Toxic Span Detection

Student Name: LAM Chun Ting Jeff
Student no: 1222973
Email: ctjlam@connect.ust.hk

Student Name: CHAN Ho Yeung
Student No: 20277805
Email: hychanbe@connect.ust.hk

Student Name: WONG Ho Yin
Student No: 03465380
Email: hywongck@connect.ust.hk

1. Introduction

In discussion forums, toxic posts exist oftenly. Toxic posts mean posts that are rude, disrespectful, or unreasonable; and which can make users want to leave the conversation (Borkan et al., 2019a). In order to maintain a healthy discussion environment, reviewing posts that are toxic or not is an essential task. Current toxicity detection systems usually assist humans for classification, while reviewing all by humans is impossible. Therefore, fully automated toxicity detection systems are required. The major challenge of a fully automated system is that a post may include no toxic span and still be marked as toxic, and vice versa. The suggested task could be addressed as supervised sequence labeling, training on the provided posts with gold toxic spans, or treated as rationale extraction (Li et al., 2016; Ribeiro et al., 2016), using classifiers trained on larger external datasets of posts manually annotated as toxic or not, without toxic span annotations.

2. Importance

Governments around the world have stepped up monitoring of social media to mitigate the hate speech problem. A notable example is that Germany's Federal Office of Justice fined Facebook €2 millions in 2019 for violating the hate speech law. Non-compliance to such laws could result in substantial financial loss to operators of social media and discussion forums alike.

Operators of discussion forums largely depend on human moderators to control hate speech. For instance, Reddit, an American social discussion forum with a global presence, relies on volunteer moderators to monitor user behavior (What's a moderator (n.d.). Reddit). However, the human moderator approach has major drawbacks. Human moderators are often overwhelmed by massive content and are incapable of dealing with a variety of languages and topics. Even worse, human moderators become the target of doxxing and receive attacks and death threats for their work (Andrew R. Chow, 2022). The severity of the issue demands a more automated and scalable approach to detect toxic posts and eradicate hate speech in the first place.

3.Dataset

The dataset is retrieved from the ipavlopoulos toxic spans repository (Pavlopoulos et al., 2021). The dataset contains word sequences (span) in posts that have been considered toxic by voluntary annotators. It consists of "tsd_train.csv" which is a subset for training our model, and "tsd_test.csv" which is another subset for testing the performance of our model. Each subset consists of 2 columns, spans and text. The spans column contains a list of integers representing the index of the annotated span(s) in the post (that is the character level position of the span(s) in the text). It can be an empty list meaning that there is no span within the text. The text column simply contains the whole content of posts.

Taking the following post as an example "I only use the word haole when stupidity and arrogance is involved and not all the time. Excluding the POTUS of course.". It consists of two spans, "stupidity" and "arrogance". The character index of the spans "stupidity" and "arrogance" are 31-39 and 45-53 respectively. Therefore, the span list will be [31, 32, 33, 34, 35, 36, 37, 38, 39, 45, 46, 47, 48, 49, 50, 51, 52, 53].

```
print("spans:", train.iat[7926,0])
print("text:", train.iat[7926,1])
```

```
spans: [31, 32, 33, 34, 35, 36, 37, 38, 39, 45, 46, 47, 48, 49, 50, 51, 52, 53]
text: I only use the word haole when stupidity and arrogance is involved and no
t all the time.  Excluding the POTUS of course.
```

Figure 1. The 7927th record in tsd_train.csv

In 'tsd_train.csv', there are a total of 7,939 post records, in which 485 texts do not contain toxic spans and 7,454 texts contain toxic spans. In terms of characters, there are a total of 1,629,637 in this subset, in which 139,600 are span characters and 1,490,037 are non-span characters.

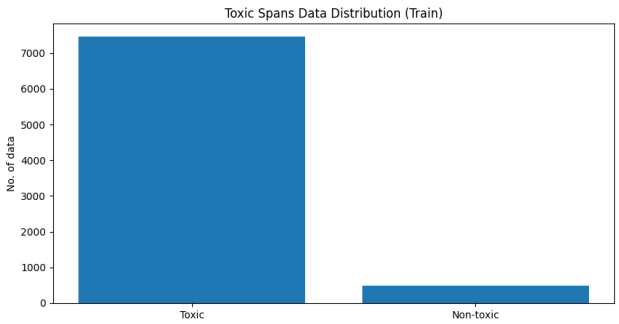

Figure 2. Training set: Distribution of toxic and non-toxic data in unit of text

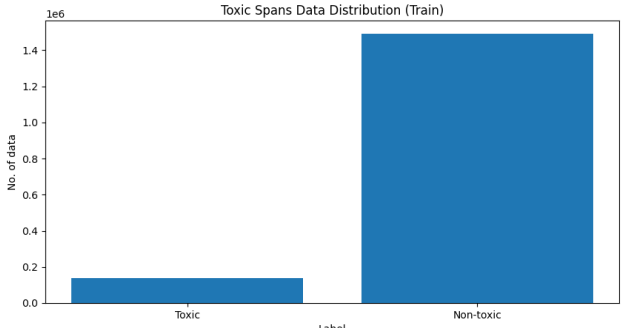

Figure 3. Training set: Distribution of toxic and non-toxic data in unit of character

In 'tsd_test.csv', there are a total of 2,000 post records, in which 394 texts do not contain toxic spans and 1,606 texts contain toxic spans. In terms of characters, there are a total of 374,007 in this subset, in which 14,983 are span characters and 359,024 are non-span characters.

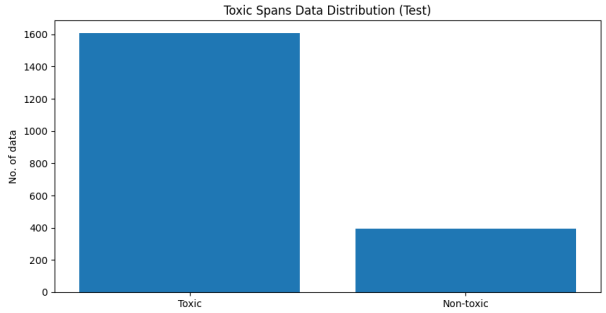

Figure 4. Testing set: Distribution of toxic and non-toxic data in unit of text

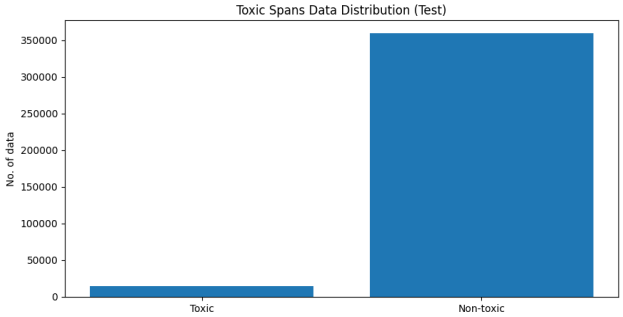

Figure 5. Testing set: Distribution of toxic and non-toxic data in unit of character

It is worth noting that the number of toxic posts (7,454) is over 14 times more than non-toxic posts (485) in the training set. The ratio is over 3 times in the testing set (1,606 vs 394). The reason is that this dataset is a further refinement on about 30,000 posts that had already been classified as toxic by a majority of voluntary annotators (Finley, 2016).

4. Methodology

The toxic span detection task is divided into five stages. First, we retrieve the training dataset and test dataset from the csv files. Second, we perform data augmentation (see section 4.1). Third, preprocess the data, mainly perform tokenization (see section 4.2). Fourth, train the model and finally, evaluate the model with the 'tsd_test.csv' (see section 5).

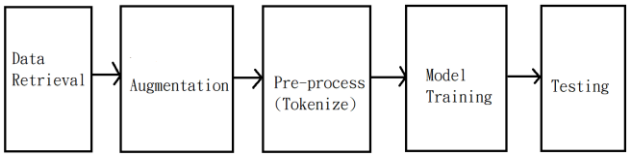

Figure 6. The solution flow of toxic spans detection

4.1. Data Augmentation

Deep neural networks usually require large quantities of training data for model training. Data augmentation can help to increase both the amount and diversity of training data by randomly augmenting the available data. Data augmentation is used to teach a model about invariances (e.g. translation, horizontal flipping) in the data domain, which is easier than hard coding invariances into the model architecture directly, i.e. allow the model to learn the features from the (augmented) data rather than manually introduce additional symmetries, such as rotation equivariance, into neural networks (D. Ho et al., 2019, Benton G et al., 2020).

It is noticed that most of the pioneers of this span detection task adopted data augmentation techniques. Our team adopts a similar approach but we perform the data augmentation on non-span text data or span text data. The rationale behind is the unbalanced proportion of span and non-span texts and characters in the training set. As mentioned before, there are 7,939 post records in the training set but only 495 are non-span in terms of text. However, the proportion of span and non-span data is the exact opposite if it is measured in terms of characters. Therefore, it is uncertain whether balancing the text-level disproportion or character-level disproportion will produce a more effective training. In view of the unbalanced training data, we experimented with data augmentation on either non-span or span text data.

Since manual data augmentation is very time consuming and error-prone, an automatic data augmentation tool for natural language processing "nlpaug" (Edward Ma, 2019) is used in this project. The tool offers various augmentation rules at character level, word level and sentence level to augment training data for natural language processing tasks.

Specifically, we augmented either non-span text data or span text data in each attempt. For the data augmentation on non-span text data, Contextual Word Embeddings Augmenter is used on the whole non-toxic text, with insert or substitute operations using similar words in the BERT model.

The following is the example of insert and substitute operations of Contextual Word Embeddings Augmenter.
Original:
The quick brown fox jumps over the lazy dog
Augmented Text (insert):
even the quick brown fox usually jumps over the lazy dog
Augmented Text (substitute):
little quick brown fox jumps over the lazy dog

For the data augmentation on span text data, Keyboard Augmenter is used on span text data except the span characters. It simulates typo error by random value with leveraged keyboard distance. We also ran a control experiment by simply duplicating the toxic posts to verify if the data augmentation tool could improve the performance of Keyboard Augmenter.

The following is the example of Keyboard Augmenter.
Original:
The quick brown fox jumps over the lazy dog.
Augmented
Text:The quick beIwn fox jKmpQ ive3 the lazy dog.

4.2. Data Preprocessing

There are two tasks in this stage. The first task is annotating the word tokens with toxicity labels, and the second task is predicting the toxic spans in a text.

We will first preprocess the data by tokenizing the text. As we have a text-span structure, after tokenizing the text field, we will map the token with the span index, and to produce a token-spans structure, using a mark of 0,1 to indicate if that token is span or not. For the structure, word-level BIO tags were used, which the tokens were labeled as B (beginning word of a toxic span), I (inside word of a toxic span), or O (outside of any toxic span), and become three columns of [0,0,1], [0,1,0], and [1,0,0] as the input for NER model.

The tokenizer we are using is Roberta Tokenizer. Roberta Tokenizer is a byte level byte-pair encoding tokenizer, derived from the GPT-2 tokenizer. This tokenizer has been trained to treat spaces like parts of the tokens, so the words will be encoded differently whether there is empty space at the beginning of the sentence or not.

The main reason that we are using this model is because it outperforms the BERT model on GLUE, RACE and SQuAD scores (Liu et al., 2019), by fine-tuning the RoBERTa model with the Kaggle toxicity datasets.

4.3. Model

The first part of the model is NER (Named entity recognition) layers, which are used to annotate part of the sentences that are toxic using the training dataset, producing a set of labels for the input tokens.

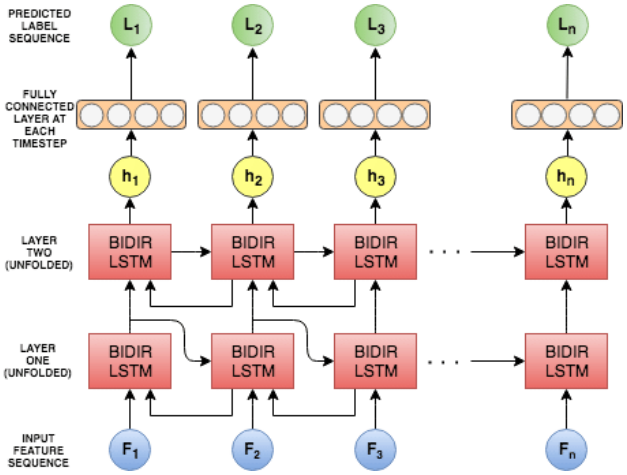

Figure 7. Illustration of NER Layers

On top of the NER layers are the pre-trained RoBERTa-base model followed by LSTM (Long short-term memory) and CRF (Conditional random field) layers. The tokenizer is extracted from the RoBERTa-base model for creating NER outputs, while the LSTM and CRF layers take the context of a token into account for the prediction. The prediction result is a set of toxicity labels, which are compared with the ground truth.

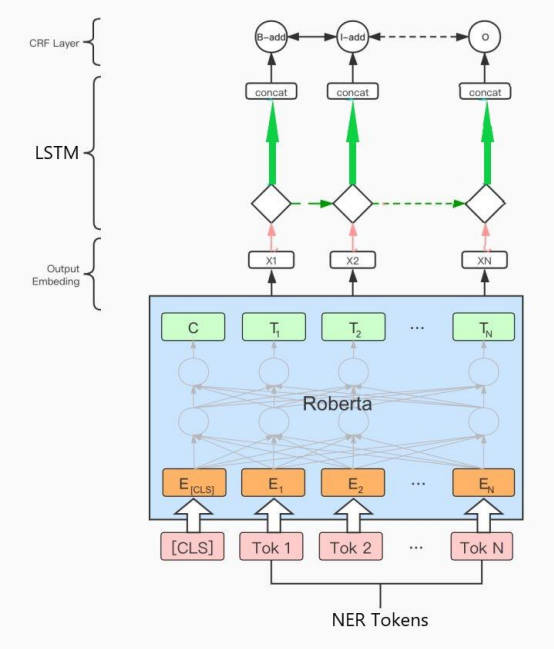

Figure 8. Illustration of Robeta+LSTM+CRF Layers

5. Result and Discussion

F1 score is used to check if our model can produce an exact result with the ground truth. A similar method developed by Nguyen et al.(2021) based on a RoBERTa model attained a benchmark 69.89%. And the following is the result that we made.

Training and Testing Result without augmentation. The model is trained without any data augmentation.

```
Epoch 5/5
397/397 [==============================] - ETA: 0s - loss: 6.6681 - accuracy: 0.2231WARNING:
INFO:tensorflow:Assets written to: /content/drive/My Drive/MSBD5018/trial04/assets
INFO:tensorflow:Assets written to: /content/drive/My Drive/MSBD5018/trial04/assets
WARNING:absl:<keras.layers.recurrent.LSTMCell object at 0x7f3019708d0> has the same name 'L
397/397 [==============================] - 275s 693ms/step - loss: 6.6681 - accuracy: 0.2231
train F1 = 0.720994
test F1 = 0.658357
train F1 = 0.720994
test F1 = 0.668241
```

Figure 9. Screenshot of training without augmentation

Training and Testing Result with augmentation on non-toxic posts The model is trained with non-toxic posts by Contextual Word Embeddings Augmenter .

```
Epoch 5/5
422/422 [==============================] - ETA: 0s - loss: 5.9774 - accuracy: 0.2301WARNING:
INFO:tensorflow:Assets written to: /content/drive/My Drive/MSBD5018/trial05/assets
INFO:tensorflow:Assets written to: /content/drive/My Drive/MSBD5018/trial05/assets
WARNING:absl:<keras.layers.recurrent.LSTMCell object at 0x7f9d9817e150> has the same name 'L
422/422 [==============================] - 383s 908ms/step - loss: 5.9774 - accuracy: 0.2301
train F1 = 0.716615
test F1 = 0.635070
train F1 = 0.716615
test F1 = 0.668170
```

Figure 10. Screenshot of training with non-toxic post augmentation by Contextual Word Embeddings Augmenter.

Training and Testing Result with keyboard augmentation on toxic post

```
Epoch 5/5
770/770 [==============================] - ETA: 0s - loss: 3.4684 - accuracy: 0.2658WARNING:a
INFO:tensorflow:Assets written to: /content/drive/My Drive/MSBD5018/trial03/assets
INFO:tensorflow:Assets written to: /content/drive/My Drive/MSBD5018/trial03/assets
WARNING:absl:<keras.layers.recurrent.LSTMCell object at 0x7f0019175d90> has the same name 'LS
770/770 [==============================] - 663s 861ms/step - loss: 3.4684 - accuracy: 0.2658
train F1 = 0.796306
test F1 = 0.722375
train F1 = 0.796306
test F1 = 0.655095
```

Figure 11. Screenshot of training with toxic post augmentation by Keyboard Augmenter.

Training and Testing Result with augmentation by simple doubling toxic post

```
Epoch 5/5
770/770 [==============================] - ETA: 0s - loss: 3.6870 - accuracy: 0.2287WARNING:a
INFO:tensorflow:Assets written to: /content/drive/My Drive/MSBD5018/trial02/assets
INFO:tensorflow:Assets written to: /content/drive/My Drive/MSBD5018/trial02/assets
WARNING:absl:<keras.layers.recurrent.LSTMCell object at 0x7f82de0df810> has the same name 'LS
770/770 [==============================] - 662s 860ms/step - loss: 3.6870 - accuracy: 0.2287
train F1 = 0.817758
test F1 = 0.751276
train F1 = 0.817758
test F1 = 0.668455
```

Figure 12. Screenshot of training with toxic post augmentation by doubling toxic posts.

Rank	Method	Training F1	Testing F1
1	Baseline	-	0.6989
2	Simple Augment with toxic post	0.817758	0.668455
3	No augmentation	0.720994	0.668241
4	Augment with non-toxic post	0.716615	0.668170
5	Keyboard Augment with toxic post	0.796306	0.655095

Table 1. Summary of training and testing result

From Table 1 above, we notice that simple data augmentation with toxic posts yields the best result among the four attempts. It is believed that because of the preprocessing, characters or word based takes a much more important role than posts when it comes to model training.

It is also noticed that keyboard augment with toxic posts resulted in the lowest F1 score. It is believed that the keyboard augmenter created a lot of words with typing mistakes, which may introduce noises (incorrect character sequence of word tokens) in the model training and worsen the performance.

In all the attempts, training F1 score is higher than testing F1, it may come to overfit with non-toxic posts as there are more non-toxic words than toxic words. Besides, span words learnt in training dataset may not be applicable to test dataset. For instance, the word "troll" is considered a span in the 3717th training record but the same word "troll" in the 44th testing record is not a span.

6. Conclusion

Toxic span text badly affects people's motivation in discussion in forums. Although manual moderation is a way to solve the issue, human moderators are overwhelmed with massive content and are incapable of dealing with a variety of languages and topics. Therefore, an automated and scalable approach is preferred to solve the issue.

Due to the imbalance of the dataset, data augmentation is used to increase both the amount and diversity of training data. As there are differences in imbalance data in terms of text level and character level, two approaches are tested out. Contextual Word Embeddings Augmenter for augmenting non-toxic text, and Keyboard Augmenter or Simple Doubling for augmenting toxic text are applied individually for different results. Therefore, with the same RoBERTa-base model, there are different results because of different data augmentation methods.

Trained with 4 different kinds of methods, which are trained without augmentation, augmented with non-toxic post, simple Augment by doubling with toxic post, and keyboard augment with toxic post, we notice the result of keyboard augment with toxic post outperformed the others methods. It is believed that with the preprocessing, characters or word based takes a much more important role than post when it comes to model training.

7. References

[1] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019a. Nuanced metrics for measuring unintended bias with real data for text classification. In WWW, pages 491–500, San Francisco, USA.

[2] Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. arXiv preprint arXiv:1612.08220.

[3] Marco T. Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?" Explaining the predictions of any classifier. In SIGKDD, pages 1135–1144, San Francisco, USA.

[4] Reddit. (n.d.). What's a moderator? Reddit Help. Retrieved March 30, 2022, from https://reddit.zendesk.com/hc/en-us/articles/204533859-What-s-a-moderator-

[5] Andrew R. Chow, (2022). Reddit Allows Hate Speech to Flourish in Its Global Forums, Moderators Say. Retrieved March 30, 2022 from https://time.com/6121915/reddit-international-hate-speech/

[6] Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. 2019. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. In Proceedings of the 36th International Conference on Machine Learning, 2731–2741, California, USA.

[7] Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. 2020. Learning Invariances in Neural Networks. In the 34th Conference on Neural Information Processing Systems, Canada.

[8] Yinhan Liu, Myle Ott, Naman Goyal, et al. 2019. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". Retrieved April 3, 2022, from https://arxiv.org/pdf/1907.11692v1.pdf

[9] Ipavlopoulos. (n.d.). Toxic_spans/SEMEVAL2021/data at master · ipavlopoulos/toxic_spans. Retrieved May 2, 2022, from https://github.com/ipavlopoulos/toxic_spans/tree/master/SemEval2021/data

[10] Kline Finley. 2016. Want to save the comments from trolls? do it yourself. Retrieved May 4, 2022, from https://www.wired.com/2016/03/want-save-comments-trolls/

[11] Makcedward. (2021). Nlpaug/textual_augmenter.ipynb at master · Makcedward/NLPAUG. Retrieved May 2, 2022, from https://github.com/makcedward/nlpaug/blob/master/example/textual_augmenter.ipynb