Avoiding Unintended Bias in Toxicity Classification with Neural Networks

Sergey Morzhov
Yaroslavl State University
Yaroslavl, Russia
smorzhov@gmail.com

Abstract—**The growing popularity of online platforms that allow users to communicate with each other, exchange opinions about various events and leave comments, has contributed to the development of natural language processing algorithms. Tens of millions of messages per day published by users of a certain social network must be analyzed in real time for moderation to prevent the spread of various illegal or offensive information, threats and other types of toxic comments. Of course, such a large amount of information can be processed quite quickly only automatically. That is why it is necessary to find a way to teach a computer to "understand" a text written by a man. It is a non-trivial task, even if the word "understand" here means only to detect or classify. The rapid development of machine learning technologies has led to the widespread adoption of new algorithms. Many tasks that for years were considered almost impossible to solve using computer now can be successfully solved with deep learning technologies. In this article, the author presents modern approaches to solving the problem of toxic comments detection using deep learning technologies and neural networks. The author introduces two state-of-the-art neural network architectures and also demonstrates how to use a contextual language representation model to detect toxicity. Furthermore, in this article will be presented the results of the developed algorithms, as well as the results of their ensemble, tested on a large training set, gathered and marked up by Google and Jigsaw.**

I. Introduction

Natural language processing (NLP) has been an attractive research goal for many years, since solving this task in its general form will allow creating a natural language interface, which will greatly simplify and expand the scope of human-computer interaction. Understanding of natural language in itself is a non-trivial problem. It is considered to be an AI-complete, because recognizing a natural language requires a lot of knowledge about the environment and the ability to interact with it. However, in solving certain classes of problems, for example, text classification or sentiment analysis, great advancement has been made recently due to the development of neural network algorithms and the advent of high-performance processors and graphic cards. This progress allowed the use of deep neural networks to solve various problems associated with NLP, which previously could not be successfully solved using classical algorithms.

Nowadays, online platforms have become widespread, allowing their users various types of interaction with each other, including through messaging. Various social networks, online game platforms, photo and video sharing applications, news portals integrate chats into their products, implement the ability to leave comments, and allow users to communicate with each other. This functionality is vulnerable to many types of Internet crimes, such as personal insults and threats, various types of propaganda, fraud, advertising of illegal goods and services. Illegal and toxic comments have to be deleted, and even better, there should be a possibility of preventing their publication. Thus, there is a need for presence of sufficiently fast and efficient algorithms capable to process all user messages in real time.

The Conversation AI team, a research initiative founded by Jigsaw and Google (both part of Alphabet), builds technology to protect voices in conversation. A main area of focus for them is machine learning models that can identify toxicity in online conversations, where toxicity is defined as anything rude, disrespectful or otherwise likely to make someone leave a discussion. That companies host the contest, which aimed to create an algorithm to solve the problem of detecting toxic comments [1], [2]. This indicates the relevance, as well as the insufficient level of research of this problem, since published algorithms that allow to solve it (see [3], [4], [5], [6]) has poor accuracy according to the organizers of the competition.

The author in this article presents two state-of-the-art neural network architectures designed to solve the problem of detecting toxic comments, and also demonstrates how to use a contextual language representation model to detect toxicity using BERT as an example to obtain an accuracy level close to that of state-of-the-art models. Also, the paper contains the results of comparison testing of introduced algorithms and some existing ones that solve the above-mentioned problem. In addition, the paper includes a number of remarks regarding further work to improve the accuracy of the presented models.

II. Problem statement

At the end of 2017 Civil Comments platform shut down and chose to make their large public comments data set available in a lasting open archive so that researchers could understand and improve civility in online conversations for years to come. Jigsaw sponsored this effort and extended annotation of this data by human raters for various toxic conversational attributes.

When the Conversation AI team first built toxicity models, they found that those models incorrectly learned to associate the names of frequently attacked identities with toxicity. Models predicted a high likelihood of toxicity for comments containing those identities (e.g. "gay"), even when those comments were not actually toxic (such as "I am a gay woman"). This happens because training data was pulled from available sources where unfortunately, certain identities are overwhelmingly referred to in offensive ways. Training a model from data with these imbalances risks simply mirroring those biases back to users. That is why Jigsaw company launched a competition to challenge researchers to build a model that recognizes toxicity and minimizes this type of unintended bias with respect to mentions of identities. Thus, the problem objective is to build a model that would be able to detect toxicity accurately enough and reduce unwanted bias.

III. Data analysis

To build a robust model, which can successfully solve the assigned task, data analysis must be performed carefully and thoroughly. Jigsaw's training set is quite large — about 2 000 000 comments. Each comment in the training set has a toxicity label — a target value, which models must be taught to predict. The target value range between 0.0 and 1.0 and represent the fraction of raters who believed the label fits the comment. A comment with the target value greater than 0.5 is considered as toxic. In addition to the target value, the data set also contains some other useful information — gradations of toxicity ("obscene", "identity_attack", "insult", etc.). This data has no empty values — all comments are tagged. To obtain the toxicity labels, each comment was shown approximately to 10 annotators. Some comments were seen by many more than 10 annotators (up to thousands), due to sampling and strategies used to enforce rater accuracy. They were asked to rate toxicity of comments as:

1) Very Toxic (a very hateful, aggressive, or disrespectful comment that is very likely to make you leave a discussion or give up on sharing your perspective)
2) Toxic (a rude, disrespectful, or unreasonable comment that is somewhat likely to make you leave a discussion or give up on sharing your perspective)
3) Hard to Say
4) Not toxic

These ratings were then aggregated with the target value.

The training set is extremely unbalanced — there are much more "clean", non-offensive, comments than toxic ones, which is logical, since in real life there is usually quite the same proportion. 29% of samples have value of target higher than 0 and only 7.99% of samples have target higher than 0.5. The counts of target bins are shown in Fig. 1.

Additionally, a subset of comments (approximately 30%) have been labelled with variety of identity attributes, representing identities that were mentioned in the comment. This attributes include data characterizing religion, race, gender, sexual orientation, etc., such as male, female, heterosexual, homosexual, christian, jewish, muslim, black, white. Information about identities mentioned in the comments were collected the same way as about toxicity. It is worth noting that different comments with similar text may have been labeled with different targets or subgroups.

Here are some examples of comments and their associated toxicity and identity labels.

1) *I'm a white woman in my late 60's and believe me, they are not too crazy about me either!!*
    - **Toxicity Labels:** All 0.0
    - **Identity Mention Labels:** female: 1.0, white: 1.0 (all others 0.0)
2) *Why would you assume that the nurses in this story were women?*
    - **Toxicity Labels:** All 0.0
    - **Identity Mention Labels:** female: 0.8 (all others 0.0)
3) *Continue to stand strong LGBT community. Yes, indeed, you'll overcome and you have.*
    - **Toxicity Labels:** All 0.0
    - **Identity Mention Labels:** homosexual: 0.8, bisexual: 0.6, transgender: 0.3 (all others 0.0)

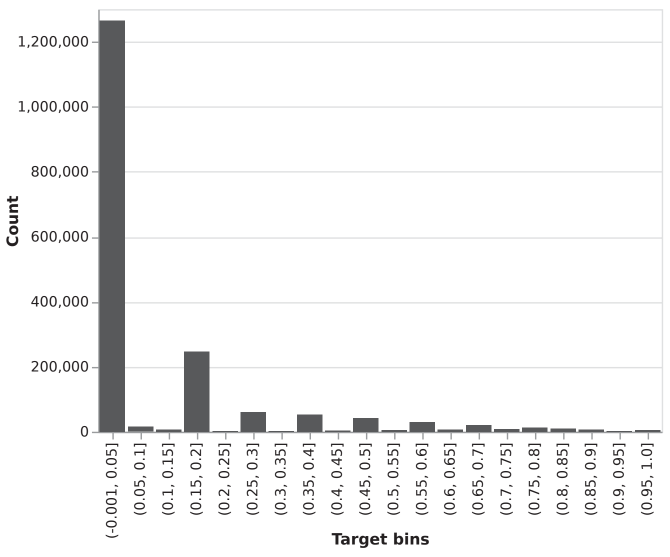


Fig. 1. Counts of target bins

In addition to the labels described above, the data set also provides some metadata such as dates when comments were created, the number of likes as well as the number of funny, wow, sad or disagree emojis pinned to some comments by other users.

The number of words in each comment also vary widely and have the following characteristics: minimum number of words is 1, maximum — 317, mathematical expectation — 53.78, standard deviation — 81.23 (see Fig. 2).

There is another important feature: the number of unique words in each comment. This characteristic can help in solving the problem, since it is easy to notice that authors of toxic comments are not very inventive in their vocabulary. They all usually use certain words specific for all toxic comment authors. Thus, it is necessary to test the following hypothesis: is there a correlation between the different characteristics of the comments related to the number of unique words and the toxicity of the comment. It is easy to note that there are eye-catching shifts in the average number of words and the number of unique words in clean and toxic comments (see Fig. 3). In addition, if you look at the graph in Fig.4, you can see the bulge next to the $0 - 10\%$ mark indicating a large number of toxic comments that contain very few diverse words.

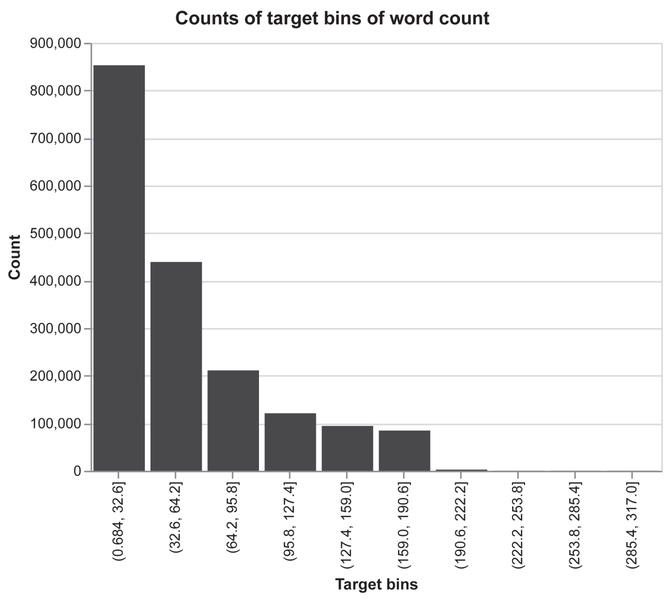


Fig. 2. Counts of target bins of word count

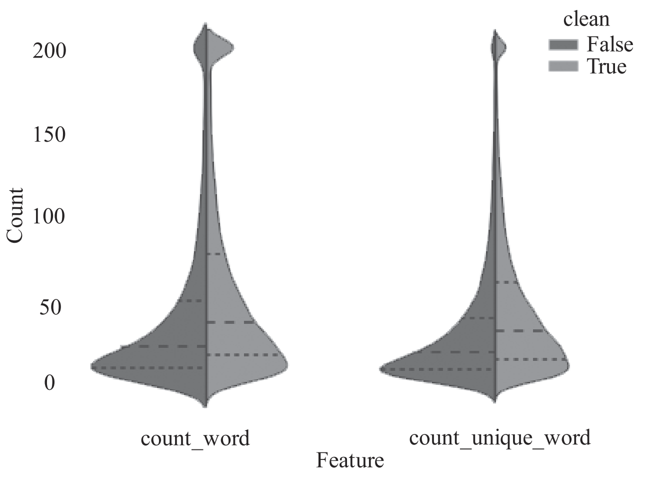


Fig. 3. Absolute word count and unique word count

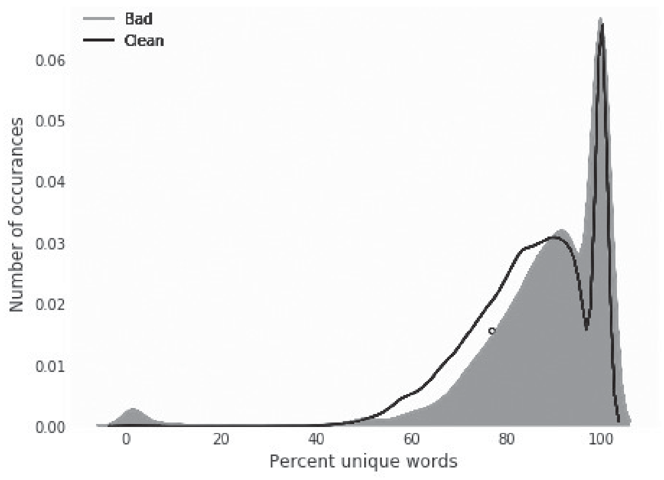


Fig. 4. Percentage of unique words of the total number of words in a comment

All these observations must be taken into account when developing neural network algorithms to solve the problem of detecting toxic comments. All the aforementioned statistical features of the training set will help later in the interpretation of the results, and they can also be used to debug and improve the accuracy of predictions of developed models.

In addition to the statistical characteristics of the training set, it is also necessary to consider characteristics of textual data. Comments are mainly written in English (the training set contains comments in other languages, but their number is less than 0.1% of the total). Many comments contain emojis, an excess of punctuation marks, web links, numbers, unusually written words, grammatical errors. Some grammatical errors were made by authors intentionally in order to disguise offensive or obscene statements. All this senselessly expands the size of the dictionary, which in turn complicates the analysis of comments for a neural network. Therefore, the data preprocessing is an important step in solving the problem.

IV. Data preprocessing

It is important to note that standard text operations, which are still widely used by many scientists in data preprocessing, such as stemming, lowercasing or stopword removal should not be used in this case. The reason they are not performed is simple: valuable information that can help a neural network to figure things out may be lost.

When solving any NLP task, the problem of representing words in a computer-friendly way arises inevitably. One way to solve this problem is to match words (and possibly phrases) from some finite fixed dictionary of size N and vectors from $\mathbb{R}^n, n << N$. Those vectors should also reflect semantic similarity of words, i.e. similar vectors (for example, in terms of cosine similarity), must denote words that have similar meaning. These requirements are met by pretrained word embeddings. In case of their use, text preprocessing must be performed carefully to get the corpus vocabulary as close to the used embedding as possible. Ideally, text preprocessing steps should be quite similar to ones were used by the creators of the chosen word embedding model. This will help to achieve the best possible accuracy.

Taking into account all of the above, it was decided to use two-stage data preprocessing. At the first stage, the basic, simplest manipulations with comments were carried out (step 6 must be omitted while using BERT):

1) Removing pieces of html code that are present in some comments.
2) Converting a substring of type “w h a t a n i c e d a y” to a type “what a nice day”. This type of distortion of the text is often used to mask obscene words.
3) Removing links, ip-addresses.
4) Removing numbers and digits.
5) Removing all punctuation except “’”, “.”, “!”, “?”.
6) Replacing the sentence end marks with special tokens. “!” was replaced by “ exclmrk ”, “?” — by “ qstmrk ”, “.” — by “ eosmkr ”. This was done in order not to lose information about these marks at the stage of transformation of the text into a word embedding.

At the second stage, more labor-intensive operations of text correction and cleaning were carried out (step 2 must be omitted while using BERT):

1) Replacing emojis with the corresponding words (":-(" was replaced by "sad" etc.).
2) Explanation of some abbreviations ("won't" — "will not", "'ll" — "will" etc.).
3) Correction of obscene words.
   For example, the words "f*ck", "fu**" etc. were replaced by the corresponding ones without asterisks.
4) Correction of other grammatical errors.
   It is a very important step because it reduces the number of unknown words and makes transformation of sentences into vectors using a pretrained word embedding more accurate increasing embedding coverage.

We are confident that the above operations are not interesting enough to examine each of them in detail, and if necessary, no one should have difficulties to implement each of them independently. It was decided to divide the data preprocessing algorithm into two stages in order to reduce the cross-correlation of the models at the ensemble stage. Each model was trained on data that passed either only the first stage of processing, or both stages, which increased the variability of their predictions. For real-world use cases both stages of preprocessing must be performed.

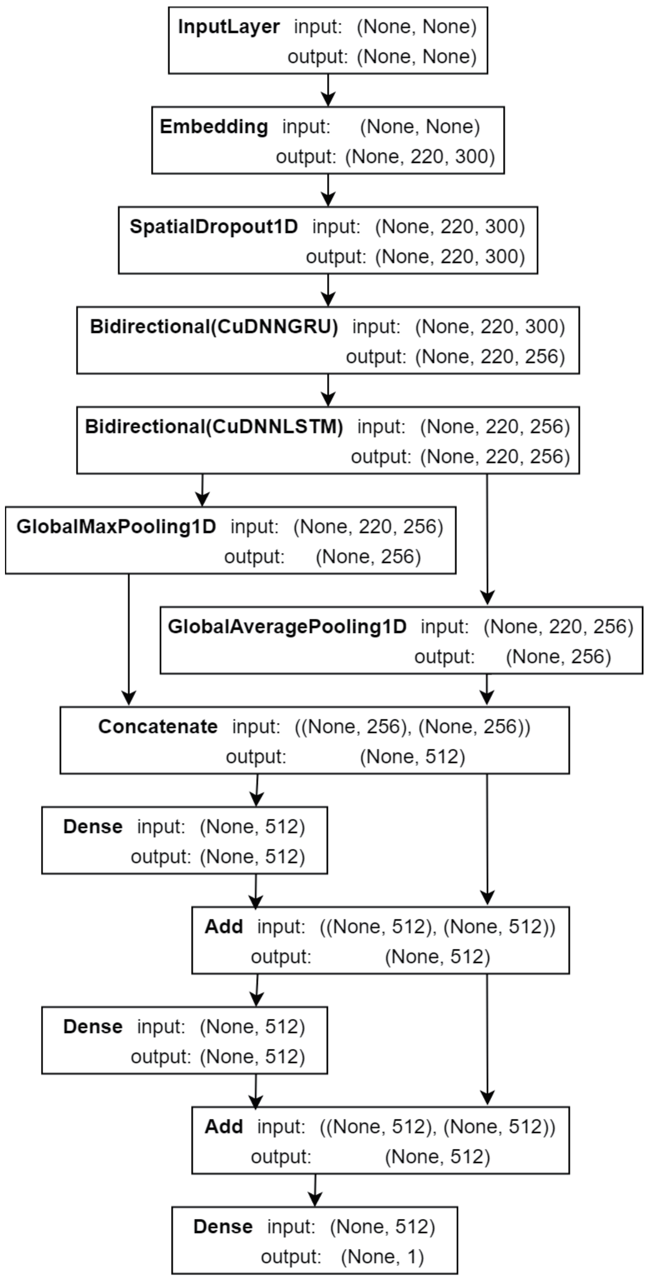
V. Models

Recurrent neural networks (RNNs) have proven themselves well to solve various NLP problems. The idea behind RNNs is to make use of sequential information. The term "recurrent" is used because they perform the same task for each instance of the sequence, so the output depends on previous calculations and results. Typically, a fixed-size vector is created to represent a sequence by feeding tokens one by one into a recurrent block. In a sense, an RNN "remember" the previous calculations and uses this information in the current processing. Such types of RNN as LSTM or GRU are best cope with the tasks of text classification, so when developing models to solve the problem of detecting toxic comments, it was decided to use them [7], [8].

A. *Bi-GRU-LSTM*

A model combining GRU and LSTM layers was chosen as the base for solving the problem of toxic comments detection [9], [10]. It is deep enough to extract the required number of features from a large training set (a total number of comments in the training set is about 1 600 000, taking into account the use of 10-fold cross-validation). The architecture of Bi-GRU-LSTM model is shown in Fig. 5. Hereinafter, CuDNNGRU and CuDNNLSTM layers denote to ordinary GRU and LSTM layers, optimized for computation on GPU.

In the first stage this model utilizing the sequence of bidirectional LSTM and GRU cells to construct the features. To deal with overfitting SpatialDropout1D layer is used with dropout rate equal to 0.2. This layer performs the same function as ordinary dropout layer, however it drops entire 1D feature maps instead of individual elements. Originally, it is



InputLayer input: (None, None) output: (None, None)
Embedding input: (None, None) output: (None, 220, 300)
SpatialDropout1D input: (None, 220, 300) output: (None, 220, 300)
Bidirectional(CuDNNGRU) input: (None, 220, 300) output: (None, 220, 256)
Bidirectional(CuDNNLSTM) input: (None, 220, 256) output: (None, 220, 256)
GlobalMaxPooling1D input: (None, 220, 256) output: (None, 256)
GlobalAveragePooling1D input: (None, 220, 256) output: (None, 256)
Concatenate input: ((None, 256), (None, 256)) output: (None, 512)
Dense input: (None, 512) output: (None, 512)
Add input: ((None, 512), (None, 512)) output: (None, 512)
Dense input: (None, 512) output: (None, 512)
Add input: ((None, 512), (None, 512)) output: (None, 512)
Dense input: (None, 512) output: (None, 1)

Fig. 5. Architecture of Bi-GRU-LSTM model.

widely used in early convolution layers, where adjacent frames within feature maps are strongly correlated, which leads to fact that regular dropout will not regularize the activations and will otherwise just result in an effective learning rate decrease. As it was shown in [11] SpatialDropout1D will help promote independence between feature maps. In the second stage a simple neural network (NN) reduces the dimensionality of the obtained feature matrix using two global pooling layers, then processes the new features with three regular densely-connected NN layer and finally gets the desired prediction.

This model was trained on one Tesla V100 GPU. The batch size was 256. Early stopping technique was used.

B. Bi-GRU with attention mechanism

This model is similar to Bi-GRU-LSTM, except that two GRU layers and an attention mechanism are used sequentially here [12]. In the original paper, the attention mechanism was applied to a machine translation problem. However, nothing prevents us from using it to solve the toxic comment detection problem, which is very similar to text classification [13]. At its core, the attention mechanism is nothing more than an improved encoder-decoder. Using GRU or LSTM lets us to extract and store information about the structure of the sequence, while not allowing to assign different weights to the elements of the obtained sequences. It is obvious that in the task of detecting toxic comments different words have different importance to predict toxicity: obscene words should have more weight, being a good signal that the comment is probably toxic. Neither LSTM nor GRU are capable of simulating this in contrast to the attention mechanism. In other words, the attention mechanism is able to assign different weights to each element of the processed sequence. The larger the weight, the more important the given word, therefore it needs to be paid more attention on it. The architecture of Bi-GRU with attention mechanism model is shown in Fig. 6.

In this model the second LSTM layer was replaced by GRU, since the comment length is short enough and the advantage of LSTM over GRU is potentially achieved only when processing sufficiently long sequences. At the same time GRU controls the flow of information like an LSTM unit, but without having to use a memory unit, and this makes it computantionally more efficient. In addition, attention mechanism is used to reduce the dimensionality of feature matrix instead of pooling layers as it was in Bi-GRU-LSTM model. While global pooling chooses the maximum hidden unit across all of the hidden vectors in the sequence, the attention mechanism instead first learns the attention score for each time step and then computes the temporal average of all hidden vectors. To prevent overfitting, SpatialDropout1D and ordinary Dropout layers were used with 0.2 and 0.5 rates respectfully.

This model was trained on one Tesla V100 GPU. The batch size was 256. Early stopping technique was used.

C. BERT

BERT (Bidirectional Encoder Representations from Transformers) is a new method of pre-training contextual language representation model developed by Google AI Language team [14]. BERT was trained using only a large plain text corpus (like Wikipedia), which is important because an enormous amount of plain text data is publicly available on the web in many languages. It is able to obtain state-of-the-art results on a wide range of NLP tasks. The main difference from context-free representations, such as FastText [16] or GloVe [15], is that BERT generates representations for each word based on its context or other words in a sentence. Therefore, it can deal with polysemantic words. For example, context-free models will generate single representation for a word “bank” even if it meant “river bank” or “bank deposit”.

The technique of using such models is quite simple. Generally, pre-trained language models are used to get embeddings. Then, on top of that embeddings a layer or two of neural networks can be added to fit a particular task. This works very well if the data on which the language model was trained is similar to the data on which we would like to train the new model. If our data is different than the data used for pretraining, the results would not be that satisfactory. To deal with it, the technique called fine-tuning can be used, because it can lead to performance enhancement [17]. To fine-tune the model all its weights except the weights of last few layers must be frozen. The last few layers can even be replaced with others that better suit to the task.

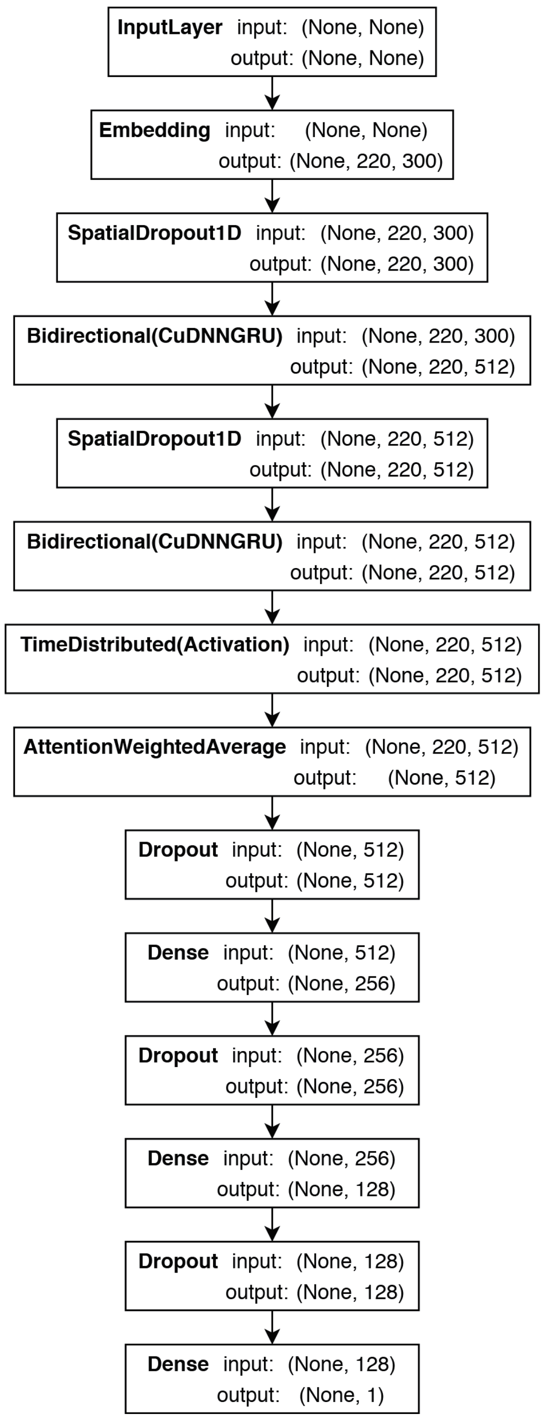

InputLayer input: (None, None) output: (None, None)
Embedding input: (None, None) output: (None, 220, 300)
SpatialDropout1D input: (None, 220, 300) output: (None, 220, 300)
Bidirectional(CuDNNGRU) input: (None, 220, 300) output: (None, 220, 512)
SpatialDropout1D input: (None, 220, 512) output: (None, 220, 512)
Bidirectional(CuDNNGRU) input: (None, 220, 512) output: (None, 220, 512)
TimeDistributed(Activation) input: (None, 220, 512) output: (None, 220, 512)
AttentionWeightedAverage input: (None, 220, 512) output: (None, 512)
Dropout input: (None, 512) output: (None, 512)
Dense input: (None, 512) output: (None, 256)
Dropout input: (None, 256) output: (None, 256)
Dense input: (None, 256) output: (None, 128)
Dropout input: (None, 128) output: (None, 128)
Dense input: (None, 128) output: (None, 1)

Fig. 6. Architecture of Bi-GRU with attention mechanism model

To be able to use BERT model for classification the text must be in the appropriate format. The process of text transformation, as well as how to fine-tune the model is described in detail in [18]. The models was fine-tuned at two Tesla V100 GPUs. Standard parameters were used.

VI. TEST RESULTS

Taking into advantage the aforementioned toxic comment detection problem, it becomes clear that evaluation metric must be able to balance overall performance with various aspects of unintended bias. It was chosen to use the following evaluation metric, which was presented in [19].

$$toxic\_score = w_0 AUC_{overall} + \sum_{a=1}^{A} w_a M_p(m_{s,a}), \qquad (1)$$

$$M_p(m_s) = \left(\frac{1}{N} \sum_{s=1}^{N} m_s^p \right)^{\frac{1}{p}}, \qquad (2)$$

where

- M_p — the pth power-mean function
- m_s — the bias metric m calculated for subgroup s
- N — number of identity subgroups
- A — number of submetrics (3)
- $m_{s,a}$ — bias metric for identity subgroup s using submetric a
- w_a — a weighting for the relative importance of each submetric; all four w values set to 0.25

To evaluate presented models, we decided to follow the advice of the organizers of the Kaggle competition and used a p value equal to -5 [2]. It helped to encourage us to improve the model for the identity subgroups with the lowest model performance.

Table I shows the results of various models using the prepared data obtained according to the previously described cleaning algorithm (two-stage preparation was used). The second column contains 10-fold cross-validation $toxic\_score$. It should be pointed out that according to the organizers of the Kaggle competition, they used the same metric to evaluate models and form the leaderboard. Thus, it means that the results shown in the table I can be compared with the results from the competition leaderboard.

TABLE I. COMPARISON OF THE RESULTS OF VARIOUS MODELS FOR TOXICITY DETECTION

Model	10-fold cross-validation $toxic\_score$	Test $toxic\_score$
CNN [3]	0.8349	0.8288
LSTM [4]	0.8789	0.8710
LSTM-CNN [5]	0.9134	0.9098
Bi-GRU-LSTM	0.9415	0.9393
Bi-GRU with attention mechanism	0.9431	0.9401
BERT base uncased	-	0.9437
BERT base cased	-	0.9439
Ensemble	0.9466	0.9460

To evaluate the models from [3], [4], [5], they were trained on the same data as the models presented in this article. Of course, it cannot be said for sure, that those models necessarily represent the state-of-the-art, but since we used for evaluation the same metric as in the competition, our results can be compared directly with the leaderboard, which better reflects the current level of progress in this problem. The result shown by our ensemble is quite high and could get into top 20 or 30.

The ensemble includes Bi-GRU-LSTM and Bi-GRU with attention mechanism models presented in this article with GloVe and FastText words embeddings and also 2 BERT models. Thus, 6 models were included in the ensemble. In addition, a technique called “seed averaging” was used (it was applied to Bi-GRU-LSTM and Bi-GRU with attention mechanism models), which consisted in launching one model with initializing a pseudorandom number generator with different values and averaging its predictions.

To better illustrate the work of the presented models, we test them on several sentences that contain references to some identities that most often suffer from toxic comments. As noted above, these sentences are complex for classical toxicity detection algorithms. An old Perspective API [20], developed by Jigsaw, was used to demonstrate the serious reduction of unintended bias. The results are shown in table II.

TABLE II. COMPARISON OF THE RESULTS OF AN OLD PERSPECTIVE API AND BI-GRU WITH ATTENTION MECHANISM MODEL

Sentence	Perspective API $toxic\_score$	Bi-GRU with attention mechanism $toxic\_score$
I am a man	0.20	0.0061
I am a woman	0.41	0.0069
I am a white man	0.66	0.01
I am a black man	0.80	0.07
I am a gay white man	0.80	0.30
I am a gay black man	0.82	0.14
I am a black woman	0.85	0.03
I am a gay black woman	0.87	0.13

It should be emphasized that these results were obtained before Perspective has updated the model used to get these toxicity ranks. At the beginning of February 2020, the results obtained with Perspective API are significantly better and quite close to the results of our Bi-GRU with attention mechanism model.

VII. CONCLUSION

Deep learning technologies can minimize human participation in the development of algorithms, since creation of features specific to a particular task is automated. In this paper it was shown how to use deep NNs to solve the problem of detecting toxic comments. The results and accuracy of predictions obtained by the ensemble and each developed model separately, overwhelmed the results of the models from previously published works on this topic, which indicates the success of the work done.

Further improvements in the accuracy of model predictions can be achieved using different augmentation techniques that increase the size of the training data set. Before each epoch during the training of a model, a subset of the comments from the training subsample has to be enriched with augmented data. This can be done, for example, using machine translation technique. Randomly chosen comments have to be translated into some language, for example, German, after which they are translated back into English. Such a conversion should not greatly distort the meaning of those comments, but at the same time it increases the size of the training data set. There is also a simpler method to increase the amount of training data: instead of translation simple string concatenation can be used. Before each epoch during the training of the model some comments from the training subsample can be concatenated together. Thus, new, longer comments will be obtained. Labels for them can be assigned by union of sets of labels of the original comments.

It is also worth trying to combine deep learning technologies with decision trees and gradient boosting [21]. During the preparation and cleaning of the data, some of the information was lost. It can be restored manually by creating new features, the vector of which should be added as an additional input to the models to get more optimal predictions. Such new features include the following: the number of grammatical errors, the length of the comment, the number of obscene words, etc. Such complication should not seriously affect the training time but according to the rough estimates it is able to improve the quality of predictions by about $0.001 - 0.002$ for the chosen metric.

Acknowledgment

We would like to show our gratitude to the Yaroslavl State University for providing access to the computing cluster for training and evaluating the models presented in this article.

References

[1] Toxic Comment Classification Challeng, Web: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge.

[2] Jigsaw Unintended Bias in Toxicity Classification, Web: https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification.

[3] S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, V. P. Plagianakos, “Convolutional Neural Networks for Toxic Comment Classification”. arXiv:1802.09957 [cs.CL], Feb. 2018.

[4] M. Kohli, E. Kuehler, J. Palowitch, “Paying attention to toxic comments online”. Web: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6856482.pdf.

[5] Th. Chu, K. Jue, M. Wang, “Comment Abuse Classification with Deep Learning”. Web: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/2762092.pdf.

[6] K. Khieu, N. Narwal, “Detecting and Classifying Toxic Comments”, Web: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6837517.pdf.

[7] S. Hochreiter, J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol.9, 1997, pp. 1735–1780.

[8] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation”. arXiv:1406.1078 [cs.CL], Jun. 2014.

[9] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. arXiv:1412.3555 [cs.NE], Dec. 2014.

[10] S. Morzhov, “Modern methods for the detection and classification of toxic comments using neural networks”, Modeling and Analysis of Information Systems, in press.

[11] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, Ch. Bregler, “Efficient Object Localization Using Convolutional Networks”. arXiv:1411.4280 [cs.CV], Nov. 2014.

[12] D. Bahdanau, K. Cho, Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate”. arXiv:1409.0473 [cs.CL] Sep. 2014.

[13] Z. Yang, D. Yang, Ch. Dyer, X. He, A. Smola, E. Hovy, “Hierarchical Attention Networks for Document Classification”. Web: https://www.cs.cmu.edu/-/hovy/papers/16HLT-hierarchical-attention-networks.pdf.

[14] J. Devlin, M. -W. Chang, K. Lee, K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. arXiv:1810.04805 [cs.CL], Oct. 2018.

[15] J. Pennington, R. Socher, C. D. Manning, “GloVe: Global Vectors for Word Representation”, *EMNLP*, 2014, pp. 1532–1543.

[16] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, “Bag of Tricks for Efficient Text Classification”, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 427–431.

[17] J. Howard, S. Ruder, “Universal Language Model Fine-tuning for Text Classification”. arXiv:1801.06146 [cs.CL] Jan. 2018.

[18] Fine-tuning with BERT, Web: https://github.com/google-research/bert#fine-tuning-with-bert.

[19] D. Borkan, L. Dixon, J. Sorensen, N. Thain, L. Vasserman, “Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification”. arXiv:1903.04561 [cs.LG], Mar. 2019.

[20] Perspective: What if technology could help improve conversations online? Web: https://perspectiveapi.com/#/home.

[21] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, D. Brown, “Text Classification Algorithms: A Survey”. arXiv:1904.08067 [cs.LG], Apr. 2019.