IITK@Detox at SemEval-2021 Task 5:

Archit Bansal **Abhay*** **Samyak Jain** * **Ashutosh Modi**
Indian Institute of Technology Kanpur (IIT Kanpur)
{architb, kabhay, samyakj}@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 main aim of the task is to detect the spans that make a text toxic, on a new dataset created from the toxic posts of the Civil Comment dataset. We have discussed various approaches to solve the problem. We started from traditional Bi-LSTM models and later moved on to the pre-trained transformer models. We then improvised our model with Data Augmentation and Semi-Supervised techniques.For semi-supervised learning, we leveraged the unlabelled data from Civil Comment dataset. We also analysed over different models Our final solution consists of an ensemble of transformer models achieving a F1 score of **0.6932** on the dev dataset.

1 Introduction

The internet user base has grown to over 4 billion and this boom has brought with it the issue of ensuring a safe environment for communication on the internet. The sheer amount of data being generated nowadays means that manual content moderation is not possible and therefore the focus has shifted to 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, but have mostly focused on identifying whole comments or documents as either toxic or not. However in a semi-automated setting where the human moderators might have to deal with lengthy comments, a model generated unexplained toxicity score can be frustrating. In order to tackle this issue, the SemEval 2021 Task 5 : Toxic Spans Detection introduces the task of accurately identifying toxic spans in a post which can give the human moderators a lot more insight about what actually contributes to the toxicity of the text.

The various challenges in this task included: a) small size of 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) inconsistency in data annotations. We experiment with different classifiers which include word based Bi-LSTM models and different pre-trained transformer models. Also, to tackle the issue of small dataset we have tried out data augmentation techniques and semi-supervised learning. Apart from that, we have experimented with different loss functions to tackle data imbalance and tried out character based transformer models to make our approach robust to spelling errors in texts. The codes are publically available on Github[1] Our best performing model was obtained through semi-supervised learning and achieves a F1 score of 0.6932 on the dev set.

The rest of this paper is arranged as follows, section 2 formally introduces the problem statement, section 3 lists previous work that has been done in related to this task, section 4 contains a description of the dataset, section 5 introduces our proposed approaches in further details followed by our results and error analysis in sections 6 and 7 respectively. Finally, we concluded in section 8.

2 Problem Definition

The shared task under SemEval 2021 Task 5 : Toxic Spans Detection is to extract a list of toxic spans for each toxic text in the dataset. Here, by toxic spans the organizers are referring to a sequence of words that contribute to the text's toxicity. The systems

* Authors equally contributed to this work.

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

are expected to return a list of the character offsets of each character in the detected spans (following 0 indexing) for each text. Therefore, the problem is clearly a span detection task which can also be approached as a sequence labelling task.

The metric used for evaluating the systems is the character level F1 score averaged over each post.

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

$$P^t(A_i, G) = \frac{|S^t_{A_i} S^t_G|}{S^t_{A_i}} \; R^t(A_i, G) = \frac{|S^t_{A_i} S^t_G|}{S^t_G}$$

Here, for the F1 score F_1^t of text t, $S^t_{A_i}$ refers to the set of character offsets returned by system A_i and G^t refers to the ground truth character offsets.

3 Related Work

As our task involves detection of toxic spans in a text, we present the related work in two parts: (i) Offensive Language detection and (ii) Span Detection. Due to the abundance of literature in both these areas, we focus our attention to those studies which we deem as pertinent to the current work.

Offensive Language Detection: Different abusive and offense language identification problems have been explored in the literature ranging from aggression to cyber bullying, hate speech, toxic comments, and offensive language. (Davidson et al., 2017) reported hate speech detection results using word n-grams and sentiment lexicon. Recent contributions of offensive language detection comes from the SemEval-2019 Task 6 OffensEval. The task was based on the OLID dataset ((Zampieri et al., 2019a)) and featured three sub tasks - (i) Identification (ii) Categorization, and (iii) Target identification. (Zampieri et al., 2019b) concluded that most of top performing teams either used BERT (Liu et al., 2019) or an ensemble model to achieved SOTA results for the corresponding subtask.

Span detection: Span detection/identification tasks form a substantial part of applied NLP. It includes numerous tasks like named entity recognition NER (Nadeau and Sekine, 2007), chunking (Sang and Buchholz, 2000), keyphrase detection(Augenstein et al., 2017) , or quotation detection(Pareti, 2016). An abundance of model architectures have been implemented for these tasks including range of models like token classification models, probabilistic models, conditional random fields, recurrent neural networks and transfer learning techniques. (Papay et al., 2020) showed that presence of BERT component in the model is highest positive predictor for most of the span identification tasks since it is robust and largely independent of span or boundary distinctiveness effects.

4 Corpus/Data Description

For this dataset, the organizers have randomly exrtacted 10K comments from a pool of crowd-rated toxic comments derived originally from the Civil Comments Dataset (Borkan et al., 2019). Each comment was then annotated by three crowd raters who extracted toxic spans from the text and the character offsets of toxic spans extracted by the majority were retained. The inter-annotator agreement was also calculated for a trial run of 35 posts using five crowd-raters. A Cohen's kappa score of 0.61 was achieved which indicates moderate consistency in the annotations.

The training data provided by the organizers consisted of a total of 7939 texts which we have further split into training, validation and test sets for evaluation purposes using a 80:10:10 split. After analysing the training split, we found that each example on an average consists of about 37 words and 206 characters. The longest post in the dataset contains 200 words in comparison to shortest single word example. After word level tokenization, we also found that there were 3996 different tokens labelled as toxic out of which 93.5% of the tokens are labelled as toxic only once while the most frequently occurring toxic label was 'stupid' with 971 annotations. The high number of toxic tokens occurring just once confirmed that their were inconsistencies in the annotations due to which many non-toxic tokens were wrongly marked positive in some cases.

5 Proposed Approach

Since our task is to detect the toxic spans, we approached the problem as tokenwise classification. Thus all the techniques we used, are done understanding the problem as a sequence labelling task.

5.1 Preprocessing

Tokenization: We first used a word level tokenizer to tokenize our data as we required a method to map the final token level output of our models to their character offsets while still maintaining the original form of the words for performing sub-word tokenization later on. We used the TreebankWord Tokenizer from NLTK library which is a rule based

tokenizer that uses regex to tokenize text and also returns the offset span for each token.

Data Cleaning: After tokenizing the text, we proceeded to perform some data cleaning operations on these tokens such as removing emojis (as they are not treated as toxic spans in any of the data samples), expanding contractions and removing digits.

5.2 Data Augmentation

Data augmentation is often not used in natural language processing due to the difficulty of maintaining the language structure in augmented data samples. However, (Wei and Zou, 2019) has shown that some simple augmentation techniques can be used in NLP classification tasks to improve model performance especially with when dealing with smaller datasets. Our training dataset consists of only about 6000 texts and therefore we use two different data augmentation techniques proposed in the paper to improve our model performance.

1. **Token Masking (TM):** Randomly remove non-toxic tokens from a data sample with probability p.
2. **Token Swapping (TS):** Randomly choose two tokens with the same toxicity label and swap them. Swaps were done L/α times where L is the number of tokens in the data sample.

The hyperparameters p and α were chosen after tuning and values used for training the models reported in this paper are specified in the appendix.

5.3 Methodology

Representation Erasure

The first method that we tried out was using a model (Li et al., 2017) that can explain the output of a neural network at the word level. We decided to take the Bi-LSTM model as the base model which we trained upon a subset of data from the Civil Comments Dataset for the task of classifying each text as toxic/non-toxic. We then used this model to compute the importance of each word as the relative change of the log-likelihood of the correct label for a text when a particular word is erased.

$$I(d) = \frac{S(e,c) - S(e,c,-d)}{S(e,c)}$$

Here $I(d)$ is the importance score of word d, $S(e,c)$ is the log-likelihood of text e for class c and $S(e,c,-d)$ if the log-likelihood for text e for class c with word d erased. The words which had an importance score above a set threshold were included in the toxic span for that text.

Bi-LSTM Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) is a powerful extension of recurrent neural networks. Our second approach was based on using a Bi-LSTM model for sequence labelling task, i.e. we classified each token in the text as toxic/non-toxic by using a linear head on top of the Bi-LSTM layer.We tried out two different embedding, the pre-trained Glove embedding and embedding generated using BERT model. These sequences were then passed to a Bi-LSTM layer which returned a hidden state vector for each token in the sequence, which were then passed through a fully connected layer with sigmoid activation function to perform binary classification on each token. The models were trained using the Adam optimizers on the Binary Cross Entropy loss.

Transformer Models Pre-trained transformer models built using the transformer architecture (Vaswani et al., 2017) have been able to achieve SOTA performance for most NLP tasks in recent times. As they are trained on a large corpus of data, transfer learning using using these models can achieve good performance even for small datasets. As the various transformer models vary in their training task and tokenization schemes, there is no one fits all model that can achieve SOTA results on every task. We have therefore tried out finetuning on BERT, RoBerta and XLNet transformer models with different classifier heads on top of them. In each case, the data has to first be tokenized using the tokenizer that was built for each pre-trained model on their training dataset. After that, we approach the task as a sequence labelling task, wherein the transformer models are used to obtain a contextually rich embedding for each token which is then fed into a binary classication layer to perform token level classification. For the binary classifier head, we tried out two different configurations. The first one was a simple linear layer with softmax activation, this is the configuration used with each transformer model. On top of that, the BERT model was also finetuned with a conditional random field(CRF) layer as the classifier head, which is additionally able to take into account the predicted tokens for previous tokens while making the prediction, hence capturing short-term relations between token labels.

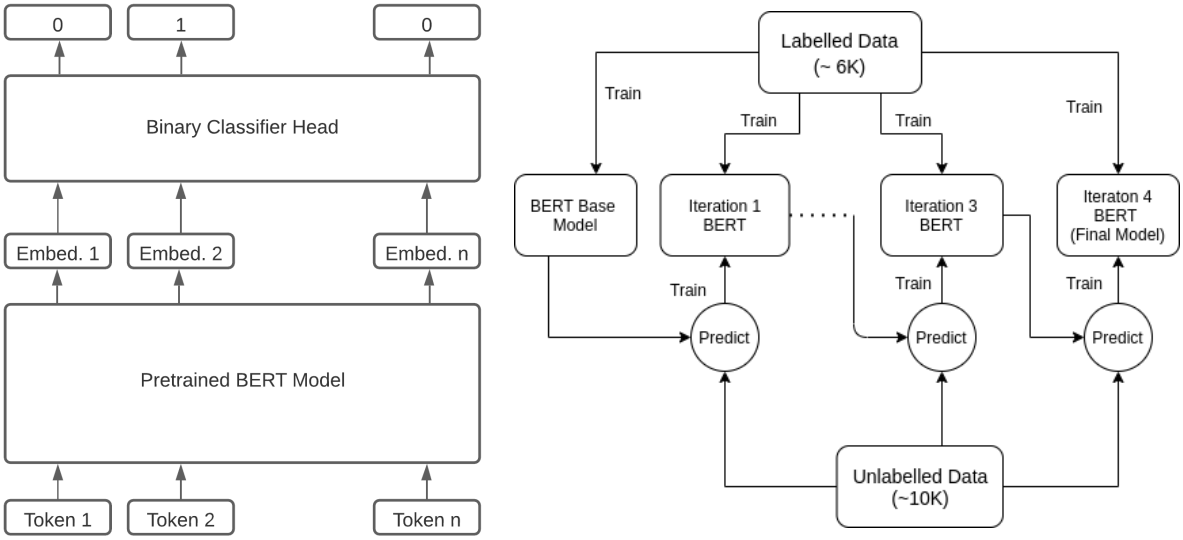

Figure 1: Model architectures for Transformer and Semi-Supervised models

Self Adjusting Dice Loss Imbalanced data is a major problem during training as it can mislead the model to focus only on the majority class in order to be able to achieve a lower loss. We could not tackle this issue with over/under sampling due to the mature of our problem and training on a weighted cross-entropy loss function did not improve the results we were getting. (Li et al., 2020) introduces a new objective function to train on when dealing with imbalanced datasets. The original dice coefficient is a F1 oriented statistic used to gauge the similarity of two sets. The paper proposed a modified dice loss which they reported to achieve a better F1 score than models trained on 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 (1 for positive class, 0 for negative class). Apart from that, the proposed loss has two hyperparameters which we tuned, α is used to reduce the weight of easy examples and γ is used to smoothen the optimization curve as well as assign a loss to negative examples.

CharBERT Most pre-trained language models construct word representations at subword level with Byte-Pair Encoding (BPE) or its variations. The problem with these methods is that they split a word into subword units and make the representation incomplete and fragile. To tackle the problem, (Ma et al., 2020) proposed a character-aware pre-trained language model named CharBERT. (Ma et al., 2020) showed in their results that CharBERT can improve both the performance and robustness of pre-trained models. We tried out both variations, CharBERT as well as CharRoBERTa for our task.

Semi-Supervised Learning The Civil Comments Dataset from which our training data was extracted consists of over 1 million comments, however due to the constraints of annotation, the training set provided had only 6000 data samples. (Shams, 2014) concluded that for text classification tasks, unlabelled data from a suitable data source can be used train semi-supervised models that achieve better results than model trained by supervised learning. Hence, we extracted 40000 toxic samples from the Civil Comments Dataset which were labelled with a toxocity score of 0.7 or higher and used them to perform 4 iterations of semi-supervised model training. As shown in Fig. 1, each iteration consists of predicting pseudo-labels for 10000 samples from the unlabelled data using the model trained in the previous iteration and then using these pseudo-labels with the true annotations of the training dataset to train the new model. The model used in training is the BERT-Base-Cased model with linear classifier head and for the first iteration, our original BERT-Base-Cased model trained on the 6000 training samples is used to predict the pseudo labels.

6 Experiments and Results

The metric used by the competition to evaluate our models is the f1 score calculated from the character offsets of toxic spans for each text and averaged over all the data points. However, while training the models, we used the token level f1 score over the whole data to perform the initial comparisons between different hyperparameter configurations. We then compared the best model from each case on the competition metric. While marking the toxic character offsets, we have included all the characters between two consecutive toxic tokens as toxic.All the results in this section are reported on the competition metric unless stated otherwise.

Our first approach, Representation Erasure gave token wise F1 score of 0.0934 on the dev set. Next, we experimented on the token wise Bi-LSTM classifier with various configurations and this achieved a tokenwise F1 score of 0.413 on the dev set. Various experiments and corresponding results for the above approaches have been discussed in appendix.

For the transformer models, we fine tuned different pretrained models with classifier heads. For the BERT-Base-Cased model, while preprocessing the data we excluded the lowercasing and punctuation removal steps, while all the other models were trained on our initially preprocessed data.

Pretrained Model	Val F1 Score
BERT-Base-Uncased	0.655
BERT-Base-Cased	0.669
BERT-Large-Cased	0.659
BERT-Base-Cased-CRF	0.669
BERT-BiLSTM-CRF	0.6701
RoBerta-Base	0.663

Table 1: Results for simple BERT and RoBERTa

For experiments using dice loss, we train the BERT-Base-Cased model with linear classifier head with different loss function parameter values. The results for our hyperparameter tuning have been reported in Table 2 . The best preforming models was able to achieve a better F1 score than the original BERT model.

Next, we tried the character-aware-pretrained models, CharBERT and CharRoBERTa. Though CharRoBERTa gave better results over RoBERTa-Base, CharBERT does not improve over BERT-base as shown in Table 3.

Until now, all our above results were obtained from using the train split only without any augmented data. Next, we train our models using the data augmentation techniques as discussed in section 5. We tried data augmetation with BERT-Base-Cased and CharRoBERTa model. The results are summarised in Table 4. Further information about the specific hyperparameters used can be found in the appendix.

Parameter Values	Val F1 Score
Alpha-0, Gamma-1	0.665
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 2: BERT-Base-Cased trained on Dice Loss

Model	Val F1 Score
BERT-Base-Cased	0.669
CharBERT	0.655
RoBerta-Base	0.669
CharRoBERTa	**0.6704**

Table 3: Results for CharBERT & CharRoBERTa

Augmentation Technique	Val F1 Score
(BERT) Original Dataset	0.669
(BERT) Original+TM	0.6757
(BERT) Original+TS	0.6755
(BERT) Original+TM+TS	0.6787
(CharRoBERTa) Original	0.6704
(CharRoBERTa) Original+TM+TS	0.6758

Table 4: Results for data augmentation techniques

Moving on, we tried the semi-supervised learning approach. As described in previous section, we run it over multiple iteration, the results of those on the dev set are summarised in Table 5. It is clear that SSL with iteration 4 gives us the best result on the validation set.

Next we tried ensembling our best performing

Model	Val 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	**0.693**

Table 5: Results for Semi-Supervised learning model

models but they gave poorer results on the validation set. Finally,we compared the performance of the best performing models corresponding to each approach on the dev set. The final result on the dev split is summarised in Table 6

Model	Val F1
BERT-Base-Cased	0.669
BERT- Augmented	0.6787
BERT-Dice Loss	0.6725
CharRoBERTa-Aug	0.675
SSL Iteration-4	**0.693**

Table 6: Final Results: Span level F1 Score

Our best model achieved a F1 score of 0.6932 on the dev set which was considerably better than the baseline score of 0.6 reported by the organizers.

7 Error Analysis

The results we have obtained till now have brought to light some problems that we will have to resolve. First of all, the data annotations have a lot of issues in them which has led to a lower F1 score even though the predicted toxic spans are more appropriate. We have included some examples in the appendix. Also, the annotations are not uniform in what they mark as toxic spans, i.e. the toxicity label of the same word is not consistent over the text samples. We have also observed that complete sentences are 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.

We also analysed the variations in the predictions of our best models on the validation set. The semi-supervised model from 3rd and 4th iteration mostly had the same predictions, with later performing slightly well while predicting some of the adverse adverbs and adjectives. While comparing the SSL models with BERT-Augmented model, we found that the SSL models capture the context more effectively in few examples. For eq, in the phrase "clown mentality", only the word clown attributes to the toxicity. The augmented model labelled the word mentality also as toxic whereas the SSL model gave the correct prediction. Similar examples can be found for BERT-Dice model also. For eg, in the phrase "May he rest in pieces", Bert-Dice doesn't find any token toxic whereas SSL correctly captures the 'rest in pieces'.

Besides the incorrect annotations, we further tried to analyse the type of mistakes our best model is making. In some of the examples, our model fails to capture the complete context. For eg, in the phrase "no more Chinese", our model predicts the word Chinese only as toxic, whereas the complete phrase attributes to the toxicity of sentence. Also, our model fails to predict some of the uncommon and informal interjections, like 'Duh!'. Another problem is the inconsistency in labelling the corresponding noun and adjective pairs. However, similar type of inconsistencies are also found in the annotations and are therefore difficult to avoid.

8 Conclusion

The task of detecting toxic spans in text is a novel one and there is no doubt about how important successfully completing this task can turn out to be for online content moderation. However, the data gathered from online platforms tends to be noisy and corrupted, and that coupled with the limitations of generating large scale annotated datasets in real life pose two daunting challenges. As a conclusion, our final submission shows that tranfer learning through pre-trained transformer models can achieve competitive results for this task, and data augmentation and semi-supervised learning can be used to extract more from a limited annotated data setting. However, their are other techniques which are worth exploring to further tackle this task.

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.

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*.

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

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Understanding neural networks through representation erasure.

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.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Charbert: Character-aware pre-trained language model. *arXiv preprint arXiv:2011.01513*.

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*.

Silvia Pareti. 2016. Parc 3.0: A corpus of attribution relations. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3914–3920.

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.

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

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.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196.

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

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. *arXiv preprint arXiv:1902.09666*.

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). *arXiv preprint arXiv:1903.08983*.

Appendix

A Representation Erasure

The architecture of the Bi-LSTM model trained for this mothod is as follows. A tokenized text sequence was fed to an embedding layer which used pre-trained Glove embeddings, the sequence was then passed through a Bi-LSTM layer which mapped each text into a feature vector which was fed to a fully connected layer with sigmoid activation for binary classification. The token level results obtained for our various experiments with Bi-LSTM model layers and pretrained Glove embeddings are given in Table 7.

Pretrained Embedding	Bi-LSTM layers	Dropout Rate	Val F1 Score
glove	1	0.5	0.0915
glove	2	0.5	**0.0934**
glove*	2	0.5	0.0921
gtwitter	1	0.7	0.0926
gtwitter**	1	0.7	0.0918
gtwitter*	1	0.7	0.0919

Table 7: Here glove and gtwitter stands for glove.6B.300D and glove.twitter.27B.200D resp.
* - Trained on lowercase dataset
** - UNK token not initialized to zero

B BiLSTM results

We experimented on the token wise Bi-LSTM classifier with various configurations. We use Bi-LSTM with different word embeddings and variations in hyperparameters like no. of Bi-LSTM layers and hidden dimensions. The token wise results on validation split are summarised in Table 8.

Pretrained Embedding	Bi-LSTM Layers	Hidden Dimension	Val F1 Score
NA	2	64	0.25
glove	2	64	0.32
gtwitter	3	64	0.38
gtwitter	2	64	**0.413**
gtwitter	2	128	0.40
BERT	1	64	0.456

Table 8: **BiLSTM**: glove and gtwitter stands for glove.6B.300D and glove.twitter.27B.200D resp.

C Data Augmentation

For creating the augmented data, we had to tune two hyperparameters, for token masking the probability p with which the non-toxic tokens were masked was tuned while for token swapping, the parameter α which decided the number of swaps to make per text sample. The values of p and α used for the datasets used to report the results in Table 4 are as follows :

1. Orig.+TM : p was taken to be 20%.
2. Orig.+TS : α was token to be 5. Therefore, for a text sample with L tokens, L/5 swaps were made. Also, no augmented example was generated for samples with less than 5 tokens.
3. Orig.+TM+TS : p was randomly chosen between (10-20)% and α was randomly chosen between (10-20) for each text sample.

D Error Analysis

The inconsistency in annotations is described through table 9

Text	Ground Truth	Predicted
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']

Table 9: shows the words labelled toxic as ground truth(from annotators) and predicted by Bert-base cased model for few examples from Val. set