine Self-training with Feature-based Learning to learn a more robust toxic-span detection model.

Acknowledgment

This work is partially supported by ***Sun-Asterisk Inc***. We would like to thank our colleagues at ***Sun-Asterisk Inc*** for their advice and expertise. Without their support, this experiment would not have been accomplished.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019a. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019b. Pooled contextualized embeddings for named entity recognition. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, page 724–728.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Tommy K.H. Chan, Christy M.K. Cheung, and Zach W.Y. Lee. 2021. Cyberbullying on social networking sites: A literature review and future research directions. *Information Management*, 58(2):103411.

Chenhua Chen, Yue Zhang, and Yuze Gao. 2018. Learning how to self-learn: Enhancing self-training using neural reinforcement learning. pages 25–30.

T. Chu, Kylie Jue, and Max L. Wang. 2017. Comment abuse classification with deep learning.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. Self-training improves pre-training for natural language understanding.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings.

Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, and Vassilis P. Plagianakos. 2018. Convolutional neural networks for toxic comment classification. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, SETN '18, New York, NY, USA. Association for Computing Machinery.

Laura Hanu and Unitary team. 2020. Detoxify. Github. https://github.com/unitaryai/detoxify.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation.

Kaiming He, Ross Girshick, and Piotr Dollár. 2018. Rethinking imagenet pre-training.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

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.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. Hatexplain: A benchmark dataset for explainable hate speech detection.

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

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*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V. Le. 2021. Meta pseudo labels.

Q. H. Pham, V. Anh Nguyen, L. B. Doan, N. N. Tran, and T. M. Thanh. 2020. From universal language model to downstream task: Improving roberta-based vietnamese hate speech detection. In *2020 12th International Conference on Knowledge and Systems Engineering (KSE)*, pages 37–42.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Julian Risch and Ralf Krestel. 2020. Toxic comment detection in online discussions.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units.

Charles Sutton and Andrew McCallum. 2010. An introduction to conditional random fields.

Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2020. Adaptive self-training for few-shot neural sequence labeling.

Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. 2021. Theoretical analysis of self-training with deep networks on unlabeled data.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. 2020. Self-training with noisy student improves imagenet classification.

A Appendices

To form our target list in 3.3.4, we include original form, plural form, upper or lower case of each word in the table 7 to that list.

Target Identities	Male, female, transgender, heterosexual, homosexual, gay, lesbian, bisexual, Christian, catholic, jewish, Muslim, Islam, hindu, buddhist, atheist, black, white, asian, latino, Nigger, Mexican.

Table 7: Common target identities in Civil Comment dataset.

Table 8 describes hyperparameter configuration for training. For Feature-based Learning, Flair embeddings are fine-tuned before training the NER model. Turn into Self-training, the best baseline model is used as the teacher. If not specified, the corresponding hyper-parameter value is used for training both baseline and student models

Task	Hyperparameter	Value	Description
Fine-tune Flair	pre-trained weights	”news-forward”/ ”news-backward”	Initial weights of the Flair models.
	sequence_length	250	Length of character sequences
	mini_batch_size	500	Size of batches during training
	learning_rate	20	Initial learning rate
	patience	10	Number of epochs without improvement
	max_epochs	5	Number of maximum training epochs
	optimizer	SGD	Optimizer used for training
	scheduler	AnnealOnPlateau	Learning rate scheduler
Train NER model	dropout	0.3995	Probability of an element to be zeroed
	locked_dropout	0.4413	Probability of entire parameters in embedding space to be zeroed
	word_dropout	0.0677	Probability of entire words (or characters) in embedding space to be zeroed
	learning_rate	0.0005	Initial learning rate
	min_learning_rate	1e-07	Minimum learning rate to terminate training
	mini_batch_size	32	Size of batches during training
	max_epochs	50	Number of maximum training epochs
	optimizer	AdamW	Optimizer used for training
	scheduler	AnnealOnPlateau	Learning rate scheduler
Baseline + Self-training	hidden_size_T	150	Size of the linear projection of the teacher’s head classifier
	hidden_size_S	160	Size of the linear projection of the student’s head classifier
	learning_rate	1e-05	The learning rate used for training
	optimizer	AdamW	Optimizer used for training
	scheduler	None	No learning rate scheduler used
	dropout_T	0.3	Dropout rate of all dropout layers in the teacher head classifer
	dropout_S	0.4	Dropout rate of all dropout layers in the student head classifer
	max_epochs	5	Total training epochs
	label_smoothing	0.15	Label smoothing coefficient
	confidence_threshold	0.7	Use in obtaining extra data for student model, all words with post-softmax score (calculated by the teacher model) less than this threshold will be ignored
	batch_size	8	Size of batches during training

Table 8: Hyperparameters for feature-based model training.