Identifying Toxic Parts of a Message

Hexin Zhang
hexinz@umich.edu

1 Introduction

In the age of Internet, one of the primary concerns is to maintain healthy online discussions. An online toxicity (abusive language) detection system is in bad need. Although various toxicity detection models have been proposed, however, they only managed to classify messages as toxic or give a toxicity score to the message, which is not quite useful when people have to modify messages in order to pass the toxicity test. Hence, I build an end-to-end model that can accurately identify specific toxic characters, words or phrases in raw messages. What I managed to do is to take in raw messages like ”This is a stupid example, so thank you for nothing a!@#!@.” and return toxic character location [10,11,12,13,14,15,51,52,53,54,55,56], which can be further visualized as ”This is a stupid example, so thank you for nothing a!@#!@.” I get an average F1 score of 0.6134, which I believe can be further improved when more data are given. Highlighting toxic spans instead of simply giving toxicity score can assist people, especially news portals moderators, to tailor only part of their messages, which is more intuitive and can save a lot of time and labor. In this work, I have tried out three methods, Hidden Markov Model (HMM), Fine-tuned RoBERTa Named Entity Recognition (NER) model, and Fine-tuned RoBERTa classification model with LIME model. Among all three methods, Fine-tuned RoBERTa Named Entity Recognition (NER) model has the best performance. Therefore, we can learn from the result that the toxicity detection task can be most effectively solved by fine-tune a pre-trained NER model.

2 Data

The data set the project use is generated from Civil-Comments Data set. In the above data set, only seven labels are included: toxicity, severe_toxicity, obscene, threat, insult, identity_attack, and sexual_explicit. The toxic spans are not included. Raters then need to identify the toxic spans in the messages manually. Therefore, SemEval-2021 organizers assigned a toxicity score to each character offset by calculating the fraction of raters who annotated that character offset as toxic. The ground truth label is provided by only retaining those character offsets with scores over 50%. The training, testing and trial data set are available on https://github.com/ipavlopoulos/toxic_spans/tree/master/data. The total size of the data set is 3M.

To be more specific, the training dataset contains two columns: ”spans” and ”text”. The text are raw messages with ”\n” between sentences and rare characters like ”a!@#!@”. For the class distribution, we preprocess the data to assign two kinds of labels: sentence labels for classification LIME model and another tokens labels for NER model. Take training data set as an example. For the former one, we consider messages containing toxic parts as 1 and others as 0, then there are 7949 rows in total, 7454 1s and 485 0s. For the latter, we consider the toxicity of tokens (either word level or character level). There are overall 334021 tokens, 306661 are non-toxic and 27360 are toxic. Detailed information of two kinds of labels are listed below.

Table 1: Data analysis for two kinds of labels

	sentence labels	tokens labels
count	7949	334021
mean	0.94	0.08
std	0.24	0.27
min	0.00	0.00
25%	1.00	0.00
50%	1.00	0.00
75%	1.00	0.00
max	1.00	1.00

Also, the distribution of number of tokens in each message is shown in Fig. 1;

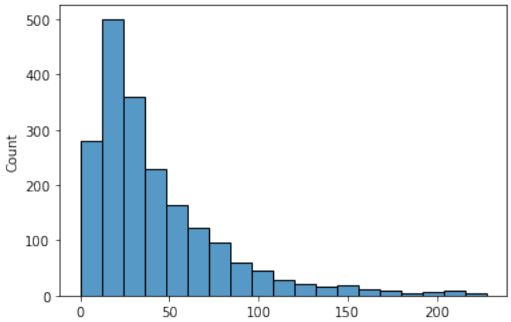


Figure 1: Distribution of RoBERTa Tokens

3 Related Works

(Li et al., 2017) and (Katsiolis, 2020) proposed the Input Erasure method, which erases parts of the message and calculates the resulting toxicity reduction. Then, threshold is decided manually to select the toxic spans that result in larger toxicity reduction.

(Ribeiro et al., 2016) and (Katsiolis, 2020) proposed the Interpretable Model-Agnostic Explanations (LIME) algorithm, which explains the model predictions in the form of words. They first classify the messages use whatever classifier (Random Forest for example) as toxic or not and give a list of features (in the form of words) that have most impact on the decision. A value will be assigned to each feature indicating the toxicity. Similarly, the threshold is decided manually to select the toxic spans.

(Katsiolis, 2020) proposed a probabilistic-baed Sequence Labeling model that assign each token 1 or 0 based on whether its character offset is in the ground truth. The tokens may be characters or words.

(Devlin et al., 2018) proposed a pre-trained model called 'bert'. BERT is a multi-layer Transformer. It has two versions, the "Base" one and the "Large" one. It is widely used in Masked Language Model (MLM) and Next Sentence Prediction (NSP) tasks. What's more, it has a very smart in-built tokenizer, which can not only consider words but also group of characters denoted as "ing". In this problem, based on the pre-trained BERT model, (Katsiolis, 2020) proposed a BERT-LIME model that employee bert classifier combined with LIME to give toxic spans. The pre-trained BERT model is available on github: https://github.com/google-research/bert.git

(Hochreiter and Schmidhuber, 1997) proposed a famous LSTM model and (Katsiolis, 2020) use Bi-LSTM binary classifier combined with LIME to give toxic spans.

Basically speaking, the above methods can be classified into three categories: Input Erasure method, probabilistic-based method, and classifiers combined with LIME method. The problem for Input Erasure method and LIME method is that they usually nee manual decision of threshold value. For sequence labeling, it usually does not have good generic performance and is unsuitable for large data set due to longer data processing time.

I decide to try out RoBERTa Named Entity Recognition (NER) model. It is also a pre-trained and thus is quite easy to fine-tune and achieve a better performance. Most importantly, it does not need a manually decided threshold value and is very easy to train with large amount of data. I also tried RoBERTa-LIME model as well as Hidden Markov Model (HMM) to compare the performance.

4 Proposed Method

As mentioned before, three kinds of model are implemented to make comparisons: RoBERTa Named Entity Recognition (NER) model, Hidden Markov Model (HMM) as well as RoBERTa-LIME model.

4.1 Fine-tuned RoBERTa NER model

For data preprocessing, I use the in-built RoBERTa tokenizer and assign each token a label based on the ground truth value. The input data set is shown below.

Table 2: Input data format for RoBERTa NER model

sentence_id	words	labels
0	another	nt
0	Ġviolent	t
0	Ġand	t
0	Ġaggressive	t
0	Ġkilling	nt
0	Ġa	nt
...	...	...
1	i	nt
1	Ġam	nt
1	Ġ56	nt
1	Ġyears	nt
1	Ġold	nt

where ”nt” means non-toxic while ”t” means toxic. Character ’Ġ’ is for the space, so that we did not lose the information of location of space after preprocess the data. To take tokens sequence into consideration, we need to assign tokens from the same sentence a same label in the ”sentence_id” column.

For the RoBERTa NER model, I use the ”simpletransformer” python package (https://simpletransformers.ai/docs/ner-model/). NER model can also be interpreted as token classification model, which can make classification on the token level. RoBERTa is short for robustly Bert Model. Ideally. it can have better performance than BERT. I train the model for 10 epochs, using the ”AdamW” optimizer with learning rate of 4e-5. To fine-tune the model, I evaluate the model each epoch on dev data set, then choose the model based on the performance on dev data set.

4.2 Hidden Markov Model

HMM is usually used for POS task. I implement the HMM by myself. I first tokenize sentences and do the frequency counts of $C(w_i, t_i)$, $C(t_i)$, and $C(t_{i-1}, t_i)$ where w is the word and t is the tag. Then, use the frequency counts to calculate transition probability:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

and Emission Probability:

$$P(w_i|t_i) = \frac{C(w_i, t_i)}{C(t_i)}$$

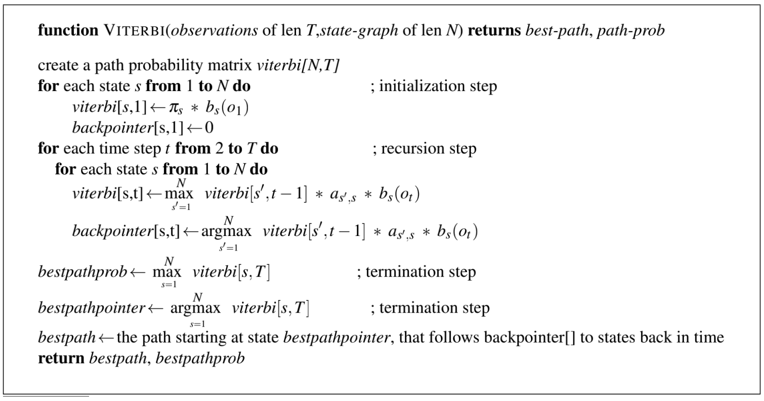
After that, I calculate the maximum-likelihood estimates with independency assumed:

$$P = P(t_i) \prod_{i=2}^{m} P(t_i|t_{i-1}) \prod_{i=1}^{m} P(w_i|t_i)$$

Furthermore, to speed up the tagging process of HMM, I also choose to implement a dynamic programming algorithm named ”viterbi”. The pseudo code is shown in Fig. 2.

4.3 Fine-tuned RoBERTa classification model with LIME

For the RoBERTa classification model, I use the ”simpletransformer” python package (https://simpletransformers.ai/docs/classification-models/). The input data has two columns: text and labels. Some samples are shown in Table 3.



```
function VITERBI(observations of len T,state-graph of len N) returns best-path, path-prob

create a path probability matrix viterbi[N,T]
for each state s from 1 to N do                        ; initialization step
      viterbi[s,1] ← $\pi_s * b_s(o_1)$
      backpointer[s,1] ← 0
for each time step t from 2 to T do                    ; recursion step
   for each state s from 1 to N do
      viterbi[s,t] ← $\max_{s'=1}^{N}$ viterbi[$s', t-1$] $* a_{s',s} * b_s(o_t)$
      backpointer[s,t] ← $\operatorname{argmax}_{s'=1}^{N}$ viterbi[$s', t-1$] $* a_{s',s} * b_s(o_t)$
bestpathprob ← $\max_{s=1}^{N}$ viterbi[$s, T$]                 ; termination step
bestpathpointer ← $\operatorname{argmax}_{s=1}^{N}$ viterbi[$s, T$]           ; termination step
bestpath ← the path starting at state bestpathpointer, that follows backpointer[] to states back in time
return bestpath, bestpathprob
```

Figure 2: Viterbi pseudo code

Table 3: Input data for RoBERTa classification model

text	lbls
A fool and our money are soon to be parted.	1
Ha ha, HILLARY LOST.	1
...	...
I always knew you were a dirty cop.	0

Similarly, I trained for 5 epoch, using the ”AdamW” optimizer with learning rate of 4e-5. To fine-tune the model, I evaluate and choose the model based on the performance on dev data set.

For the LIME part, I use the ”lime” python package. Take in to-predict sentences, predict_proba method generated by the classification, and then the model will output a list of features (tokens) with a value that indicates the contribution of word to the predicted class.

5 Evaluation and results

5.1 Evaluation method

F1 score will be employed to evaluate the performance of the model.

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * P * R}{P + R}$$

Where P is further defined as the number of elements in the intersection between prediction and ground truth over the number of elements in prediction. R is defined as the number of elements in the intersection between prediction and ground truth over the number of elements in ground truth. Moreover, F1 is zero if ground truth is empty but prediction is not, and vice versa.

$$P = \frac{|S_a \cap S_t|}{|S_a|}, R = \frac{|S_a \cap S_t|}{|S_t|}$$

$$F1 = \frac{2 * P * R}{P + R}$$

Where S_a is the list of locations of characters that a method detects as toxic span in a comment and S_t is the list of locations of characters of the ground truth annotated toxic span.

Then, the final average F1 score can be simply calculated by average each F1 score in above formula for each message. In terms of formula, it should be simply:

$$\frac{1}{m} \sum_{i=1}^{m} F1$$

5.2 Result

5.2.1 Quantitative result

The baseline algorithm for the model is Random Forest Classifier in Sklearn python library, taking in data format shown in Table 3. If the message is classified as toxic, then locations of all characters are returned; if the message is classified as non-toxic, then an empty list will be returned. The F1 score for classification of messages is 0.8907 while the average F1 score for predicting toxicity token use the above evaluation plan is 0.1384, as shown in Table 5.

For Fine-tuned RoBERTa NER model, the precision, recall and f1_score on dev data set are shown in Fig. 3.

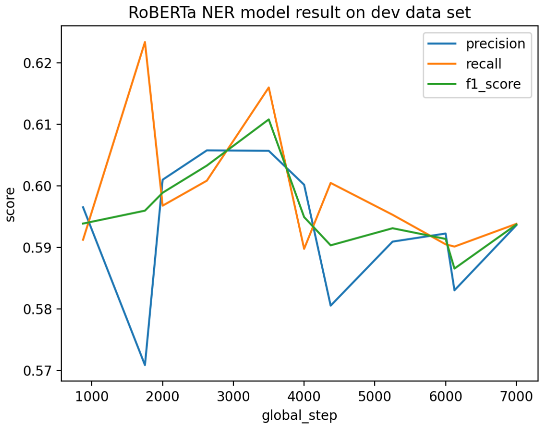


Figure 3: RoBERTa NER model score with respect to global steps

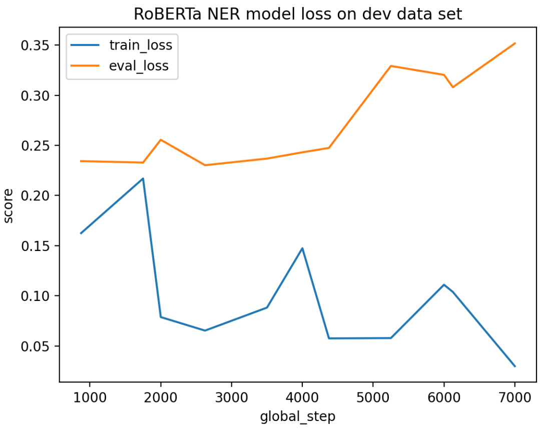


Figure 4: RoBERTa NER model score with respect to global steps

The training loss and evaluation loss are shown in Fig. 4. From the ten models of RoBERTa NER model, I select the best model, which then results in an average F1 score of test data set as high as 0.6134, which is a great improvement of previous baseline method, and is also larger than HMM testing average F1 score. The result is listed in Table 5.

I also implement the HMM to make comparison. For HMM, the resulting scores are shown in Table 4.

Table 4: HMM Result

	Training	Testing
Precision	0.5354	0.6951
Recall	0.6350	0.3205
Total F1	0.5810	0.4387
Average F1	0.5689	0.5615

Now, all the average F1 score for Baseline, fine-tuned RoBERTa NER model and HMM are listed in Table 5. Among all the three methods, fine-tuned RoBERTa NER model has the best performance with average F1 score as high as 0.6134.

Table 5: Comparison between Baseline, fine-tuned RoBERTa NER model and HMM

	Average F1 score
Baseline	0.1384
Fine-tuned RoBERTa NER model	**0.6134**
HMM	0.5615

5.2.2 Qualitative result

Finally, I would like to provide some raw outputs and the visualized outputs (that can be more easily read by human) to qualitatively show how the models are doing. The highlighted parts are the result predicted by the model while the bold parts are the original ground truth.

There are in total three situations: 1) the predicted spans and ground truth spans are exactly the same. 2) the model fails to predict all ground truth spans of a message, only part of spans or no span is detected. 3) the model predict more toxic spans than the ground truth.

Some successful examples that the predicted spans and ground truth spans are exactly the same are shown below.

- Because you are that **stupid**.
- oh save your **stupidity**, any government negotiating is NOT in any way the market **loser**, this is a federal issue for all of canada, they have jurisdiction not BC!
- Anything to keep the **idiot** out of the house. Must be tough on Trudy to have opposition party members constantly laughing at him.
- You are a **simpleton**, who is full of racial hatred.
- This **fucking idiot** has no clue.

Sometimes the model fails to predict all ground truth toxic spans. The parts in bold without highlighting are not recognized by the model.

- A rock has more mental horsepower than the indubitably **ignorant**. And more integrity as well. They're largely very **dishonest people** who repeat the same lies over and over.
- **Trump troll**...
- a **racist**, **misgynistic**, liar who only brings hate to the table

Still other situations, the model predict toxic spans that are considered to be non-toxic in the ground truth. For example:

- 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.

6 Discussion

6.1 Quantitative analysis

From Table 5 we can see that both fine-tuned RoBERTa NER model and HMM show a great improvements compared to the baseline. Most importantly, the fine-tuned RoBERTa NER model has a F1 score which is 5% higher than the HMM. There can be multiple reasons. a) HMM only encodes 2-gram phrases, which is much smaller than the RoBERTa NER model, which take the whole sentence into consideration. b) HMM can have better performance on frequently appeared toxic spans, but can be really unhelpful to detect frequently appeared words ”and” when used in toxic situation like ”violent and aggressive” since ”and” is appeared to be non-toxic span much more than toxic span.

Moreover, from the score graph of RoBERTa NER model we can see that the precision score is high while the recall score is low. It suggests that the model detects less toxic spans than the ground truth label. Toxic parts that the algorithm successfully detects have high accuracy while there are still a lot of toxic parts that the algorithm does not detect. The main reason is that, some words tend to be toxic in only certain circumstances. The same result is shown is HMM result on testing data set. Notice that the recall of HMM testing result is only 0.32, which is much lower than the lowest recall score of RoBERTa NER model. It corresponds to our previous guess that it is harder for HMM model than RoBERTa NER model to detect neutral words that become toxic spans in the toxic context.

6.2 Qualitative analysis

The most frequently recognized toxic tokens are single or multiple offensive words like ”stupid”, ”simpleton”, ”loser”, ”fucking idiot”, and so on.

In the second situation, words ”people” and ”Trump” in the above context are toxic, but most of the time they are neutral. Also, words like ”misgynistic” is misspelled word of ”misogynistic”. Hence, words that are toxic only under certain situations, and words that are misspelled, can be hard for the model to detect.

In the third situation, the whole sentence is positive, so real people don't classify it as toxic at all. They think that the usage of ”moron” is all right under this situation. In reality, not all offensive words should be completely banned, especially when we need to express some positive opinions with strength.

The misspelled word can be addressed. But neutral words in toxic context and toxic words in non-toxic context are indeed hard to be detected. Through the qualitative analysis we could see that the model still has some limitations. Maybe some manual removal of wrong predictions during training is needed.

6.3 Summary

Overall speaking, I would like to say that the RoBERTa NER model is actually doing well despite that the average F1 score does not seem very high. The model manages to recognize single or multiple offensive words like ”stupid” and ”fucking idiot”; able to recognize continuous phrases like ”violent and aggressive”. And I believe that the problem of hard to detect misspelled offensive words and neutral words used in toxic context can be addressed by manual removal of wrong predictions during training.

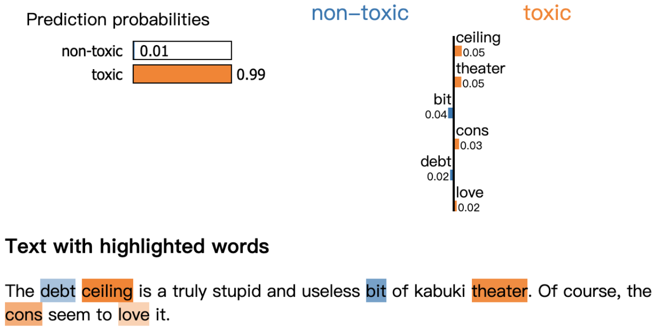
7 Conclusion

To figure out the toxic spans in the raw messages, I employ three methods, fine-tuned RoBERTa NER model, HMM, and fine-tuned RoBERTa classification model. I also preprocess the data set accordingly. From quantitative result, fine-tuned RoBERTa NER model has the highest average F1 score on test data set, which is 0.6134. From qualitative result we can see that fine-tuned RoBERTa NER model do well on single or multiple offensive words; the model can hardly recognize neutral words when used in toxic context and misspelled offensive words; the model can also mistakenly recognize toxic words when used in proper situation. All codes are available on https://github.com/hexinz/SI630_final_project.git

8 Other things I tried

For the third method in section 4.3: Fine-tuned RoBERTa classification model with LIME, the result is unsatisfactory, as a result I did not try out calculating average F1 score. A visualized form of sample result of LIME model is shown in Fig. 3.

The ground truth toxic spans should be ”The debt ceiling is a truly stupid and useless bit of kabuki theater. Of course, the cons seem to love it.” Although the predicted class is correct, the LIME model fails to recognize what is the toxic parts of the sentence. Actually, it is quite common that a model has a high classification accuracy without truly know why one message belongs to one class, because we train the classifier olny considering the loss (or difference) between the predicted class and the truth class. This is why LIME model is proposed.



Text with highlighted words

The debt ceiling is a truly stupid and useless bit of kabuki theater. Of course, the cons seem to love it.

Figure 5: Sample result of LIME model

Moreover, classification in this project also has the problem of uneven number of toxic and non-toxic messages. For all 7949 samples, 7454 are toxic while only 485 samples are non-toxic, which lead to further failure of this method since the classifier itself tend to classify all non-toxic as toxic.

9 What I would have done

It is quite a pity that I failed in using RoBERTa-LIME model. From my perspective, the method is definitely feasible. The uneven distribution of the data is the main problem. If I have time to start over, I would try to find more non-toxic messages, or at least drag sentences in toxic messages that do not have toxic spans to be non-toxic messages. It will largely increase the reliability of the classifier and thus make improvements in the LIME model result.

Besides, I also don’t quite have the time to output all my results in the form like ”This is a stupid example, so thank you for nothing a!@#!@.”. I only output the location of characters: [10,11,12,13,14,15,51,52,53,54,55,56]. If I had time, I would try to design a real-life API with in-built trained model to take in raw messages and output results on a web page that looks like the former one. I might adopt a layout that look like the html shown in Fig. 4.

Finally, I have great interests to also try out method of BiLSTM-LIME or RNN-LIME method. LIME will probably have better performance with a stronger classifier like BiLSTM or RNN. I also want to try out multiple python libraries including keras and transformer for more deep learning models.

In summary, this is a quite interesting project that has great potential and practical usage. Although the model may not be as smart as human inspector, it can save a lot of time and labor especially when so many information are poured into the Internet. Hope such a system can be actually applied on the Internet soon to help maintain a healthy online discussion.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Athanasios Katsiolis. 2020. Toxic span detection in online posts.

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

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938.