IITK@Detox at SemEval-2021 Task 5: Semi-Supervised Learning and Dice Loss for Toxic Spans Detection

Archit Bansal **Abhay Kaushik** **Ashutosh Modi**
Indian Institute of Technology Kanpur (IIT Kanpur)
{architb, kabhay}@iitk.ac.in
ashutoshm@cse.iitk.ac.in

Abstract

In this work, we present our approach and findings for SemEval-2021 Task 5 - Toxic Spans Detection. The task's main aim was to identify spans to which a given text's toxicity could be attributed. The task is challenging mainly due to two constraints: the small training dataset and imbalanced class distribution. Our paper investigates two techniques, semi-supervised learning and learning with Self-Adjusting Dice Loss, for tackling these challenges. Our submitted system (ranked ninth on the leader board) consisted of an ensemble of various pre-trained Transformer Language Models trained using either of the above-proposed techniques.

1 Introduction

Content moderation has become the topic of most conversations regarding social media platforms. However, with over 4 billion active internet users, it is impossible to moderate each piece of message generated online manually. Therefore, the focus is now shifting towards tackling the issue using machine learning methods.

Various toxicity detection datasets (Wulczyn et al., 2017; Borkan et al., 2019) and models (Pavlopoulos et al., 2017; Liu et al., 2019; Seganti et al., 2019) have been successfully developed over the years to tackle the issue of moderation. However, these have mostly focused on identifying whole comments or documents as either toxic or not. In semi-automated settings, a model merely generating a toxicity score for each comment, some of which can be very lengthy, is not of much help to human moderators. To tackle this issue, the *SemEval 2021 Task 5 : Toxic Spans Detection* is introduced (Pavlopoulos et al., 2021). The task involves identifying text spans in a given toxic post that contributes towards the toxicity of that post. The task aims to promote the development of a system that would augment human moderators by giving them more insights into what actually contributes to the text's toxicity.

The task is challenging mainly due to the following reasons: a) small size of the dataset b) characteristics of text samples extracted from social media leading to difficulties such as out-of-vocabulary words and ungrammatical sentences c) class imbalance in the dataset d) inconsistencies in data annotations. We approached this task as a subtoken level sequence labeling task. Fine-tuned pretrained transformer language models (Qiu et al., 2020) are the backbone of all our approaches. We investigated two main techniques to enhance the results of the fine-tuned transformer models, namely Semi-Supervised Learning (Yarowsky, 1995; Liu et al., 2011) and fine-tuning with Self-Adjusting Dice Loss (Li et al., 2020). This paper reports the results of our experiments with these different techniques and pre-trained transformer models. Our submitted system consisted of an ensemble of different pre-trained transformer models and achieved an F1 score of 0.6895 on the test set and secured 9th position on the task leaderboard. All of our code is made publicly available on Github[1].

The rest of this paper is organized as follows. Section 2 discusses the previous works in the fields of offensive language detection and span identification. Section 3 describes the dataset. Section 4 explains the proposed approaches. Section 5 reports the results of various experiments with the proposed approaches, and section 6 analyzes the proposed approaches via ablation studies. We conclude with an error analysis of our model performance in section 7 and concluding remarks in section 8.

[1]https://github.com/architb1703/Toxic_Span

2 Related Work

As the task involves detecting toxic spans in a text, we present the related work in two parts: (i) Offensive Language Detection and (ii) Span Identification.

Offensive Language Detection: Research work has been done on different abusive and offensive language identification problems, ranging from aggression (Kumar et al., 2018) to hate speech (Davidson et al., 2017), toxic comments (Saif et al., 2018), and offensive language (Laud et al., 2020; Pitsilis et al., 2018). Recent contributions to offensive language detection came from the SemEval-2019 Task 6 OffensEval (Zampieri et al., 2019). The task organizers concluded that most top-performing teams either used BERT (Liu et al., 2019) or an ensemble model to achieve SOTA results. Interestingly, the task of locating toxic spans is relatively novel, and its successful completion can be groundbreaking. A recent approach with a narrower scope is by Mathew et al. (2020), who focused on the rationality of decision in the task of hate speech detection.

Span Identification: Span detection/identification tasks include numerous tasks like named entity recognition (NER) (Nadeau and Sekine, 2007), chunking (Sang and Buchholz, 2000) and keyphrase detection (Augenstein et al., 2017). (Papay et al., 2020) analyzed the span identification tasks via performance prediction over various neural architectures and showed that the presence of BERT component in the model is the highest positive predictor for these tasks. Inspired by this observation, we have built our model based on the transformer architecture, further exploiting the benefits of semi-supervised learning and modified Dice Loss.

3 Dataset

3.1 Data Description

The competition dataset comprises around 10K comments extracted from the Civil Comments Dataset and annotated using crowd-raters. The organizers released the dataset in 3 phases: trial, train, and test. The trial dataset consisted of 690 texts, whereas the training dataset consisted of 7939 texts. Moreover, the test set on which our system was finally evaluated consisted of 2000 text samples.

In the initial stages of the competition, we decided to use only the training dataset to build upon our approaches. We further split the training set into train, dev, and test sets for evaluation purposes using an 80:10:10 split (Div A). Once we tested and finalized our approaches, we combined the train and test set of Div A with the trial set as our final training set (Div B). Due to the small size of the dataset, these additions to the training set of Div A will positively impact the model performance. However, to ensure that we could compare our final models with our previous results, we transfer the dev set directly to Div B. Further details regarding the constitution of these splits is provided in the Appendix A.

3.2 Pre-processing

Tokenization: For the sake of preserving the token spans, we first tokenized our data and then performed data cleaning. For tokenizing, we used the NLTK TreebankWord Tokenizer[2], which is a rule-based tokenizer that tokenizes text on spaces and punctuation, hence preserving the original form of the words.

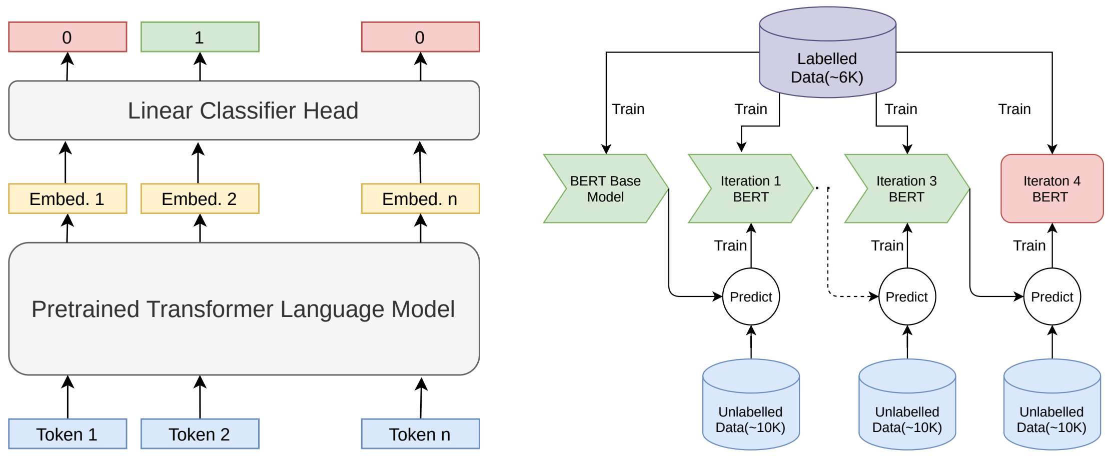
Data Cleaning: We then cleaned each token using different operations such as expanding contractions and removing digits and full stops.

4 Proposed Approach

4.1 Methodology

Pre-trained transformer models built using the transformer architecture (Vaswani et al., 2017) have been able to achieve, via transfer learning techniques, SOTA performance for most NLP tasks in recent times. We fine-tuned pre-trained transformer models with linear classifier head for performing sequence labeling for this task, which meant performing subtoken-level classification (Fig.1a). Our baseline model used the pre-trained BERT-Base-Cased model, fine-tuned with cross-entropy loss and AdamW optimizer. The different hyperparameter values used for training the baseline and all subsequent models are reported in the Appendix B to facilitate replication of results. Subsequently, we improved upon this baseline using two techniques, semi-supervised learning and Self Adjusting Dice Loss. Along with this, we fine-tuned multiple different transformer models like BERT(Devlin et al., 2019), Electra(Clark et al., 2020), DistilBERT(Sanh et al., 2020), and XLNet(Yang et al.,

[2]https://www.nltk.org/_modules/nltk/tokenize/treebank.html



(a) Sequence Labelling with Transformer Model

(b) Semi-Supervised Learning architecture

Figure 1: Model architectures

2020) for our Dice-Loss Approach and found differences in the predictions of the different transformer models to be beneficial for the final ensemble.

Self Adjusting Dice Loss One of the main issues with our dataset was that of class imbalance. For the sub-tokens derived from the BERT-Base-Cased tokenizer, the ratio of toxic to non-toxic sub-tokens was 1:10.16. However, we could not tackle this issue with over/under-sampling due to the nature of our problem, and training with a weighted cross-entropy loss function did not improve results. Therefore, we experimented with training with the Self-Adjusting Dice Loss (Li et al., 2020) which was proposed as an objective function for dealing with imbalanced datasets in NLP. The original dice coefficient is an F1-oriented statistic used to gauge the similarity of two sets. The paper proposed a loss function based on a modified dice coefficient, which they reported to achieve a better F1 score than models trained with cross-entropy loss.

$$DL = 1 - \frac{2(1-p_{i1})^{\alpha}(p_{i1}).y_{i1} + \gamma}{(1-p_{i1})^{\alpha}(p_{i1}) + y_{i1} + \gamma}$$

Here, for the i_{th} training instance, p_{i1} is the predicted probability of positive class and y_{i1} is the ground truth label. The loss function also has two hyperparameters, alpha and gamma, which we tuned for our models.

Semi-Supervised Learning The Civil Comments Dataset from which our training data was extracted consists of over 1 million comments; however, due to annotation constraints, the training set only had 7000 data samples. (Shams, 2014) have shown that for text classification tasks, unlabelled data from a suitable data source could be used to train semi-supervised models that achieve better results than a model trained using supervised learning. Also, (Jurkiewicz et al., 2020) showed that the semi-supervised learning technique of self-training could improve performance on span identification tasks. Hence, we extracted 40000 toxic samples from the Civil Comments Dataset, which were labeled with a toxicity score of 0.7 or higher, and used these to perform four iterations of semi-supervised model training (Fig. 1b). We exhaustively divided the unlabelled samples into four batches of 10000 each and used each batch for exactly one iteration. As shown in Fig. 1b, for each iteration, pseudo labels were predicted for the complete batch using the model trained in the previous iteration, then these pseudo-labels along with the ground truth training labels were used to train the next model. For this approach, we only fine-tuned one transformer model, namely the pre-trained BERT-Base-Cased model.

4.2 Post-preprocessing

After obtaining the sub-token level labels from our model, we post-processed the results to convert them into an array of toxic character offsets. To perform this, we had mapped each sub-token to its offset span during tokenization and used that to retrieve the offsets of all the characters in the toxic sub-tokens. We also include all characters lying between two consecutive sub-tokens if both the sub-tokens are marked toxic. This was necessary as spaces and punctuation were included in the toxic spans given by the annotators, as shown by the results in section 6.

5 Experiments and Results

The competition used the span level F1 score, calculated individually for each text sample from its character offsets and averaged over all the text samples, as the metric to evaluate system performance. We decided to use this metric for hyperparameter tuning and reporting the final results. However, during the training process, the models were checked for overfitting using the token level F1 score, which was also a good indicator of model performance as our training approach was that of a sequence labeling task.

The first set of experiments we performed were on the Div-A dataset. Our baseline model achieved an F1 score of 0.669 on the dev split on this set. The organizers also released a baseline model, consisting of a Spacy statistical model trained on the competition training dataset and evaluated on the competition trial dataset. The organizer's baseline achieved an F1 score of 0.600 on the trial dataset. This score was significantly lower than that of our baseline model, and therefore we use our baseline model only to compare the performance of our subsequent models.

We then fine-tuned a BERT-Base-Cased model with the Self-Adjusting Dice Loss and AdamW optimizer and tuned the loss function's two hyperparameters. The scores we obtained for the different hyperparameter values are reported in Table 9 in Appendix C. We got our best performing model with the hyperparameter values alpha-0.7 and gamma-0.25, achieving an F1 score of 0.6725 on the dev split.

Model	Dev F1 Score
BERT-Base-Cased	0.669
SSL Iteration-1	0.6837
SSL Iteration-2	0.6842
SSL Iteration-3	0.6882
SSl Iteration-4	**0.6893**

Table 1: Results for Semi-Supervised learning model

Next, we trained the BERT-Base-Cased model on the semi-supervised learning paradigm with cross-entropy loss and AdamW optimizer. For the first iteration, we used our baseline model to compute the pseudo labels. The model achieved improved results with each iteration (Table 1), and our final model was scoring 0.6893 on the dev split.

To end this stage of experimentation, we computed the results on the test split of the Div-A dataset. We were able to make two inferences. Firstly the semi-supervised learning model had the best performance with an F1 score of 0.6774 on the test set but had a significantly worse score than it had on the dev set. Secondly, the dice loss trained model performed significantly better than the cross-entropy trained baseline with an F1 score of 0.662 compared to 0.648.

After this, we changed to the Div-B dataset and trained multiple different transformer models with the Self Adjusting Dice Loss. We found that the BERT-Base-Cased, Electra-Small, Electra-Base, and DistilBert-Base-Uncased models had peak performance for the hyperparameter values alpha-0.7 and gamma-0.25. However, for the XLNet-Base model, peak performance was achieved for alpha-0.4 and gamma-0.25. On further experimentation with these models, we also found that adding a full stop to the text samples during evaluation provided consistently better results on the dev set. The results obtained have been reported in Table 2.

Model	WFS	FS
BERT-Base-Cased	0.6754	0.6827
Electra-Small	0.6813	0.6861
Electra-Base	0.6776	0.6846
DistilBERT-Base-Unc.	0.6749	0.6773
XLNet-Base	0.6798	0.6852
SSl Iteration-4	0.6893	0.6932

Table 2: Effect of full stop on dev set during evaluation. Here WFS and FS represent without full stop and with full stop resp.

The final results of models trained either on modified Dice Loss or using Semi-Supervised learning, with full stop added during evaluation, are reported on the Div-B dev split and the competition test set in Table 3.

6 Ablation Study

After the competition, we wanted to study the effect of our different preprocessing and postprocessing techniques. We employ three main data cleaning techniques during our preprocessing, expanding contractions, removing numbers, and removing full stops. To study each particular technique's impact, we created three new Div-B datasets, each having

	Model	Dev	Test
1	BERT-Base-Cased*	0.6827	0.668
2	Electra-Small*	0.6861	0.6771
3	Electra-Base*	0.6846	0.6720
4	DistilBERT-Base-Unc.*	0.6773	0.6822
5	XLNet-Base*	0.6852	0.6757
6	SSl Iteration-4	0.6932	0.672
	Ensemble (1,2,3,4,5,6)	0.6927	**0.6895**

Table 3: F1 Score on dev and competition test set
* - Models trained with modified Dice Loss

Example Set	Val	Test
E.S Val:41 Test:394	0.0731	0.0380
N.E.S Val:753 Test:1606)	0.7265	0.8493
All Val:794 Test :2000	0.6927	0.6895

Table 5: System performance over empty span (E.S) and non-empty span(N.E.S) examples over Div-B split

one of the preprocessing techniques missing. We then trained BERT-Base-Cased and Electra-Small models with Self Adjusting Dice Loss on each of these sets and evaluated the performance on their respective dev sets. The results are reported in Table 4 with the following acronyms:

- **TD** - All preprocessing steps used
- **WNUM** - Without removing numbers
- **WFS** - Without removing fullstops
- **WCON** - Without expanding contractions

Dataset	BERT-Base-Cased	Electra-Small
TD	0.6754	0.6813
WNUM	**0.6781**	0.6809
WFS	0.6713	0.6743
WCON	0.671	**0.6829**

Table 4: F1 Score for different preprocessing techniques on dev set

The results show that removing numbers and expanding contractions both had contrasting effects on the two models. This shows that we could have yielded better results by trying different preprocessing techniques for the different transformer models. Apart from that, we see that the most positive effect on model performance came from removing full stops from the training data in both cases.

We also wanted to see the effect of our postprocessing step. For that, we compared the performance of the BERT-Base-Cased model on the Div-B dev split. As expected, the results showed minor improvement due to our postprocessing as the score increased from 0.6748 to 0.6754.

7 Error Analysis

The results we have obtained have brought to light some problems that need to be resolved. First of all, the data annotations have many issues, leading to a lower F1 score even though the predicted toxic spans are more appropriate in many cases. We have included some examples in Appendix D. In some cases, the annotations are not uniform in what toxicity label they assign to the same word over different text samples. We have also observed that complete sentences were marked as toxic just because of the presence of a few toxic words in them. These irregularities in the annotations make it difficult for the model to generalize on the data.

Besides the incorrect annotations, we further try to analyze the type of mistakes our system is making. The dataset contains numerous examples where no toxic spans are annotated. Such a case arose when the annotators had difficulty in attributing toxicity to a particular span. Investigating our model performance shows that our model highly under-performs on such examples. Table 5 depicts the drastic difference in the performance of the system over empty span examples (E.S) and non-empty span examples (N.E.S). Upon closely following E.S examples, we discovered that annotations of such examples carry more subjectivity than the others. In such cases, our model usually labels the word with the most negative sentiment as toxic and thus performs poorly.

In addition to the empty span examples, we also discover that our model fails to capture the full context in some cases. For e.g., in the phrase “no more Chinese,” our model only predicts the word Chinese as toxic, whereas the complete phrase attributes to the toxicity of the sentence. Another problem is our model’s inconsistency in labeling the corresponding noun and adjective pairs in a sentence. However, similar types of inconsistencies were also found in the annotations and are therefore difficult to avoid [Appendix D].

8 Conclusion

The task of detecting toxic spans in the text is a novel one, and there is no doubt about how important a model trained successfully for this task can turn out to be for online content moderation. However, the data gathered from online platforms tend to be noisy and corrupted. Coupled with the limitations of generating large-scale annotated datasets in real life, they pose two daunting challenges. In conclusion, our final submission shows that transfer learning through pre-trained transformer models can achieve competitive results for this task. Using modified loss functions and semi-supervised learning, even more can be extracted from limited annotated data. Moreover, considering the subjectivity involved in span detection, the task can also be expanded to report severity scores of spans and classify the type of toxicity. This will further help simplify and rationalize online content moderation.

References

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. *arXiv preprint arXiv:1704.02853*.

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.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. Applicaai at semeval-2020 task 11: On roberta-crf, span cls and whether self-training helps them. *arXiv preprint arXiv:2005.07934*.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.

Karishma Laud, Jagriti Singh, Randeep Kumar Sahu, and Ashutosh Modi. 2020. problemconquero at semeval-2020 task 12: Transformer and soft label-based approaches. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2123–2132.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. Dice loss for data-imbalanced nlp tasks.

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*, pages 87–91.

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 359–367, Portland, Oregon, USA. Association for Computational Linguistics.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. Hatexplain: A benchmark dataset for explainable hate speech detection. *arXiv preprint arXiv:2012.10289*.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Sean Papay, Roman Klinger, and Sebastian Padó. 2020. Dissecting span identification tasks with performance prediction. *arXiv preprint arXiv:2010.02587*.

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.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Mujahed A Saif, Alexander N Medvedev, Maxim A Medvedev, and Todorka Atanasova. 2018. Classification of online toxic comments using the logistic regression and neural networks models. In *AIP* *conference proceedings*, volume 2048, page 060011. AIP Publishing LLC.

Erik F Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. *arXiv preprint cs/0009008*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Alessandro Seganti, Helena Sobol, Iryna Orlova, Hannam Kim, Jakub Staniszewski, Tymoteusz Krumholc, and Krystian Koziel. 2019. NLPR@SRPOL at SemEval-2019 task 6 and task 5: Linguistically enhanced deep learning offensive sentence classifier. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 712–721, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Rushdi Shams. 2014. Semi-supervised classification for natural language processing. *CoRR*, abs/1409.7612.

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

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. Xlnet: Generalized autoregressive pretraining for language understanding.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

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). *arXiv preprint arXiv:1903.08983*.

Appendix

A Dataset

We worked on two different splits of the data across different stages of competition. Table 6 represent the no. of examples in train, val and test across Div-A and Div-B split.

	Div-A	Div-B
Train	6351	7835
Dev	794	794
Test	794	2000
Total	7939	10629

Table 6: Distribution of examples across Div-A and Div-B split

Div-A is basically a 80:10:10 split of the training data released by the organisers whereas Div-B split uses the train and test set of Div-A along with competition trial data as its training set. Div-B uses the official test set as its test set while keeping the dev set same as that of Div-A.

B Model Training

In this section, we provide the hyperparameter values we used while training our final models to facilitate the replication of our results at a later time. The acronyms correspond to:

- **LR** : Learning Rate
- **ML** : Max Len
- **LC** : Data Lowercase
- **DL** : Dice Loss (Alpha, Gamma)

Hyperpara.	BERT-Base-Cased	Electra
LR	1E-5	3E-5
ML	500	500
LC	False	True
DL	0.7,0.25	0.7,0.25

Table 7: Hyperparameter Values for BERT-Base-Cased and Electra (Small and Base)

For baseline model and semi-supervised learning model, the cross-entropy loss function provided in PyTorch was used with default hyperparameters. For AdamW optimizer, we used weight decay rate of 0.01 for all model parameters apart from the bias, gamma and beta terms, for which we used 0 weight decay.

Hyperpara.	Distil-Base-Unc.	XLNet-Base
LR	1E-5	3E-5
ML	500	400
LC	True	False
DL	0.7,0.25	0.4,0.25

Table 8: Hyperparameter Values for Distil-Base-Unc. and XLNet-Base

C Results

C.1 Dice Loss Results

In this section, we report the results of our hyper-parameter tuning for training with Self Adjusting Dice Loss. The result reported are for BERT-Base-Cased model on the Div-A dataset.

Parameter Values	Dev F1 Score
Alpha-0, Gamma-1	0.665
Alpha-0, Gamma-2	0.657
Alpha-0.5 Gamma-1	0.652
Alpha-0.7, Gamma-1	0.671
Alpha-0.8, Gamma-1	0.668
Alpha-0.7, Gamma-0.5	0.6723
Alpha-0.7, Gamma-0.25	**0.6725**

Table 9: BERT-Base-Cased trained with Self Adjusting Dice Loss on Div-A train split

D Error Analysis

We manually crawled through a part of dataset and found several inadequate annotations as described through table 11. Apart from erroneous annotations, we also found some examples of inconsistency in the structure of annotations. Table 10 outlines a few such pairs where different strategies are used for annotating similar texts. These examples highlight the inconsistencies in labelling of corresponding noun-adjective pairs.

Text	Ground Truth
You are an idiot	['idiot']
You are truly witless	All tokens
F***ing moron	['F***ing']
Fucking moron needs...	['Fucking','moron']
Hilary is a liar...	All tokens
Trump is an idiot.	['idiot']

Table 10: Irregularity in annotation for similar text-samples.

Text	Ground Truth	Suggested
I have no doubt it was Photoshopped(by an idiot, for an idiot)	[’idiot’]	[’idiot’,’idiot’]
She’s a Marxist twit.	[’Marxist’]	[’Marxist’, ’twit’]
Move to Russia and be a fascist pig all you want.	All tokens	[’fascist’, ’pig’]
lame. really really lame and cheap shot. pathetic actually.	[’lame’, ’lame’, ’actually’]	[’lame’, ’lame’, ’pathetic’]
So ”f*** you all”, ”kiss my a–” and ...	[’f***’]	[’f***’,’a–’]
You are just a stick buthole!	[]	[’stick’,’buthole’]

Table 11: Shows the words labelled toxic as ground truth(from annotators) and suggested accurate annotations