Multi-label Toxic Language Classification of Wikipedia Comments

Lovro Kordiš, Marko Šmitran

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{lovro.kordis, marko.smitran}@fer.hr

Abstract

This paper tackles the problem of classification of toxic text in Wikipedia comments featured in Toxic Comment Classification Challenge. It is a multi-label problem in nature as each comment can fall into multiple categories of toxicity. We approach this task from both classical machine learning and deep-learning angles. In our classical ML approach we used several classifiers, one for binary classification of each category of toxicity. Classifiers were trained on already NLP pre-processed data. We also tried different categories of models – Linear SVM and Logistic Regression. For our deep-learning approach we used models based on LSTMs. Evaluation of models was done on data provided in the challenge. In addition to evaluating models on provided labeled data we further evaluated them with Challenge's official automatic evaluation.

1. Introduction

In the last two decades popularity and usage of the internet exploded. And as the population of users increased, the ways of communication over it also evolved. Today most people in the developed world have access to the internet. In any form of communication between people, situations that might be interpreted as toxic arise. However, perhaps due to the perceived anonymity and physical safety more people engage in toxic behavior online than in real face-to-face communication. A lot of content on the internet is publicly available, and/or stored for a significant amount of time, so any such behavior also has a wider reach. Many places on the internet do not want to condone toxic behavior, so the need to identify such content and perhaps take further action (removal of content, banning the user etc.) is higher then ever. Identification by a human is both inefficient and expensive. So people turned to automatic approach.

In this paper we propose possible solutions to the problem. We focus on identification of toxic content in textual form. Specifically, text in the form of comments from Wikipedia. Although our approaches can be applied to any form of short text. This specific problem was featured in a competition on *Kaggle*[1] ("Toxic Comment Classification Challenge"). Dataset used in this paper is provided by the competition organizers.

Initially we approached the problem from standard machine learning model perspective using SVM and Logistic Regression classifiers for comment classification. Later on we switched to deep-learning model using Bidirectional LSTM in hope of getting better results. Following sections will fully explain all the steps of both approaches. Section 4.1 describes NLP preprocessing steps taken to prepare the data to a satisfactory degree. In Sections 4 and 5 we introduce our models, training and optimization methods used. Section 7 features experimental evaluation of our models and their comparison to both baselines and one another. Finally Section 8 concludes our paper with a summary of our work, insights gathered during our work on the project as well as possible improvements.

[1]https://www.Kaggle.com/c/jigsaw-toxic-comment-classification-challenge

2. Related Work

Problem of identification of toxic texts or similar topics has already been approached from several angles in previous works. Wang and Manning (2012) show that machine learning approach can be effective in NLP problems. Specifically, in (Kennedy et al., 2017) it is shown that ML can be effective in detecting harassment in online communication, topic that is similar to our own.

In the paper with another topic closely related to our own (Sax, 2016), author approaches the problem from a deep-learning standpoint. The author uses several different models (both deep-learning and others) for automatic insult detection. Some of these models use LSTMs and according to their evaluation, these models show great results. In order to tackle the problem with overfitting and to improve results, Lin et al. (2014) propose using Global Average Pooling method, while in (Zhou et al., 2016) we see advantages in combining Convolutional networks with LSTMs.

3. Dataset

We used the training dataset provided by Toxic Comment Classification Challenge on Kaggle.com website. Data consists of Wikipedia comments labeled by human annotators for toxic behavior. Comment consists only of text, no additional information are given (for example prior conversation). Each comment has six distinct labels: toxic, severe toxic, obscene, threat, insult, identity hate. Label value of 1 indicates that the comment belongs to that specific class of toxicity, while 0 indicates that it does not. Each comment can belong to multiple classes. There is a total of 159571 labeled Wikipedia comments in the dataset. Of those, 89.83% are not toxic (they do not belong to any of the above mentioned classes), 9.58% are considered toxic, 0.99% severely toxic, 5.2% obscene, 0.29% are threats, 4.93% insults and 0.88% identity hate. A cursory glance at the data reveals a big imbalance of classes. Imbalance of classes makes training and evaluation of model substantially more difficult.

Table 1: Label distribution in the dataset

Toxic	Severe toxic	Obscene	Threat	Insult	Identity hate	None
9.58%	0.99%	5.29%	0.29%	4.93%	0.88%	89.83%

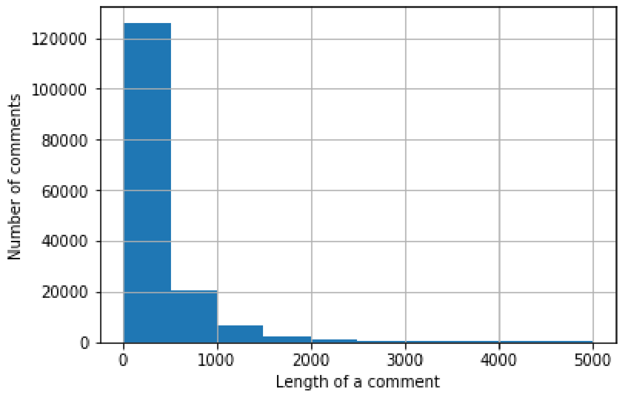

Figure 1: Distribution of comments length

4. Basic Models

The initial task of multi-label classification was made more difficult with the unbalanced dataset (with almost 89 percent of all comments labeled as not toxic). Because of that the baseline was set with over 90 percent accuracy and our main goal was to improve the recall, precision and F1 score metrics.

In this section we describe the process of data preparation and multi-label classification task framing for the SVM and Logistic Regression models.

4.1. Feature Selection

In text pre-processing we only used Snowball[2] stemmer. Lemmatiser was not used as the words in comments can also have lengthened characters or use acronyms and while its often possible to get the original lemma, it comes at the cost of losing information. Features were generated by tokenizing each comment, hashing the resulting n-grams, and computing a TF/IDF value for each token. The resultant feature vectors were used to train a Naive Bayes baseline model, Logistic Regression models and finally SVM model. We used the following features:

- **Unigram and Bigram TF-IDF**: this is a standard feature used in text classification. We used unigrams and bigrams. Bigger ngrams were not used as they provided no more useful information. The size of the dataset meant almost all other ngrams were too rare to have any significance.

- **Named entity recognition**: we used basic named entity recognition with tools provided by NLTK library. We postulated that toxic comments might have some named entities in cases similar to this this example:

  "F##k **Donald Trump** and f##k anyone who still defends him"

- **Language detection**: we used language detection library langdetect. Because the great majority of the comment in the dataset are in English, we supposed foreign language comments might be toxic more often.

To reduce the dimensionality of the feature matrix we also preformed KBest feature selection with χ^2 test on the TF/IDF matrix and selected 500000 best features to use. Named entity recognition (Number of named entities in a comment) and Language detection (A fixed number for every non-english comment) were then added to the reduced matrix as separate columns.

We used NLTK[3] library for all steps except language detection, for which used the langdetect[4] library.

4.2. Multi-label Classification

As previously stated, our task is set as a multilabel classification problem. In machine learning, multi-label classification and the strongly related problem of multi-output classification are variants of the classification problem where multiple labels may be assigned to each instance. Multi-label classification is a generalization of multiclass classification, which is the single-label problem of categorizing instances into precisely one of more than two classes; in the multi-label problem there is no constraint on how many of the classes the instance can be assigned to.

With the absence of useful algorithms that support multi-label classification in the sklearn library, we decided to train 6 separate binary classifiers for each multi-label output. This setup also made the hyperparameter optimization and model evaluation more difficult, as the computing had to be done for every classifier.

5. Deep Learning Models

Multi-label classification is made a lot simpler with deep learning networks, as we do not have to worry about multiple classifiers.

In this subsection we describe the architectures of our several deep learning models that we have created in order to explore the data and to provide context for our results. We also describe the core concepts behind LSTM and Convolutional networks.

5.1. Long Short Term Memory Networks

The main idea of RNN (Recurrent Neural Net), and a big reason of usage in NLP, is in retaining information from

[2]http://snowballstem.org

[3]www.nltk.org

[4]pypi.org/project/langdetect

Table 2: The optimized C parameters for SVM and Logistic Regression models

	Toxic	Severe toxic	Obscene	Threat	Insult	Identity hate
Logistic Regression	128	1	128	64	16	16
SVM	2	0.25	1	1	1	1

Table 3: The optimized parameters of Bi-LSTM and Bi-LSTM + Conv models

	Bi-LSTM	Bi-LSTM+Conv
Dropout	0.15	0.25
LSTM L2 rate	0.001	0.001
BLSTM hidden state	50	40

context of the sentence from the currently processed word to start of the sentence. Despite good results with short sentences, RNN loses its performance with longer text, because of the vanishing gradient problem.

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber (1997). They work tremendously well on a large variety of problems, and are now widely used. Although LSTMs can capture the past context very well, it is desirable to know the future context as well. Bidirectional LSTM solves this by adding another LSTM layer with backwards information flow, which means it can capture the future context.

5.2. Convolutional Networks

CNN networks are famous for their appliance in the Computer Vision domain but have also demonstrated an ability to extract morphological information from word characters, encoding them into neural representations.

5.3. Global Pooling

Dropout and batch normalization can be very effective with feedforward neural networks. However, they can cause problems with RNNs and combined with small dataset can cause overfitting. Thus, we have used pooling methods to stabilise and regularize the training process.

- **Global Average Pooling**: In the last few years, experts (Lin et al., 2014) have turned to global average pooling (GAP) layers to minimize overfitting by reducing the total number of parameters in the model. Similar to max pooling layers, GAP layers are used to reduce the spatial dimensions of a three-dimensional tensor.

- **Global Max Pooling**: Global max pooling is similar to ordinary max pooling layer with the exception of pool size equals to the size of the input.

5.4. Used Models

To test how every architecture handles toxic comment classification we have implemented 2 models based on the following architectures: Bi-LSTM, and a combination of the Bi-LSTM and CNN. We have used pre-trained GloVe word embeddings of vector size 50.

Bidirectional LSTM: The first model is a bidirectional recurrent model with LSTM cells. As input features we use 50 dimensional GloVe (Pennington et al., 2014) vectors trained on the 6B dataset. The Bi-LSTM layer is then preceded by concatenated global max and average pooling values. Fully connected softmax layer is then placed at the networks end.

Bidirectional LSTM with Convolutional layer: A variation of the first model, with Convolutional layer put on top of Bi-LSTM.

6. Model Optimization

We split this section into two parts: basic models and deep learning models parameter optimization.

6.1. Basic Models

As previously stated, because of our task framing, we had to optimize parameters for 6 different binary classifiers. We optimized parameter *C* for **Linear SVC** and **Logistic Regression**. Table 2 shows the optimized parameters for each classifier and model.

6.2. Deep Learning Models

This was a much harder task; because of our limited computing resources and slow training time, we properly optimized Dropout and L2 regularization rates, whereas for hidden layer sizes and batch sizes our method was to test each parameter low and extreme values and then interpolate the best values. Table 3 shows the optimized parameters for each deep learning model.

7. Evaluation and Results

As predicted, all our models have very high accuracy, but the evaluation scores differ on recall, precision and f1 metrics. During the model training and evaluation, we used nested k-fold cross-validation (10 folds) technique with grid search for model parameters. We also compare the results with other solutions at *Kaggle* competition. Our experiments were conducted on an Intel i5 CPU and GTX 1050Ti GPU.

To get an idea of how well our models perform, we used two baseline models. A Naive Bayes classification model and a dummy classifier where we assigned all classes values to 0. The evaluation results shown in Table 4 show we got

Table 4: Evaluation results for our models and baselines

	Accuracy	Precision	Recall	F1
Dummy	0.963268	/	/	/
Naive Bayes	0.971666	0.738822	0.118246	0.188864
LogReg	0.981859	0.655741	0.560120	0.601495
SVM	**0.981954**	0.675585	0.490702	0.552313
Bi-LSTM	0.980480	**0.820342**	0.575812	**0.659872**
Bi-LSTM+Conv	0.980136	0.809786	**0.580549**	0.655871

Table 5: Results of Kaggle competition

Team name	ROC AUC score
Toxic Crusaders	0.9885
Bi-LSTM	0.9804
BI-LSTM+Conv	0.9772
SVM	0.9769
LogReg	0.9767
WeAreRobot	0.9785

the best recall, precision and F1 results with LSTM models, while Logistic Regression and SVM models had higher accuracy. We think the cause of lower accuracy of LSTM models might be due to overfitting on the later epochs of the training process.

Submissions on *Kaggle* competition are evaluated on the mean column-wise ROC AUC. In other words, the score is the average of the individual AUCs of each predicted column. Table 5 shows our models comparison with the winner ("Toxic Crusaders") and an mid-table solution ("WeAreRobot"). Our solution was placed in the upper part of the table

8. Conclusion and Future Work

We proposed 4 different models for solving multi-label toxic comment classification as part of the *Kaggle* competition: Logistic Regression, SVM, Bi-LSTM and Bi-LSTM with Convolutional layer. The provided dataset proved to be somewhat challenging because of the sparsity of toxic comments. We used TF-IDF, Named entity recognition and Language detection features with SVM and Logistic Regression models, and pre-trained GloVe word embeddings for deep learning models. All models had very high accuracy, but deep learning models preformed better in precision, recall and F1 metrics.

For future work, we would experiment with new word embeddings trained only on the toxic comment dataset. We believe it is crucial for word vectors to learn semantic meaning from the domain concrete data because of the toxic comment semantic strucutre. Also, we propose using a different dataset or even creating a new one from Twitter and other sources, which would provide more toxic comment examples.

References

Jennifer Bayzick and Lynne Edwards. 2011. Detecting the presence of cyberbullying using computer software.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. 9:1735–80, 12.

George W. Kennedy, Andrew W. McCollough, Edward Dixon, Alexie Bastidas, John Ryan, Chris Loo, and Saurav Sahay. 2017. Hack harassment: Technology solutions to combat online harassment. *Proceedings of the First Workshop on Abusive Language Online*, pages 73—-77.

Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network in network.

Andrew McCallum and Kamal Nigam. 2001. A comparison of event models for naive bayes text classification. *Proceedings of the First Workshop on Abusive Language Online*.

Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. *Glove: Global Vectors for Word Representation*, volume 14. 01.

Sasha Sax. 2016. Flame wars: Automatic insult detection.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. 11.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *COLING*.