TROLL TWEET DETECTION USING CONTEXTUALIZED WORD REPRESENTATIONS

Seyhmus YILMAZ¹ And Sultan ZAVRAK¹

Seyhmus YILMAZ
¹Department of Computer Engineering, Duzce University, Duzce/TURKEY
seyhmusyilmaz@duzce.edu.tr
ORCID ID: 0000-0001-9987-2797

Sultan ZAVRAK
¹Department of Computer Engineering, Duzce University, Duzce/TURKEY
sultanzavrak@duzce.edu.tr
ORCID ID: 0000-0001-6950-8927

*Corresponding Author:
Sultan ZAVRAK

Department of Computer Engineering,
Faculty of Engineering, Duzce University, 81620, Duzce / TURKEY
Phone: +90 (380) 542 1036
E-mail: sultanzavrak@duzce.edu.tr
TROLL TWEET DETECTION USING CONTEXTUALIZED WORD REPRESENTATIONS

Abstract
In recent years, numerous troll accounts that manipulate social media sentiment have emerged. Due to the use of fake and automated accounts by businesses, abusers, and nation-state-sponsored troll farms, detecting and eliminating trolling is a crucial issue for social networking platforms. Various NLP techniques are used to extract information from social networking text, such as tweets on Twitter, to identify the messages originating from fake accounts. In this context, this paper implements and compares nine deep learning-based architectures for troll tweet detection, with three models for each BERT, ELMo, and GloVe word embedding model. The majority of BERT-based architectures improve the detection of trolling tweets, as demonstrated by experimental results. An ELMo-based architecture with a GRU classifier has the highest AUC for detecting troll messages. Future socially-based systems can utilize the proposed architectures to identify troll messages.

Keywords: troll tweet detection, BERT, ELMo, natural language processing, word embedding

1. Introduction
Recently, the growth and ubiquity of the Internet results in the construction of many social media environments, which specialize in communication services [1]. In addition to firms [2]–[4], the users [5] have used social media in an extensive diversity of countries. Furthermore, "being social" has been proved to be tremendously important for the relationship quality between users in the online environment instead of being a troll; this behavior might be assessed in a diversity of manners [6]–[8]. Trolls are individuals who generate the character of truly desiring to be part of a group at issue, delivering pseudo-sincere intents or containing professing, but their real intents are to make disruption and/or to activate or aggravate disagreement in online discourse [9], [10]. In some cases, a troll is described as the anti-social behavior feature of gaming groups and side discussion forums such as 4chan [11], [12]. Additionally, the trolls are part of a government-sponsored effort to spread deliberately false information about public health issues, such as vaccines and social justice, and political candidates [13]–[16]. Besides, people who generate or deceive arguments to sow mistrust and manipulate public opinion are trolling on sociopolitical platforms. [17]. Consequently, trolling is a malicious form of online behavior that disrupts public discourse and interactions, irritates conversational users, and attracts users to inconclusive conflict [18].

Over time, trolls' tactics shifted from employing aggressive speech to spreading false news, rumors, and misinformation [19]. In addition, trolling became an organized effort of trolling farms or user society from sporadic action of individuals, which organize the propagation of the content of the troll on social networking environments (SNEs), typically using the social automated program to increase by generating misunderstandings of consensus on a separate subject [20]. Government-sponsored trolling farms use an automated
program and fake accounts to generate troll content and manipulate community thoughts, making it challenging to detect and eliminate trolling in SNEs. These environments rely on moderators to ban/mute trolls and flag/delete troll content to circumvent this issue. Nevertheless, this manual remedy results in scalability, delayed activities, and subjective judgment [21], [22]. To address the limitations that are inherent in manual approaches, some detection software designed to operate automatically has been developed. The core functionality of these programs is founded on natural language processing (NLP) algorithms and artificial intelligence (AI) approaches, such as machine learning and deep learning, respectively. To put it another way, due to the importance of automatic troll detection, the researchers have implemented a number of well-known methods, such as sentimental and linguistic analysis [21], [23]–[26], social network analysis [27], [28], and metadata analysis [21], [29].

In this study, several model architectures using both NLP and deep learning techniques are developed to perform automatic detection of troll tweets. More specifically, a total of nine deep learning-based troll tweet detection architectures are implemented and compared, with three models for each of the three word embedding models: BERT, ELMo, and GloVe. The performance of each architecture is measured in terms of precision, recall, F1 score, area under the ROC curve (AUC), and classification accuracy. Troll tweet detection was improved across all architectures that use BERT models, according to the results of the experiments. Besides, it shows that GRU classifiers have the best AUC score for detecting troll tweet messages in an ELMo-based architecture.

The organization of this article is as follows. Section 2 presents relevant research on detecting troll tweets. The theoretical foundation of the various research methods used in this study is briefly explained in section 3. Section 4 provides an explanation of the proposed model architecture. Section 5 discusses the experiments that were performed and their results. Concluding observations are made in the final section.

2. Related Work

The troll detection task is very difficult since everyone can send trolling messages on the Internet [10]. The author of the [30], which analyzes the antecedents of personal trolling behavior, discovered that the use of both the debate and mood context can describe the behavior of trolling better than a person’s trolling history. According to a trolling study [14], which links to the Russian troll farm Internet Research Agency (IRA) and is sponsored by the state, a minor part of original trolling content for example hashtags, posts, memes, etc. is generated by trolls and for about a specific time of interest such as the Brexit referendum, they deeply involve in re-tweeting about a specific point in time of interest. Dissimilar state-sponsored approaches are found in a study into the activity of trolling during the US election in 2016. Trolls from Iran are the opposite of Trump whereas Internet Research Agency trolls are pro-Trump. To avoid easy detection of troll content on the Internet, these two troll farms are not consistent over time in spreading trolling content [31]. To permit constructive general discourse and a high degree of participation, one of the aims of communication-based social networking platforms is to detect trolling actions, which makes them unwilling to instantly externalize individuals showing the behaviors of trolling to keep away from perceptions of censorship and extreme control [21]. When considering the risk to the reliability of public view and the integrity of the online conversation generated by various
motives, forms, and types of trolls, the necessitate for automatic recognition of trolling in communication-based social networking platforms is obvious.

To address the trolling issue, many studies have been carried out to design troll detection algorithms. A sentiment and linguistic analysis are one of the methods to detect trolls since the contents of trolls are naturally textual in general and consist of a form of social media messages or comments that include provocative and virulent language. To distinguish trolling activities, a domain-adapting sentimental analysis is designed in [25]. In this study, linguistic, thread-level, user-level, and post-level features are used and encouraging results of detecting trolling about 70 percent of the time are obtained in an online forum. In addition, measurement of emotions and sentiments of messages aid to detect trolling on Twitter. To distinguish Twitter trolling more than 76 percent of the time, the authors of the [10] evaluate the abusive textual contents with other meta-data from troll messages. In [32], the authors reach an accuracy of 88.95%, recall of 93.12%, and precision of 86.88% by using an out-of-box sentimental analysis tool named VADER along with syntactic, lexical, and aggression analyzers when the Kaggle Twitter cyber-trolls dataset is used [23], [33]. The authors of [24] used a troll detection algorithm that analyzes the IRA Twitter writing style trolling that investigates the moral, sentimental, and emotional modifications illustrating an F1 score of 94%. A great deal of metadata about people that research also used towards detecting and learning the behavior of the trolls is provided by online and social media platforms. In [29] the authors generated two classifiers, enhancing the sentimental analysis that has data regarding the publication time of the troll messages (weekend, workday, nighttime, work time). The first classifier attempts to detect sponsored trolls which make manipulation the view of the users and the second one attempts to detect individualistic trolling which incites anger and gives offense to people. Parallel results of around 82% accuracy are obtained by two detection methods. In another study in [27], the authors obtain a precision of 51% by using the metadata in a greater communication-based social networking investigation of the Slashdot Zoo website when troll users are discriminated against. To distinguish real people from state-sponsored trolls and individual trolls, the authors of [28] implemented a different method to analyze the activity and effects of the users on Twitter. The authors of [34] propose an approach to detect troll tweets in English and Russian by reducing the detection process to authorship verification. The proposed method is evaluated using monolingual, cross-lingual, and bilingual training scenarios with multiple machine learning algorithms, such as deep learning. With AUCs of 0.877 and 0.82 for tweet classification in English and Russian test sets, respectively, bilingual learning achieves the best results.

3. Background

3.1. Word Embedding Methods

3.1.1. GloVe

To generate word embeddings, GloVe (Global Vectors for Word Representation) is an alternative technique [35]. The GloVe word representations are a mixture of prediction approaches and count-based approaches such as Word2Vec. The GloVe stands for Global
Vectors [36], which represents the idea: that the technique employs global data from a corpus to acquire word vector representations [37]. To calculate the relation between context C and word W: N(W, C), the easiest count-based technique employs co-occurrence counts. To build the loss function, Global Vectors utilizes such counts as well. GloVe is based upon matrix factorization methods on the word-context matrix. A big matrix of co-occurrence information is built, and each ‘word’ (the rows) is counted, and how often this word in some ‘context’ (the columns) is seen in a huge corpus. Generally, the corpus is scanned in the following way: context terms are searched for in a specified range set by a window size for every term before the term and a window size after the term. In addition to this, a smaller amount of weight is given to more distant words. The amount of contexts is huge because this is fundamentally combinatorial in size. Consequently, to produce a lower-dimensional matrix at that time, this matrix is factorized, where for each word each row produces an embedding vector. It is generally completed by making a reconstruction loss minimum. Discovering the lower-dimensional word embeddings that can describe many variations in the high-dimensional information is the aim of this loss.

3.1.2. ELMo

A novel sort of deep contextualized word representation that models both (1) complex features of word usage (e.g., semantics and syntax), and (2) how these employs vary across linguistic contexts such as modeling polysemy [38] are introduced. ELMo is different from conventional types of word embedding in that every token is assigned a representation that is the complete input sentence function. In this model, a bi-directional LSTM that is trained by a coupled language model objective on a huge amount of text corpora generates vectors. Because of this, it is called ELMo (Embeddings from Language Models) representations. ELMo representations, in contrast to earlier methods for contextualizing word vectors [39], [40], are deep, in the sense that they are a function of the entire internal layers of the bidirectional language model. For each end task, more particularly, a linear vector combination is stacked above every input word, which increases performance over just utilizing the top LSTM layer learned. In this way, the internal states combination permits for very rich word representations. It has been proven that context-dependent aspects of word meaning (for example they can be utilized without alteration to make well performance on supervised word sense disambiguation tasks) are captured by the higher-level LSTM states whereas lower-level states model aspects of syntax (for example the part-of-speech tagging is done by them). Simultaneously revealing these signals is extremely beneficial, as it enables the learned models to select the semi-supervision techniques that are most advantageous for each final task.

3.1.3. BERT

Google introduces Bidirectional Encoder Representations from Transformers (BERT), which is a language transformation method [41]. The BERT learns the deep representation of texts by taking into account both, right and left contexts. For that reason, it is called deeply bidirectional. BERT is a technique that can be utilized for training a general-purpose language model on very big corpora and then these models can be used for the Natural Language Processing (NLP) tasks [42]–[44]. Consequently, pre-training and fine-tuning are
two different phases used when implementing BERT. is trained on unlabeled data is used to train during the pre-training step. After that, the pre-trained parameters are used to initialize the framework and then the framework will fine-tune for certain NLP tasks. Enormous computation resources are required to fine-tune the BERT framework. The identical framework is used by BERT in various tasks. The Transformers is used to build the BERT framework [45]. The framework has two variations, which are the BERT-large and BERT-base models. 16 self-attention heads, a hidden size of 1024, and 24 Transformer blocks are included in BERT-large and hidden size of 768, 12 self-attention heads and 12 Transformer blocks are included in the BERT-base model.

3.2. Encoder Methods

3.2.1. CNN

A particular sort of feed-forward neural network is represented by CNN (Convolutional neural networks), which includes neurons of a layer performing a convolution operation [42]. CNN networks are inspired by the ocular nerve function. Following a specific size of the convolution filter, neurons react to the activations of the input of the nearby neurons named filters as well. Shifting the convolutional filter over the entire set of values is the process of convolution. In such a situation, input values and the product of the convolution filter are shown by the convolution process (see Figure 1) [46]. To decrease the number of outputs, and computational complexity and to prohibit overfitting in networks, pooling layers are used in CNN. Immediately after the convolution layers, the sampling layers are generally processed since when the convolution filters are shifting throughout the single inputs, the duplicated values are produced. The pooling layers are used to remove surplus data. The global-max pooling layers where we select between the global max-pooling layer and the global-average pooling layer are used in this study. Such layers perform on the same principle as the conventional max-pooling layer. The dissimilarity is that average pooling is calculated for the complete input, not just a specific field.

3.2.2. RNN

Recurrent neural network (RNN) is a traditional sort of neural network with recurring connections, which permits a type of memory [47]. In addition, these make it appropriate for sequential prediction tasks that contain arbitrary spatiotemporal dimensions. Therefore, various NLP problems employ the RNN architecture to analyze sequence tokens by concerning a sentence interpretation.

Long Short-Term Memory (LSTM) [48] is a sort of RNN. To solve the vanishing gradient problem in RNN, LSTM contains a more complicated framework. Therefore, the Long Short-Term Memory will make certain that long-term context and data are preserved in any sequence-based problems (see Figure 2). In comparison with other neural network architectures, instead of interconnected neurons, the LSTM structure includes memory blocks that are linked in layers. To deal with the data flow, the block output, and the block state, the LSTM uses gateways in the block. Gateways can decide which information should be kept
and which is significant in a sequence. LSTM has four gates, which perform the functions below (see Figure 3): i) Input gate—this controls the entry of data into the memory block. ii) Cell state—long-term data is stored in this gate. iii) Forget gate—this gate can learn what data will be preserved and what data will be removed and then perform actions according to that. iv) Output gate—this gate will make a decision on what action to carry out on the output according to the memory unit and the input. In addition to that, the authors of [49] aim to resolve the vanishing gradient problem illustrated in Section 3.3 in the publication, where they present a type of RNN model called Gated Recurrent Unit (GRU). Gated Recurrent Unit can be thought of as a type of the Long Short-Term Memory model since the design of both networks is done in a parallel way. To deal with the vanishing gradient problem, GRU uses reset and update gates. The reset gate aids the network to decide, how much data will be deleted from the prior time steps and the update gate assists the network to decide, how much of the prior data from the earlier time steps will be required in the next steps.

### 3.2.3. Transformer

Nowadays, transformer models are very common techniques used to resolve a variety of natural language problems for example language understanding, summarization, and question answering but especially it is effectively employed in text categorization problems [50]. Similar to RNN structure, sequential information for example natural language tasks such as text summarization and translation [51] is dealt with by Transformers. On the other hand, in comparison with recurrent neural networks, processing the sequential data in order is not needed in Transformers. For instance, if we have a natural language sentence, the start of this sentence is not required to be processed before the end in the Transformer model. Because of this characteristic, the Transformers permit considerably more parallelization in comparison with recurrent neural networks and as a result decrease training times [45]. In NLP, Transformers have quickly turned out to be the architecture of choice [52] by substituting previous RNN structures for instance LSTM.

A transformer is an encoder-decoder model. In the encoder part, some encoding layers repetitively process the input data one after the other. In the decoder part, some decoding layers perform the identical thing to the encoder output. Processing the input to produce encodings, which contain data regarding which input parts are related to each other, is the purpose of every encoder layer. The encoders pass their encodings set to the following encoder layers as an input.

To produce output sequences, each decoder will take the entire encodings and process them via their incorporated contextual data. In other words, each decoder layer performs the opposite of the encoder [53]. To accomplish this, an attentional mechanism is used by each decoder and encoder layer, which for every input, measures every other input relevance and takes data from them accordingly to generate the output [54]. In addition, all decoder layers contain an extra attentional mechanism taking data from the prior decoder’s outputs before the decoder layer takes data from the encodings. For extra operating of the outputs, a feed-forward neural network is included in both the decoder and encoder layers. Additionally, they have layer normalization phases and residual connections [54].

### 3.2.4. Evaluation Metrics
There are four categories for the results of binary classification [55]: i) True Positive (TP): Classifying positive samples correctly; ii) False Negative (FN): Classifying a positive sample inaccurately; iii) False Positive (FP): Classifying a negative sample inaccurately; VI) True Negative (TN): Classifying negative samples correctly; as a starting point, following metrics can be obtained via the previous ones as well.

For classification, the most fundamental evaluation measure is accuracy. The fraction of the accurate predictions is represented by the evaluation measure accuracy in the network. The calculation of accuracy is as follows:

\[
\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}
\]

The fraction of accurate predictions between the entire predictions with the positive label is the precision, which represents how many of the positively predicted samples are true positive samples. The calculation of the standard precision is shown below:

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

Between all positive samples, the fraction of accurate predictions is the recall, which indicates how many positive samples are positively classified. The calculation of the standard recall is shown below:

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

The harmonic average of recall and precision represents F-measure, which can be calculated as follow:

\[
F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}
\]

Here, the weight of recall and precision is equal. For that reason, this is called an F1 measure as well.

Receiver Operating Characteristics (ROC): ROC curve is selected as a standard criterion to test classifiers if there is a dataset with a class imbalance issue [56], [57]. To evaluate a classifier's performance, the area under the ROC curve (AUC) metric is commonly utilized as a de facto measure if a dataset has a class imbalance issue. The AUC is a metric, which calculates how much a positive class sample is higher ranked than a negative class sample randomly when the samples are sorted with classification probabilities.

4. The proposed model architecture

The interactions between individuals and computers are studied by NLP by employing the human’s natural language. To process and analyze natural language data, it is very popular to utilize NLP methods for instance machine translation models, language models, and so on. The representations of the text, which should be learned first, are the first thing to be accomplished to figure out the text. Self-supervised learning (SSL) is being broadly employed to obtain the pre-trained representations of the document with the present a sequence of the text from a huge corpus for instance the prediction of some hidden part of the text utilizing some other part of their nearby text. Without costly labeling efforts, thus, the
architectures can learn via supervision from large document information. If all words or sub-words are treated as a different token, the pre-trained representations of each word can be obtained on a massive corpus using some methods such as GloVe, ELMo, BERT, etc. Each word representation is shown as a vector when the pre-training process is finished. In some techniques, on the other hand, the same word vector representation is always used for the same word the same regardless of the context such as GloVe and FastText. For example, if we have two different sentences: “He went to the bank to deposit some money” and “He went to the bank to sit down.”. In this case, the same word representation is used for the word “bank” in the first and second sentences. The word vector representation of the same token is adapted to diverse contexts by various pre-trained methods. But in the sentences above, the word “bank” has various senses depending upon where it is used. Unlike GloVe and FastText, some methods provide contextualized word-embeddings based upon the context in which the word is used to both obtain the meaning of the word in that context in addition to other context-based data [58]. Moreover, BERT is a much deeper self-supervised method based upon the transformer encoder. In this study, different pre-trained word representations will be used to test the troll dataset.

Pre-trained word embeddings can be fed to many deep neural network models for a variety of NLP tasks. As can be seen from the Figure, the many pre-trained word embedding methods can be used to feed a range of deep neural network models for diverse NLP problems.

Detecting troll tweets is one of the fundamental tasks of natural language processing (NLP), which involves evaluating the category of a sentence containing a bunch of words. Troll tweet detection mainly consists of two parts:

1) Word embedding, for example, uses the classic BERT, ELMo, and GloVe, which have been popular in recent years, to convert words into word vectors.
2) The encoder performs the next feature extraction of word embedding inputs, such as the commonly used CNN, GRU and Transformer models that have been very popular in recent years, which are more high-level semantic features that can extract words.
This study focuses on these two parts, testing the effects of text classification in different word vectors with different Encoder.

5. Experiments

5.1. Dataset Description

Experiments are conducted using a troll dataset compiled by Miao et al. [34]. This troll dataset consists of 18,514 trolls collected from 2018 to 19. Each message is labeled with one of the following two classes: Troll (9257 messages), and non-troll (9257 messages). In addition, duplicates and retweets are filtered out to prevent over-fitting in this dataset [34]. The same number of normal tweet messages is gathered at random to generate a balanced dataset for binary classification. The same hashtags as those of the troll messages are utilized by the list of random messages. The ultimate troll dataset includes 18,514 English tweet messages, after adding tweet messages from real accounts. The dataset is divided into three parts so that it contains 12,959 training samples, 1,851 validation samples, and 3,702 test samples. The sample number distribution reveals that this data set is well-balanced.

5.2. Experimental setup

The following word embedding vector types were chosen for this study: GloVe utilizes the word vector Glove.6B.300d. ELMo employs the context word vector of the pre-trained model supplied by AllenNLP; the dimensions of the LSTM's implicit and output layers are 2048/256. BERT employs the Bert-large-uncased word vector supplied by Transformers, which has a 12-layer transformer structure and a 512-byte output layer size.

Following are descriptions of three distinct encoder methods. CNN utilizes convolutional neural networks for feature extraction. RNN extracts features utilizing a GRU network. The transformer was combined with Positional Embedding and Transformer Encoder to extract features. The implementation of PyTorch was utilized for this procedure.

5.3. Performance Evaluation

The proposed models have been evaluated in terms of F1 measure and classification accuracy on the dataset explained above in the experiment. If the categories of the dataset are balanced, the accuracy metric is one of the best evaluation metrics in terms of performance [59]. But the troll dataset used in this study has a balance over its classes. Although the classifier's performance can be validated by the accuracy metric only, high values for the F1 score along with a high AUC score have been tried to accomplish in addition to the good accuracy score in the experiment.

It is vital to use decent embedding methods to preprocess the specified dataset to achieve excellent troll detection performance [60]. Nowadays, the word embedding methods generated by the BERT, GloVe, and ELMo algorithms are good selections.
Table 1. Performance results in terms of accuracy, precision, recall, F1, and AUC

| Pretrained Embedding Method | Encoder Methods | Accuracy | Precision | Recall | F1    | AUC  |
|-----------------------------|-----------------|----------|-----------|--------|-------|------|
| BERT                        | CNN             | 0.838    | 0.849     | 0.822  | 0.835 | 0.915|
|                             | GRU             | 0.845    | **0.865** | 0.817  | 0.840 | 0.924|
|                             | Transformer     | **0.856**| 0.825     | **0.904**| **0.863**| **0.909**|
| ELMo                        | CNN             | 0.842    | 0.833     | 0.854  | 0.844 | 0.916|
|                             | GRU             | 0.855    | 0.839     | 0.878  | 0.859 | **0.929**|
|                             | Transformer     | 0.843    | 0.827     | 0.867  | 0.847 | 0.917|
| GloVe                       | CNN             | 0.743    | **0.743** | 0.741  | 0.742 | 0.818|
|                             | GRU             | 0.767    | 0.760     | 0.779  | 0.769 | 0.831|
|                             | Transformer     | **0.732**| 0.749     | **0.698**| **0.723**| **0.806**|

*Bold values indicate the best-performed method and bold italics indicate worst performed method.

The performance results of architectures in terms of accuracy, precision, recall, and F1 metrics are shown in Table 1. As shown in Table 1, the F1-score for BERT ranged from 0.835 to 0.863, whereas the ELMo-based architectures ranged from 0.843 to 0.858. CNN architecture, which makes use of word embeddings from the BERT model ranged from 0.822 to 0.915. In general, all ELMo and BERT-based architectures outperform the GloVe-based architectures by about 13%. Because the same word vector representation is always used for the same word regardless of the context in GloVe. Unlike GloVe, BERT and ELMo provide contextualized word embeddings based upon the context in which the word is used to obtain both the meaning of the word in that context and other context-based data. Additionally, BERT is a much deeper self-supervised method based upon the transformer encoder.

Table 2. Comparison of the AUC results with the previous study

| Ref. | Method                  | AUC  |
|------|-------------------------|------|
| [34] | SVM (Stylometric)       | 0.836|
|      | Logistic Regression (N-grams) | 0.861|
|      | RNN+CNN (Raw Text)      | 0.824|

This study

| Method                  | AUC  |
|-------------------------|------|
| BERT+CNN                | 0.915|
| BERT+GRU                | 0.924|
| BERT+Transformer        | 0.909|
| ELMo+CNN                | 0.916|
| ELMo+GRU                | **0.929**|
| ELMo+Transformer        | 0.917|
| GloVe+CNN               | 0.818|
| GloVe+GRU               | 0.831|
| GloVe+Transformer       | **0.806**|
From the evaluation results illustrated in Table 1 and Table 2 for troll tweet detection tasks, the next inferences are drawn:

- The usage of BERT and ELMo word embedding with all state-of-the-art deep neural learning frameworks performs improved results for troll detection tasks in comparison to GloVe word embeddings.
- GRU deep learning architecture in combination with any word-embedding methods ELMo, BERT and Glove perform the best result than other present benchmarks deep learning-based troll detection models.
- BERT word embedding models perform all other state-of-the-art troll detection architectures by giving the maximum detection F1, Precision, Recall, and Accuracy score Troll detection architecture. On the other hand, the ELMo word embedding method provides the best result in terms of the AUC score.
- In general, the GloVe word embedding model gives the worst results in all other word embedding architectures by giving the minimum detection in terms of F1, Precision, Recall, AUC, and Accuracy score.

6. Conclusion

In recent years, many trolling accounts have emerged to manipulate public opinion on social media. Detecting trolling messages and taking action is vital to maintaining a safe social media environment. Detecting and eliminating trolling is a critical issue for social-networking platforms since businesses, abusers, and nation-state-sponsored troll farms use false and automated accounts. Such farms can use a variety of polarized topics to manipulate public opinion and produce troll content. NLP techniques are used to extract data from social media text, such as Twitter tweets. In many text processing applications, word embedding representation methods, such as BERT, have performed better than prior NLP architectures, offering novel breaks to precisely comprehend and categorize social-networking information for various tasks.

In this study, a total of nine different model architectures for troll tweet detection that is based on deep learning have been implemented and compared. There are three different models for each of the BERT, ELMo, and GloVe word embedding models. We evaluate the performance of each architecture by calculating its classification accuracy, F1 score, AUC, and precision. Based on the findings of the experiments, it is possible to draw the conclusion that the majority of the architectural designs that make use of the Bidirectional Encoder Representations from Transformers models have, in general, improved their capability of detecting trolling messages. In terms of area under the curve (AUC), a customized ELMo-based architecture that employs a Gated Recurrent Unit (GRU) classifier is the most effective architecture for detecting troll messages. In the future, various social-based systems will be able to use the architectures that have been proposed to detect troll messages.

Conflict of interest

The authors declare that they have no conflict of interest.
REFERENCES

[1] M. Tomaiuolo, G. Lombardo, M. Mordonini, S. Cagnoni, and A. Poggi, “A Survey on Troll Detection,” Future Internet, vol. 12, no. 2, p. 31, Feb. 2020, doi: 10.3390/fi12020031.

[2] C. Meske and S. Stieglitz, “Adoption and use of social media in small and medium-sized enterprises,” in Lecture Notes in Business Information Processing, 2013, vol. 151, pp. 61–75, doi: 10.1007/978-3-642-38774-6_5.

[3] C. Meske, I. Junglas, and S. Stieglitz, “Explaining the emergence of hedonic motivations in enterprise social networks and their impact on sustainable user engagement: A four-drive perspective,” Journal of Enterprise Information Management, vol. 32, no. 3, pp. 436–456, Jun. 2019, doi: 10.1108/JEIM-08-2018-0177.

[4] C. Meske, K. Wilms, and S. Stieglitz, “Enterprise Social Networks as Digital Infrastructures - Understanding the Utilitarian Value of Social Media at the Workplace,” Information Systems Management, vol. 36, no. 4, pp. 350–367, Oct. 2019, doi: 10.1080/10580530.2019.1652448.

[5] G. D. Larosiliere, L. D. Carter, and C. Meske, “How does the world connect? Exploring the global diffusion of social network sites,” J Assoc Inf Sci Technol, vol. 68, no. 8, pp. 1875–1885, Aug. 2017, doi: 10.1002/asi.23804.

[6] A. Chinnov, P. Kerschke, C. Meske, S. Stieglitz, and H. Trautmann, “An Overview of Topic Discovery in Twitter Communication through Social Media Analytics,” 2015.

[7] S. Stieglitz, C. Meske, B. Ross, and M. Mirbabaie, “Going Back in Time to Predict the Future - The Complex Role of the Data Collection Period in Social Media Analytics,” Information Systems Frontiers, vol. 22, no. 2, pp. 395–409, Apr. 2020, doi: 10.1007/s10796-018-9867-2.

[8] C. Meske, I. Junglas, J. Schneider, and R. Jaakonmäki, “How Social is Your Social Network? Toward A Measurement,” in ICIS 2019 Proceedings, 2019, pp. 1–9.

[9] C. Hardaker, “‘I refuse to respond to this obvious troll’: an overview of responses to (perceived) trolling,” Corpora, vol. 10, no. 2, pp. 201–229, 2015, doi: 10.3366/cor.2015.0074.

[10] P. Jachim, F. Sharevski, and P. Treebridge, “TrollHunter [Evader]: Automated Detection [Evasion] of Twitter Trolls During the COVID-19 Pandemic,” vol. 20, 2020, doi: 10.1145/nnnnnn.nnnn.

[11] B. Kirman, C. Lineham, and S. Lawson, “Exploring Mischief and Mayhem in Social Computing or: How We Learned to Stop Worrying andLove the Trolls,” Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts - CHI EA ’12. ACM Press, 2012.

[12] M. Samory and E. Peserico, “Sizing Up the Troll: A Quantitative Characterization of Moderator-Identified Trolling in an Online Forum,” in Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 2017, doi: 10.1145/3025453.3026007.

[13] D. A. Broniatowski et al., “Weaponized Health Communication: Twitter Bots and Russian Trolls Amplify the Vaccine Debate,” Am J Public Health, vol. 108, no. 10, pp. 1378–1384, Oct. 2018, doi: 10.2105/AJPH.2018.304567.

[14] C. Llewellyn, L. Cram, A. Favero, and R. L. Hill, “Russian Troll Hunting in a Brexit Twitter Archive,” Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries. ACM, 2018.
[15] S. Zannettou, T. Caulfield, W. Setzer, M. Sirivianos, G. Stringhini, and J. Blackburn, “Who Let The Trolls Out?,” Proceedings of the 10th ACM Conference on Web Science - WebSci ’19. ACM Press, 2019.

[16] L. G. Stewart, A. Arif, and K. Starbird, “Examining Trolls and Polarization with a Retweet Network,” 2018.

[17] A. Badawy, K. Lerman, and E. Ferrara, “Who Falls for Online Political Manipulation?,” Companion Proceedings of The 2019 World Wide Web Conference. ACM, 2019.

[18] B. A. Coles and M. West, “Trolling the trolls: Online forum users constructions of the nature and properties of trolling,” Computers in Human Behavior, vol. 60, pp. 233–244, 2016, doi: 10.1016/j.chb.2016.02.070.

[19] Y. Benkler, R. Farris, and H. Roberts, Network Propaganda: Manipulation, Disinformation, and Radicalization in American Politics, vol. 1. Oxford University Press, 2018.

[20] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race,” 26th International World Wide Web Conference 2017, WWW 2017 Companion, pp. 963–972, Jan. 2017, doi: 10.1145/3041021.3055135.

[21] P. Fornacciari, M. Mordonini, A. Poggi, L. Sani, and M. Tomaiuolo, “A holistic system for troll detection on Twitter,” Computers in Human Behavior, vol. 89, pp. 258–268, Dec. 2018, doi: 10.1016/j.chb.2018.08.008.

[22] I. Jeun, Y. Lee, and D. Won, “A Practical Study on Advanced Persistent Threats,” in Computer Applications for Security, Control and System Engineering: International Conferences, SecTech, CA, CES3 2012, Held in Conjunction with GST 2012, Jeju Island, Korea, November 28-December 2, 2012. Proceedings, T. Kim, A. Stoica, W. Fang, T. Vasilakos, J. G. Villalba, K. P. Arnett, M. K. Khan, and B.-H. Kang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 144–152.

[23] J. L. C. Capistrano, J. J. P. Suarez, and P. C. Naval, “SALSA: Detection of cybertrolls using sentiment, aggression, lexical and syntactic analysis of tweets,” in ACM International Conference Proceeding Series, Jun. 2019, pp. 1–6, doi: 10.1145/3326467.3326471.

[24] B. Ghanem, D. Buscaldi, and P. Rosso, “TexTrolls: Identifying Russian Trolls on Twitter from a Textual Perspective,” arXiv, Oct. 2019.

[25] C. W. Seah, H. L. Chieu, K. M. A. Chai, L. N. Teow, and L. W. Yeong, “Troll detection by domain-adapting sentiment analysis,” in 2015 18th International Conference on Information Fusion, Fusion 2015, 2015, pp. 792–799.

[26] J. Naim, T. Hossain, F. Tasneem, A. N. Chy, and M. Aono, “Leveraging fusion of sequence tagging models for toxic spans detection,” Neurocomputing, vol. 500, pp. 688–702, Aug. 2022, doi: 10.1016/J.NEUCOM.2022.05.049.

[27] S. Kumar, F. Spezzano, and V. S. Subrahmanian, “Accurately detecting trolls in Slashdot Zoo via decluttering,” 2014.

[28] F. Riquelme and P. González-Cantergiani, “Measuring user influence on Twitter: A survey,” Information Processing and Management, vol. 52, no. 5, pp. 949–975, Sep. 2016, doi: 10.1016/j.ipm.2016.04.003.

[29] T. Mihaylov and P. Nakov, “Hunting for Troll Comments in News Community Forums,” Nov. 2019.
[30] J. Cheng, M. Bernstein, C. Danescu-Niculescu-Mizil, and J. Leskovec, “Anyone can become a troll: Causes of trolling behavior in online discussions,” in Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing, 2017, pp. 1217–1230.

[31] S. Zannettou, T. Caulfield, W. Setzer, M. Sirivianos, G. Stringhini, and J. Blackburn, “Who let the trolls out? towards understanding state-sponsored trolls,” in Proceedings of the 10th acm conference on web science, 2019, pp. 353–362.

[32] C. Hutto and E. Gilbert, “VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text | Proceedings of the International AAAI Conference on Web and Social Media,” Proceedings of the International AAAI Conference on Web and Social Media, 2014. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14550. [Accessed: Jul. 09, 2022]

[33] DataTurks, “Tweets Dataset for Detection of Cyber-Trolls | Kaggle,” 2022. [Online]. Available: https://www.kaggle.com/datasets/dataturks/dataset-for-detection-of-cybertrolls. [Accessed: May 31, 2022]

[34] L. Miao, M. Last, and M. Litvak, “Detecting Troll Tweets in a Bilingual Corpus,” 2020.

[35] H. Mujtaba, “What is Word Embedding | Word2Vec | GloVe.” [Online]. Available: https://www.mygreatlearning.com/blog/word-embedding/. [Accessed: Apr. 12, 2021]

[36] L. Voita, “NLP Tutorials — Part 5: GloVe – Applied Singularity.” [Online]. Available: https://appliedsingularity.com/2022/01/25/nlp-tutorials-part-5-glove/. [Accessed: Jul. 09, 2022]

[37] L. Voita, “Word Embeddings.” [Online]. Available: https://lena-voita.github.io/nlp_course/word_embeddings.html. [Accessed: Jul. 09, 2022]

[38] M. E. Peters et al., “Deep contextualized word representations,” in NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, Feb. 2018, vol. 1, pp. 2227–2237, doi: 10.18653/v1/n18-1202.

[39] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, “Semi-supervised sequence tagging with bidirectional language models,” arXiv preprint arXiv:1705.00108, 2017.

[40] B. McCann, J. Bradbury, C. Xiong, and R. Socher, “Learned in translation: Contextualized word vectors,” Adv Neural Inf Process Syst, vol. 30, 2017.

[41] J. Devlin, M. -W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.

[42] V. Maslej-Krešňáková, M. Sarnovský, P. Butka, and K. Machová, “Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification,” Applied Sciences (Switzerland), vol. 10, no. 23, pp. 1–26, Dec. 2020, doi: 10.3390/app10238631.

[43] J. Bedi and D. Toshniwal, “CitEnergy: A BERT based model to analyse Citizens’ Energy-Tweets,” Sustainable Cities and Society, vol. 80, p. 103706, May 2022, doi: 10.1016/J.SCSC.2022.103706.

[44] B. Zhou et al., “VictimFinder: Harvesting rescue requests in disaster response from social media with BERT,” Computers, Environment and Urban Systems, vol. 95, p. 101824, Jul. 2022, doi: 10.1016/J.COMPENVURBSYS.2022.101824.
A. Vaswani et al., “Attention Is All You Need,” doi: 10.5555/3295222.3295349.

I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

J. Zhou, J. X. Huang, Q. Chen, Q. V. Hu, T. Wang, and L. He, “Deep learning for aspect-level sentiment classification: Survey, vision, and challenges,” IEEE Access, vol. 7. Institute of Electrical and Electronics Engineers Inc., pp. 78454–78483, 2019.

S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

K. Cho et al., “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” arXiv preprint arXiv:1406.1078, 2014.

C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to fine-tune bert for text classification?,” in China national conference on Chinese computational linguistics, 2019, pp. 194–206.

“Transformer (machine learning model) - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)#cite_note:-0-1. [Accessed: Apr. 12, 2021]

T. Wolf et al., “Transformers: State-of-the-art natural language processing,” in Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, 2020, pp. 38–45.

“Sequence Modeling with Neural Networks (Part 2): Attention Models – Indico Data.” [Online]. Available: https://indicodata.ai/blog/sequence-modeling-neural-networks-part2-attention-models/. [Accessed: May 31, 2022]

J. Alammar, “The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time.” [Online]. Available: http://jalammar.github.io/illustrated-transformer/. [Accessed: May 31, 2022]

T. Fawcett, “ROC Graphs: Notes and Practical Considerations for Researchers,” HP Labs Tech Report HPL-2003-4, pp. 1–38, 2004, doi: 10.1.1.10.9777.

H. He and Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications. Wiley, 2013.

A. Tharwat, “Classification assessment methods,” Applied Computing and Informatics, pp. 1–13, Aug. 2018, doi: 10.1016/j.aci.2018.08.003.

J. Alammar, “The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) – Jay Alammar – Visualizing machine learning one concept at a time.” [Online]. Available: https://jalammar.github.io/illustrated-bert/. [Accessed: Jul. 07, 2022]

K. N. Singh, S. D. Devi, H. M. Devi, and A. K. Mahanta, “A novel approach for dimension reduction using word embedding: An enhanced text classification approach,” International Journal of Information Management Data Insights, vol. 2, no. 1, p. 100061, Apr. 2022, doi: 10.1016/J.JJIMEI.2022.100061.

B. Guo, C. Zhang, J. Liu, and X. Ma, “Improving text classification with weighted word embeddings via a multi-channel TextCNN model,” Neurocomputing, vol. 363, pp. 366–374, Oct. 2019, doi: 10.1016/J.NEUCOM.2019.07.052.