TOXIC SPANS DETECTION

Miho Hren, Vedran Kurdija, Luka Pavlović, Dunja Šmigovec
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb
{miho.hren, vedran.kurdija, luka.pavlovic, dunja.smigovec}@fer.hr

January 21, 2021

ABSTRACT

For our bachelors project we participated in SemEval-2021 (the International Workshop on Semantic Evaluation). We chose task 5 - Toxic Spans Detection and learned a lot about deep learning and natural language processing, especially the topics of recurrent neural networks and transformers. We implemented an RNN model, which achieved the F1 score of 0.6.

Keywords NLP · sequence labeling · toxic · LSTM · BERT

1 Introduction

Toxic Spans Detection was the fifth task of the SemEval-2021 workshop. The task itself was to create a model which takes in text, and outputs the toxic part of that text. In machine learning, this problem is known as sequence labeling, and many approaches have been developed, e.g. recurrent neural networks (RNNs, GRUs, LSTMs), convolutial neural networks (CNNs) and transformer neural networks (BERT). This paper covers the use of recurrent neural networks and transformer neural networks in solving the given problem.

2 Dataset

The dataset given by task authors is a proper subset of the Civil comments dataset. The original dataset consisted of comments from approximately 50 English-speaking news sites, created from 2015 to 2017. Each comment was labeled on 7 different criteria, and the authors of this task then filtered only the toxic comments (i.e. the toxic label was 1 instead of zero). So, the given dataset consists of toxic comments, and our task is to figure out which part of the text is toxic.

The data itself is given in a `.csv` file in two columns, the first being `spans`, and the second being `text`. The `spans` column represents indices of toxic parts of the text given in the `text` column. An example entry of the dataset can be `[8,9,10], "he's an ass"`. In this example, the part of the text that is toxic are the last three letters, `ass`. It is worth noting that `spans` are zero indexed.

The dataset is very inconsistently labeled. We believe the reason for this is that the original dataset was collected from random news sites. After that, for the purpose of this task, three crowd-raters were employed per post, which further sullied the data.

We noticed three different types of errors in the toxic labeling.

- complete mislabeling
  something is labeled toxic, but is not
- overlabeling
  many words which aren't toxic are labeled toxic

- whole text labeling or the absence of
  sometimes the whole text is labeled, sometimes nothing is labeled toxic

Since the data contains only toxic comments, we expected at least one word to be toxic. The absence of labels was explained by the authors of the task: when no text is labeled toxic, then the whole text is toxic inherently, but there is no explicit toxicity. An example from the dataset is `"Exposing hypocrites like Trump and Pence is therapeutic for you? Good job!"`.

Even in this context, there were inconsistencies. For example, the text `"Should have just ran his useless meth head ass over."` and `"Only a fectless bitch-boy would cave to such excrement..."` had no spans whatsoever, i.e. there were no toxic words, even though it is obvious which part of the sentence is toxic. There were also examples where every word was labeled toxic, instead of having empty labels. This remark must be taken with a grain of salt, because the authors of this paper might also be biased towards toxicity.

3 Methods and Implementation

We decided to start with the recurrent neural network approach, because this neural network architecture is the standard when dealing with sequence labeling tasks.

We used PyTorch for our deep learning models, pandas for basic data handling, Podium for data processing and spaCy for sentence tokenization and other helper functions.

In the beginning, we needed to introduce deep learning to ourselves, so we built a simple LSTM model without batching. This model later proved to be the best.

Then, we introduced Podium preprocessing and batching into our environment, which greatly simplified our scripts, but as we will later discuss, it reduced our $F1$ score. We believe the cause of this reduction is our misunderstanding of deep learning and the frameworks.

These two models had two LSTM layers, followed by a fully connected layer that reduced the number of hidden dimensions to 3, because we had 3 labels:

- 0 = not in toxic span
- 1 = beginning of toxic span
- 2 = in toxic span

In the end, we took a BERT model from the huggingface library into consideration, since it is the state of the art architecture for many NLP tasks.

4 Experimental results

Our simple model without batching produced the best results. The hyperparameters for that model are shown in Table 1.

Table 1: Simple model hyperparameters

Word embeddings	**FastText**
Architecture	**LSTM**
Input dimension*	300
Hidden dimension	6
Optimizer	**Adam**
Learning rate	0.001
Loss function	**NLL**
Epochs	6
Batch size	1

*the input dimension is the same as the word embedding dimension

This model achieved an F1 score of **0.6**. When predicting spans in our own made up sentences, we saw that the predictions are subjectively good, and deduced that the model is good enough for application in toxic word filtering or censoring.

When using the model with batching, we changed our hyperparameters due to usage of other packages. Those hyperparameters are shown in Table 2.

Table 2: Hyperparameters of the model with batching

Word embeddings	**GloVe**
Architecture	**RNN, GRU, LSTM**
Input dimension*	300
Hidden dimension	300
Optimizer	**Adam**
Learning rate	0.001
Loss function	**Cross Entropy**
Batch size	64, 128, 256

The model results and the epoch on which the result was realized are given in Table 3.

Table 3: Comparison of model performance depending on batch size

batch_size	RNN	GRU	LSTM
64	**0.371 (20)**	0.271 (2)	0.314 (15)
128	0.326 (33)	0.272 (28)	0.291 (32)
256	0.253 (4)	0.229 (2)	0.228 (7)

The best model from the grid search achieved an $F1$ score of 0.3765 on the test set. We expected the LSTM base model to achieve best results, because its architecture is the most suited for this sequence labeling problem, but to our surprise, the RNN model bested the LSTM in each case.

It should be noted that we suspect a logical error in the model with batching, because this model does not produce the same result for the same hyperparameters as the simple model without batching.

We started implementing the BERT model, but due to complications with the dataset format and the nature of the model we ran out of time. Its training is expensive and took a lot of time on our hardware. The problem was that in our dataset, the toxic spans were given as indices of characters which belonged in the toxic spans, as opposed to the traditional way of giving spans - labels. We successfully implemented the conversion from spans to labels and the inverse, and we used the spaCy tokenizer to do it. Implementing the conversion in the BERT model proved to be a more complex problem, because the model used its own tokenizer, which did not provide the same API as the spaCy tokenizer.

5 Lessons learned

Right away we learned that one should first cover the theoretical approach to deep learning, analyze and study it deeply, and then the practical approach will follow easily and intuitively. Since we had little to none background in the topics of DL and NLP, our path was rocky.

Then, we learned the value of reading framework documentation thoroughly - one can save lots of hours by reading the documentation twice from A to Z, rather than playing with methods in interactive environments for hours upon end. Of course, a little bit of both is needed to successfully master a framework in the least possible amount of time.

Also, we learned not to be afraid of asking for help - sometimes one gets fixed on some problem and gets lost in it without realizing it. These moments often produce no fruit, and a wise scientist will reach for help in these situations - at least when they know there is help out there.

6 Possible improvements

First off, we believe the BERT model would improve our F1 score by a lot, so the first step to continuing this project is to implement it.

Also, a wider grid search could be made on the recurrent models - e.g. one of the hyperparameters not included in our grid search was the number of LSTM layers - we kept it at two. Further improvements could be made to the recurrent models, like implementing a CRF layer on top of the recurrent layer, or making the model bidirectional instead of

unidirectional, or implementing a character level CNN below the recurrent layer. These methods should improve the model performance for this task, as shown in [2] and [1].

References

[1] Zhiheng Huang, Wei Xu, Kai Yu *Bidirectional LSTM-CRF Models for Sequence Tagging.* arXiv:1508.01991 [cs.CL], 2015

[2] Xiang Zhang, Junbo Zhao, Yann LeCun. *Character-level Convolutional Networks for Text Classification.* arXiv:1509.01626v3 [cs.LG], 2016