Deep Fake Recognition in Tweets Using Text Augmentation, Word Embeddings and Deep Learning

Senait G. Tesfagergish[1], Robertas Damaševičius[1(✉)], and Jurgita Kapočiūtė-Dzikienė[2]

[1] Kaunas University of Technology, 51368 Kaunas, Lithuania
robertas.damasevicius@ktu.lt
[2] Vytautas Magnus University, 44404 Kaunas, Lithuania

Abstract. Spreading of automatically generated clickbaits, fake news, and fake reviews undermines the veracity of the internet as a credible source of information. We investigate the problem of recognizing automatically generated short texts by exploring different Deep Learning models. To improve the classification results, we use text augmentation techniques and classifier hyperparameter optimization. For word embedding and vectorization we use Glove and RoBERTa. We compare the performance of dense neural network, convolutional neural network, gated recurrent network, and hierarchical attention network. The experiments on the TweepFake dataset achieved an 89.7% accuracy.

Keywords: Deep fake · Fake news detection · Text classification · Natural language processing · Social media analytics · Deep learning

1 Introduction

High-quality texts generated Artificial Intelligence (AI)- algorithms (aka deep fakes) have swarmed social networks and messaging services and started to influence real-world events. For example, bots on social media use auto-generated messages to subvert the democratic voting process, promote violence, and defy the values of democratic countries [1]. A recent but rapidly spreading phenomenon of "fake news", i.e., viral social media posts viral posts that are made to look like real news reports [2], have overshadowed the political discourse on major recent political events such as Brexit [3] and US Presidential elections of 2016 [4]. Other controversial and widely discussed topics of interest such as vaccination [5] and most recently, the COVID-19 pandemic [6] have gone beyond misinformation and may have caused literal deaths. The phenomenon of "fake news" has been aggravated by computer-aided tools such as generative chatbots [7] and news headline generators [8]. In particular, auto-generated clickbait, i.e., social media posts or online articles aimed at increasing the network traffic to the actual article or user page [9]. Another example is that of fake reviews, which aim to improve or disrupt the popularity of some product on review aggregators or e-commerce websites [10] can cause tangible financial damage both to product producers as well as consumers, which follow the

O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12954, pp. 523–538, 2021.
https://doi.org/10.1007/978-3-030-86979-3_37

advice of distorted consumer recommenders [11]. Finding fake reviews may involve solving related problems such as intent detection [12], phishing detection [13], topic propagation [14], and topic recommendation [15]. These tools employ natural language processing (NLP) techniques and AI methods such as chatbots [16], artificial neural networks (ANN), supervised learning, and deep learning techniques such as generative adversarial networks (GAN) [17] to create realistic texts that can promote disinformation and sow discord among the society. The latter has caused considerable debate on the dark side of AI and its future trends and threats to our political institutions and social development [18, 19]. The complexity of this effort requires multidisciplinary effort from the research community [20] as it deals with multifaceted aspects of technology, social structures, and psychology.

Recent efforts of recognizing fake content on the internet have focused on supervised learning techniques [21]. Both traditional machine learning techniques such as Support Vector Machine (SVM), Decision Trees (DT), Random Forest (RF) and Hidden Markov Models (HMM) as well as move recent deep learning models such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) are applied [22]. Specifically, Ren and Ji [23] suggested a gated recurrent NN to recognize deceptive spam. They employed word embeddings that are learned using the continuous bag-of-words (CBOW) approach to obtain global semantic representation that can alleviate the problem of scarce data. Hajek et al. [24] adopted n-grams, a Skip-Gram Word2Vec model to create word embeddings from a consumer review corpus, which is combined with and various lexicon-based emotion features. For classification, they used a hybrid "Network in Network" architecture that allowed them to capture the complex features in the high-dimensional representation space. Zheng et al. [25] suggested a clickbait convolution-al neural network (CBCNN) that used pre-trained Word2Vec to capture the semantics of the clickbait headlines and used various kernels to find the characteristics of the headlines. Ajao et al. [26] used unidirectional Long Short-Term Memory (LSTM) with Convolutional Neural Network (CNN) model for classification of fake vs genuine tweets. The initial unidirectional LSTM layer is used to keep information from the previous context, without storing the context information. Asghar et al. [27] used the Bidirectional LSTM-CNN model to classify tweets into rumors and non-rumors, achieving 86.12% accuracy. In this model, CNN layer receives input from BiLSTM with contextual information, thus improving over the Ajao et al. [26] approach. Fang et al. [28] built a model for determining the authenticity of the news based only on their content by using a combination of CNN with a self multi-head attention mechanism. Ghanem et al. [29] suggested an LSTM neural network model that is emotionally enriched to recognize false news and clickbait. Jwa et al. [30] apply the Bidirectional Encoder Representations from Transformers (BERT) model to identify fake news by checking the relationship between headline and the main body of news. Kaliyar et al. [31] propose a deep CNN (FNDNet) for fake news detection. The network learns the features for fake news classification via multiple layers of the network achieving an accuracy of 98.36%. Liu & Wu et al. [32] proposed a deep neural network to detect fake news early using a custom feature extractor, a position-aware attention mechanism, and a multi-region mean-pooling mechanism to perform feature aggregation. Umer et al. [33] proposed a hybrid model that combines CNN with LSTM, in combination with dimensionality reduction using

Principal Component Analysis (PCA) and Chi-Square. The approach achieved 97.8% accuracy on the fake news challenge dataset.

In the context of small data [34], i.e., the unavailability of sufficient data for efficient training of neural networks, especially deep network architectures with a large number of trainable parameters, the problem is especially relevant in under-resourced languages, for which there are a limited amount of texts or only small corpora available. The problem in the NLP domain may be solved through a variety of techniques such as using pre-trained networks and applying transfer learning [35] or increasing dataset size via data augmentation [36]. In the latter cases, the size of the dataset can be increased by creating new samples via thesaurus substitution, wod2vec substitution, and addition of meaningless words [37], by masking and replacing words in original samples [38], replacing high-frequency words with low-frequency words source [39], replacing the word with a word sampled from the frequency distribution of the dictionary [40], using a bi-directional language transformation model to generate a replacement word [41], or using a soft probability distribution to change the word representation [42].

Summarizing, the classical machine learning methods require tedious feature engineering by experts to assure high performance. Deep learning solves this problem by using word embedding and deep neural networks, but are not very good if the semantics of words changes over time.

The novelty and contribution of this paper are as follows: We developed a deep learning model to detect the fake tweets generated by bots by combining the advantages of CNN and the hierarchical attention network. The best model obtained high performance in fake news detection and achieved an accuracy rate of 89.7% on the TweepFake [43] benchmark dataset. We compare the performance of classical machine learning and deep learning methods applied to this problem. Our results showed that the proposed classifier can achieve better performance when comparing the accuracy with previous works.

2 Methods

2.1 Problem Definition

We address the problem of classifying tweets into artificially generated (i.e., created by a bot) and human-created. To discriminate between the tweets, the task is treated as a binary classification problem. We define the training dataset as $D = \{d_1, d_2, d_3, \ldots d_n\} \in \mathcal{R}^{zxm}$, while each row $d_i \in R_n$ is a data sample and each column $C_i \in \mathcal{R}^z$ is a label for train dataset $y \in \{0, 1\}$, if 1, then it is fake (generated), else it is real (created by a human). We aim to develop and analyze deep learning models, which can learn from available data to accurately classify fake and real tweets.

2.2 Text Preprocessing

Before translating the text into a vector form, it goes through a pre-processing process. The pre-processing process consists of several parts: filtering (removing punctuation marks, hyphens and extra spaces), replacing upper cases with lower cases, spelling correction, stopword removal and stemming (replacing a word with its semantic base).

For spelling correction, we use Python 3 Spelling Corrector. For stopword removal, we remove 421 stopwords for English text included in the Fox [44] list. Tokenization is performed by using whitespaces and non-alphanumerical characters as delimiters. We follow the following set of heuristic rules as suggested in [45]:

(1) replace characters ! " # $ % & *< = > ? @ \ | ~ with spaces;
(2) remove any of " . : ;" if followed by a space:,
(3) remove the brackets () [];
(4) remove the quotation marks;
(5) remove apostrophes and slashes if followed by space.

After tokenization, we use stemming to further normalize the morphological variants of the base word. In this paper, we use the S stemmer [46], which removes only a few common word endings, and it is less aggressive as other stemmer.

2.3 Dataset Augmentation

Data augmentation is advantageous for low-resource NLP tasks [47]. We used the techniques described in [48] as follows: (1) random replacement of words in tweets with their synonyms; (2) random insertion of synonyms of words (using wordnet) in a tweet. (3) random swap of words in the tweet. (4) random deletion for each word in the tweet with a probability p. The preliminary checking ensures that the word can be replaced, word for replacement is not a determiner and it does have synonyms.

2.4 Vector Representation

Feature selection is very important for document classification problems [49]. We used GloVe [50], a model that combines the features of a singular decomposition and methods Word2Vec. The first step is to construct a co-occurrence matrix X from the training dataset. The meaning of X_{ij} indicates how often the word j occurs in the context of the word i. To quantify the semantic similarity between words i and j the ratio of the probabilities of their joint occurrence in the context is used, where w_i, w_j are word vectors, $\tilde{}\,\widehat{w}_k$ is a context vector.

$$F\left(w_i, w_j, \widehat{w}_k\right) = \frac{P_{ik}}{P_{jk}} = \frac{X_{ik}/\sum_m X_{im}}{X_{jk}/\sum_n X_{jn}} \tag{1}$$

The semantic proximity of the vectors obtained is determined by their scalar product. The GloVe model learns vectors in such a way that their scalar product approached the logarithm of the probability of the appearance of words in the training set. To reduce the weight of the joint occurrences of words that are rare (carry less information) or do not occur at all, as well as reduce weight for too frequent joint appearances. We adopt a weighted least squares regression model as target learning function (loss function) (1), where w_i is the vector of the main words, $\widehat{w}_j$ is the context vector, b_i and $\widehat{b}_j$ are scalar

values of deviations for the main word and context word, respectively, V is the size of the dictionary:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \widehat{w}_j + b_i + \widehat{b}_j - logX_{ij}\right)^2 \tag{2}$$

Here $f()$ is the weight function.

Another vector representation we use is a Robustly Optimized BERT Pretraining Approach (RoBERTa) [51], which uses BERT-based dynamic masking, a transformer model to masks and predict tokens, which extracts contextual features of texts.

2.5 Dense Neural Network

A dense neural network is a network made of regular deeply connected layers. It is a common and frequently used neural network type. The architecture is such that all the neurons, in one layer are linked to the neurons in the next layer.

Let the output of the dense neural network be known $y(t)$ at the input $X(t)$, where $X(t)$ is a vector with components $(x_1, x_2, \ldots, x_n)$, t is the number of the sequence value, $t = \overline{1, T}$ (T is predetermined). To find model parameters $w = (w_0, w_1, \ldots w_m)$ and $V_k = (V_{1k}, V_{2k}, \ldots V_{nk})$, h_k, $k = \overline{1, m}$ such that the model output $F(X, V, w)$ and the real output of the MLP $y(t)$ would be as close as possible. The relationship between the input and output of a two-layer perceptron is established by the following relationships:

$$Z_k = \sigma(V_{1k}x_1 + V_{2k}x_2 + \ldots V_{nk}x_n - h_k), k = \overline{1, m} \tag{3}$$

$$y = \sigma(w_1 Z_1 + w_2 Z_2 + \ldots w_m Z_m + w_0) \tag{4}$$

Here we used the three-layer network architecture as shown in Fig. 1, with 768 neurons in the input layer (corresponding to the number of RoBERTa features) and 10 neurons in the intermediary layer. Network training occurs by applying a gradient descent algorithm (such as error backpropagation) similar to a single-layer perceptron.

2.6 Convolutional Network

A convolutional neural network is usually an alternation of convolutional layers, subsidizing layers, and with fully connected output layers. All these layers can be placed in any order. In the convolutional layer, neurons that use the same weights are put into feature maps, and each neuron is linked with a portion of the neurons of the previous layer. When calculating the network, each neuron performs a convolution of a certain area of the previous layer. A layer in which each neuron is connected to all neurons at the previous level, with each connection having its weight. Unlike fully connected, in the convolutional layer a neuron is connected only with some neurons of the previous level, that is, the convolutional layer is similar to the convolution operation, where only a small weight matrix (convolution kernel) is used. Layers of this type perform dimensionality reduction. the method of selecting the maximum element is used - the entire feature map is divided into cells, from which the maximum value is selected. The dropout layer is a

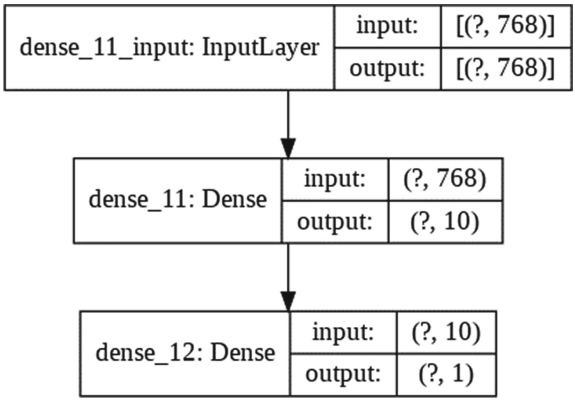

Fig. 1. Schematic representation of the densely connected neural network

way to combat overfitting in neural networks. Dropout regulation consists of changing the structure of the network: each neuron is ejected with a certain probability p.

The architecture we used is presented in Fig. 2. The model has one embedding layer, one convolutional layer, global max pooling, and a dropout layer.

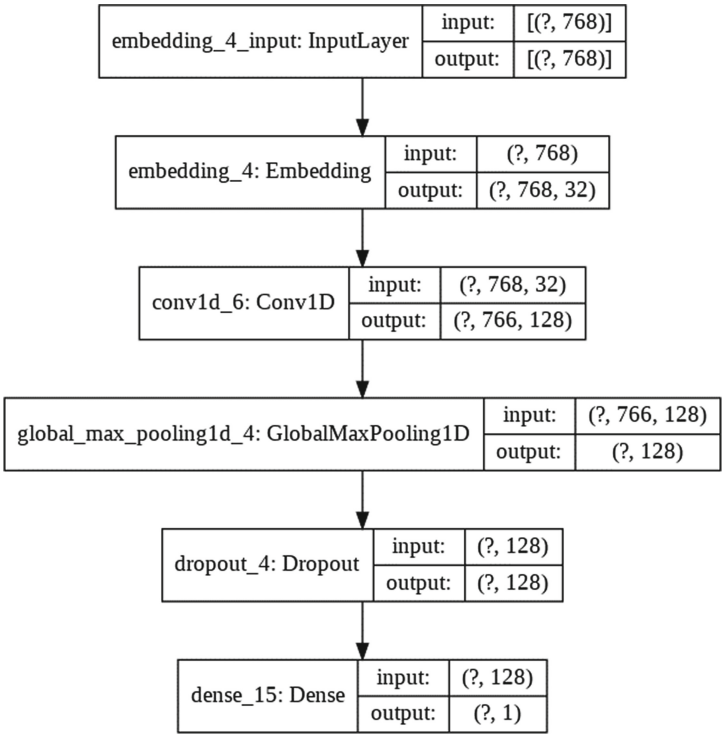

Fig. 2. Schematic representation of CNN

2.7 Recurrent Neural Network and Gated Recurrent Unit

The Recurrent Neural Network (RNN), with the help of the hidden layer h, the model can save information about the previous input signals, and at the end of the data sequence, carry out the sentiment classification. The extensions of RNN are the GRU (Gated Recurrent Unit) [52] and LSTM (Long-Short Term Memory) [53] models. In them, unlike the usual RNN, each neuron is a memory cell, the contents of which can be updated or discarded.

The memory cells GRU and LSTM are shown schematically in Fig. 3. In the GRU network, the OUT value is determined by the activation of the reset r and update z filters. LSTM uses a more complex computation scheme, applying three filters: input filter i, forget filter f, and output filter o. A controlled recurrent neuron contains one gate less than a long short-term memory cell. The update gate determines the amount of information obtained from the past state. The reset gate works like the forget valve in the long short-term memory cell. As GRU only has two a reset gate and an update gate, its training speed is faster than other RNNs.

Based on previous output h_{t-1} and current input x_t, a reset gate is used to determine which part of information should be reset, Eq. (5), while an update gate is used to refresh the output of the GRU h_t, Eq. (8). The hidden layer is calculated according to Eq. (7). The latest output can be calculated according to Eq. (8). The gates z_t and r_t, and parameters, W_z, W_r and W, of GRU are updated in the training process.

$$z_t = \sigma(W_z \bullet [h_{t-1}, x_t]), \tag{5}$$

$$r_t = \sigma(W_r \bullet [h_{t-1}, x_t]), \tag{6}$$

$$h_t{}' = \tanh(W \bullet [r_t * h_{t-1}, x_t]), \tag{7}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t{}'. \tag{8}$$

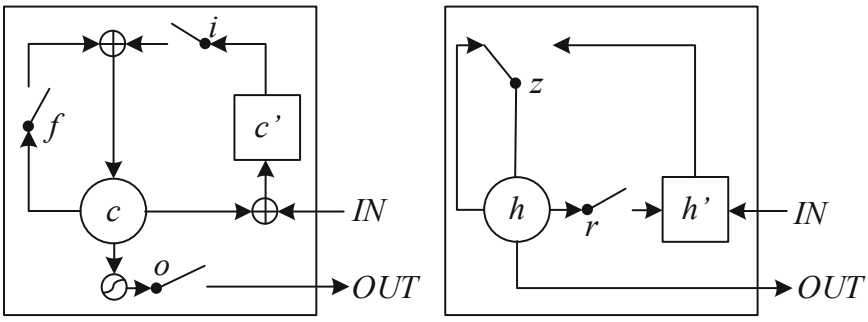

Fig. 3. Schematic representation of LSTM (left) and GRU (right)

2.8 Hierarchical Attention Network

The hierarchical attention network (HAN) model [54] aims to capture two basic insights a hierarchical structure of a text on different levels (words form sentences, sentences form a message), as well as document representation. The model uses a word encoder, i.e., a bidirectional GRU, along with a word attention mechanism to encode each sentence into a vector representation. These sentence representations are passed through a sentence encoder with a sentence attention mechanism resulting in a document vector representation. This final representation is passed to a fully connected (FC) layer with the activation function for prediction as follows (Fig. 4):

$$x_{it} = W_e w_{it}, \tag{9}$$

$$\overrightarrow{h}_{it} = \overrightarrow{GRU}(x_{it}), \tag{10}$$

$$\overleftarrow{h}_{it} = \overleftarrow{GRU}(x_{it}). \tag{11}$$

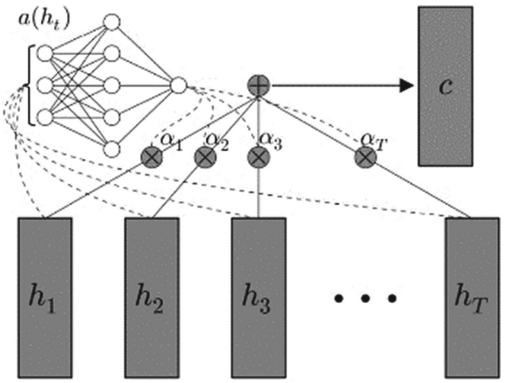

Fig. 4. Hierarchical attention network

2.9 Network Training Optimization

Optimization of neural network hyper-parameters, which rule how the network operates and governs its accuracy and validity, is still an unsolved problem. Here we adopt the Exponential Adaptive Gradients (EAG) optimization [55]. EAG optimization (Algorithm 1) measures the past gradients exponentially more and consecutively reduces adaptivity of the second moment to the latest gradients when network parameters are close to the best values.

3 Dataset and Results

3.1 Dataset and Exploratory Analysis

We used TweepFake [43], a Twitter deep fake dataset available from Kaggle (https://www.kaggle.com/mtesconi/twitter-deep-fake-text). The dataset is balanced and contains 25,836 tweets (half human and half bots generated, 23647 in the training dataset, and 2922 in the testing dataset), which were randomly extracted from the human and corresponding imitating bot account pairs. The bots use various NLP generation techniques, i.e., Markov Chains, RNN, GPT-2, and LSTM. Both bot and human-created texts have a similar distribution of the number of words (Fig. 5), while humans tend to use longer words (Fig. 6). On the other hand, bots use more stop words than humans as indicated by the stop word ratio (Fig. 7). The figures show histograms with corresponding probability distributed functions (PDFs) approximated using kernel density estimation. Since PDFs are approximated, their tails in some cases exceed the minimum and maximum values in the dataset.

Algorithm 1. Exponential Adaptive Gradients

Input: $x \in F, \{\alpha_t\}_{t=1}^{T}, (\beta_1,\beta_2) = (0.9,10^{-4})$

Output: x_{t+1}

Initialize $m_0 = 0, v_0 = 0$

FOR $t = 1$ to T DO

$g_t = \nabla f_t(x_t)$

$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$

$v_t = (1+\beta_2)v_{t-1} + \beta_2 \cdot g_t^2$

$\hat{v}_t = v_t/[(1+\beta_2)^t - 1]$

$V_t = diag(\hat{v}_t)$

$x_{t+1} = \prod_{F,\sqrt{V_t}}(x_t - \alpha_t m_t/\sqrt{V_t})$

ENDFOR

3.2 Evaluation of Performance

The true labels are compared against the predicted labels and the true positive (TP), false positive (FP), true negative (TN), and false-negative (FN) values are calculated. Recall, Precision, Accuracy, Error Rate, and F-score are calculated as follows:

$$FPR = \frac{\sum_{i=1}^{m}[a(x_i) = +1][y_i = -1]}{\sum_{i=1}^{m}[y_i = -1]} \tag{12}$$

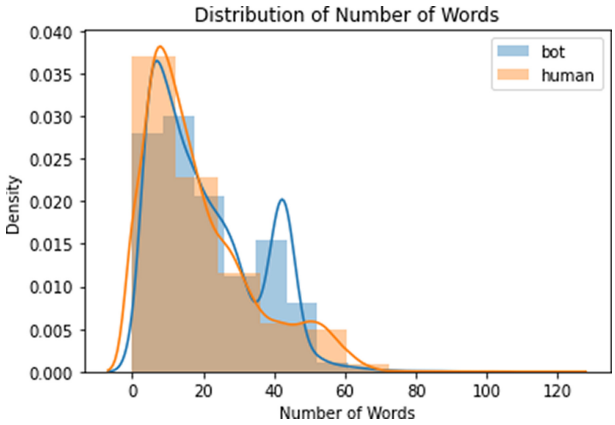

Fig. 5. Distribution of the number of words in the training dataset

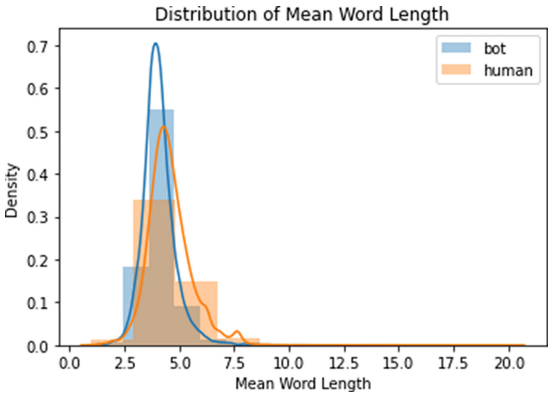

Fig. 6. Mean word length in the training dataset

$$TPR = \frac{\sum_{i=1}^{m}[a(x_i) = +1][y_i = +1]}{\sum_{i=1}^{m}[y_i = +1]} \tag{13}$$

$$FNR = \frac{\sum_{i=1}^{m}[a(x_i) = -1][y_i = +1]}{\sum_{i=1}^{m}[y_i = +1]} \tag{14}$$

Here $a(x)$ is classifier with inputs $X^m = (x_1, \ldots, x_m)$, and $(y_1, \ldots, y_m)$ are outputs. Precision, Recall and Accuracy are calculated as follows:

$$Precision = \frac{TPR}{TPR + FPR} \tag{15}$$

$$Recall = \frac{TPR}{TPR + FNR} \tag{16}$$

$$Accuracy = \frac{\sum_{i}^{p} N_i}{T} \tag{17}$$

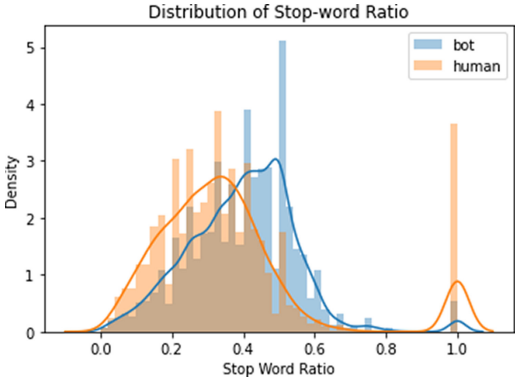

Fig. 7. Stop word ratio in the training dataset

Here FPR is False Positive Rate, TPR is True Positive Rate, FNR is False Negative Rate, N_i is the sum of correctly classified data samples, and T is the total number of data samples. The F1 measure is a harmonic mean between precision and recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{18}$$

If the classifier allows you to estimate the probability of an object belonging to the desired class, then a qualitative assessment of the curve, constructed for different values of this probability, we also assume AUC.

3.3 Settings

We have implemented our model using Python on Google Colab environment. For text augmentation, we use EDA [48], which adopted yje techniques described in Sect. 2.3. For embedding, we use Glove and RoBERTa. To obtain word embeddings, we use Flair [56], in which we have selected a language model enabling fine-tuning on BERT, and the training the GRU RNN along with the classification layer included language model fine-tuning. To tune the hyper-parameters of our deep learning network, we used Hyperopt [57]. We tuned learning rate, batch size, the number of convolutional and fully connected (FC) layers, the size of the kernels and number of filters in CNN, and the number of the neurons in the FC layers, and the number of LSTM cells. To mitigate the effect of overfitting, we used the early stopping approach.

3.4 Results

To compare, as baselines we use simple classical machine learning models: term frequency–inverse document frequency (TF-IDF) with logistic regression (LR) classifier and a bag-of-words (BoW) with logistic regression. We also implemented a simple dense neural network, with only 2 layers. The results are presented in Table 1.

Table 1. Summary of the classification performance using 10-fold cross-validation. Best values are boldened.

Model	F1	Pr	Re	AUC	Acc
BoW + LR	0.686	0.613	0.780	0.759	0.673
TF-IDF + LR	0.681	0.568	0.853	0.753	0.635
Glove + DN	0.703	0.599	0.862	0.789	0.691
RoBERTa + DN	0.801	0.645	0.832	0.821	0.811
RoBERTA + CNN	0.816	0.657	0.845	0.834	0.820
RoBERTa + LSTM	0.835	0.690	0.864	0.852	0.854
RoBERTA + HAN	**0.855**	**0.71**	**0.923**	**0.913**	**0.897**

Classification performance of the best deep network model is given in Fig. 8.

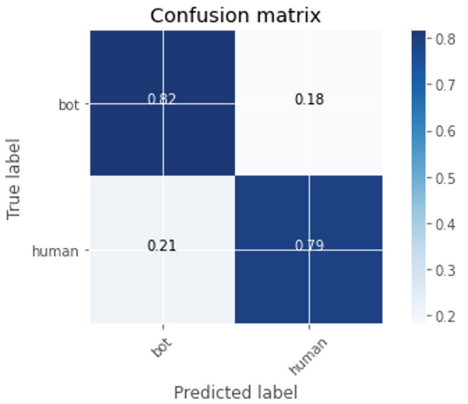

Fig. 8. Confusion matrix of the classification results (RoBERTa + HAN)

We compare our results with those of Fagni et al. [43], which is as far as we know the only work that has reported the results on this dataset. They also used BERT-type transformers (BERT, DISTILBERT, ROBERTA) and XLNET, however, they did not employ text augmentation. The comparison of results is presented in Table 2. Surprisingly, Fagni et al. [43] achieved better precision, whereas we achieved higher recal.. By using the text augmentation technique combined with RoBERTa, we were able to achieve similar results in terms of accuracy (89.7%), which underlines the importance of text augmentation for text classification using short texts and small datasets.

Table 2. Comparison of the proposed model performance with the results achieved by Fagni et al. [43]. Best values are boldened.

Model	F1	Pr	Re	Acc
BERT [43]	0.892	0.884	0.892	0.891
DISTILBERT [43]	0.888	0.882	0.888	0.887
ROBERTA [43]	**0.897**	0.891	0.897	0.896
XLNET [43]	0.882	**0.922**	0.882	0.877
RoBERTA + HAN (proposed)	0.855	0.71	**0.923**	**0.897**

4 Conclusions

In this paper we have addressed the problem of recognizing automatically generated tweets by exploring different neural network models. To improve the classification results, we used the text augmentation techniques. To obtain features from the text we used word embedding and vectorization. We compared the performance of dense neural network, convolutional neural network, gated recurrent network, and hierarchical attention network on the TweepFake dataset. The best results were achieved by RoBERTA + HAN architecture, which reached an accuracy of 89.7%.

Acknowledgment. Future work will aim on the improvement of discussed architectures specifically focusing on the problem of the small dataset.

References

1. Paterson, T., Hanley, L.: Political warfare in the digital age: cyber subversion, information operations and 'deep fakes.' Aust. J. Int. Aff. **74**(4), 439–454 (2020)
2. Tandoc, E.C., Lim, Z.W., Ling, R.: Defining "Fake news": a typology of scholarly definitions. Digit. Journal. **6**(2), 137–153 (2018)
3. Bastos, M.T., Mercea, D.: The brexit botnet and user-generated hyperpartisan news. Soc. Sci. Comput. Rev. **37**(1), 38–54 (2019)
4. Assibong, P.A., Wogu, I.A.P., Sholarin, M.A., Misra, S., Damasevičius, R., Sharma, N.: The politics of artificial intelligence behaviour and human rights violation issues in the 2016 US presidential elections: An appraisal. In: Sharma, N., Chakrabarti, A., Balas, V.E. (eds.) Data Management, Analytics and Innovation. AISC, vol. 1016, pp. 295–309. Springer, Singapore (2020). https://doi.org/10.1007/978-981-13-9364-8_22
5. Wang, Y., McKee, M., Torbica, A., Stuckler, D.: Systematic literature review on the spread of health-related misinformation on social media. Soc. Sci. Med. **240**, 112552 (2019)
6. Shimizu, K.: 2019-nCoV, fake news, and racism. Lancet **395**(10225), 685–686 (2020)
7. Kapočiute-Dzikiene, J.: A domain-specific generative chatbot trained from little data. Appl. Sci. **10**(7), 2221 (2020)
8. Dandekar, A., Zen, R.A.M., Bressan, S.: Generating fake but realistic headlines using deep neural networks. In: Benslimane, D., Damiani, E., Grosky, W.I., Hameurlain, A., Sheth, A., Wagner, R.R. (eds.) DEXA 2017. LNCS, vol. 10439, pp. 427–440. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64471-4_34

9. Chakraborty, A., Paranjape, B., Kakarla, S., Ganguly, N.: Stop clickbait: detecting and preventing clickbaits in online news media. IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. Mining, ASONAM **2016**, 9–16 (2016)
10. Malbon, J.: Taking fake online consumer reviews seriously. J. Consumer Policy **36**(2), 139–157 (2013)
11. Ji, Z., Pi, H., Wei, W., Xiong, B., Wozniak, M., Damasevicius, R.: Recommendation based on review texts and social communities: a hybrid model. Access **7**, 40416–40427 (2019)
12. Kapočiūtė-Dzikienė, J., Balodis, K., Skadiņš, R.: Intent detection problem solving via automatic DNN hyperparameter optimization. Appl. Sci. **10**(21), 1–21 (2020)
13. Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., Woźniak, M.: Accurate and fast URL phishing detector: a convolutional neural network approach. Comput. Netw. **178**, 107275 (2020). https://doi.org/10.1016/j.comnet.2020.107275
14. Zhang, B., Wei, W., Wang, W., Li, Y., Cui, H., Si, Q.: Modeling topic propagation on heterogeneous online social networks. In: 2018 IEEE 18th International Conference on Software Quality, Reliability, and Security Companion, QRS-C 2018, pp. 641–642 (2018)
15. Lin, J., et al.: Attention-based high-order feature interactions to enhance the recommender system for web-based knowledge-sharing service. In: Huang, Z., Beek, W., Wang, H., Zhou, R., Zhang, Y. (eds.) WISE 2020. LNCS, vol. 12342, pp. 461–473. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62005-9_33
16. Omoregbe, N.A.I., Ndaman, I.O., Misra, S., Abayomi-Alli, O.O., Damaševičius, R.: text messaging-based medical diagnosis using natural language processing and fuzzy logic. J. Healthcare Eng. **2020**, 1–14 (2020)
17. Li, C., Su, Y., Liu, W.: Text-to-text generative adversarial networks. International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, pp. 1–7 (2018)
18. Wogu, I.A., Misra, S., Assibong, P., Adewumi, A., Damasevicius, R., Maskeliunas, R.: A critical review of the politics of artificial intelligent machines, alienation and the existential risk threat to America's labour force. In: Gervasi, O., et al. (eds.) ICCSA 2018. LNCS, vol. 10963, pp. 217–232. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95171-3_18
19. Wogu, I.A.P., Misra, S., Roland-Otaru, C.O., Udoh, O.D., Awogu-Maduagwu, E., Damasevicius, R.: Human rights' issues and media/communication theories in the wake of artificial intelligence technologies: The fate of electorates in twenty-first-century american politics. In: Advances in Electrical and Computer Technologies, pp. 319-333 (2020)
20. Lazer, D.M.J., et al.: The science of fake news: addressing fake news requires a multidisciplinary effort. Science **359**(6380), 1094–1096 (2018)
21. Reis, J.C.S., Correia, A., Murai, F., Veloso, A., Benevenuto, F., Cambria, E.: Supervised learning for fake news detection. IEEE Intell. Syst. **34**(2), 76–81 (2019)
22. Bondielli, A., Marcelloni, F.: A survey on fake news and rumour detection techniques. Inf. Sci. **497**, 38–55 (2019)
23. Ren, Y., Ji, D.: Neural networks for deceptive opinion spam detection: an empirical study. Inf. Sci. **385**, 213–224 (2017)
24. Hajek, P., Barushka, A., Munk, M.: Fake consumer review detection using deep neural networks integrating word embeddings and emotion mining. Neural Comput. Appl. **32**(23), 17259–17274 (2020). https://doi.org/10.1007/s00521-020-04757-2
25. Zheng, H., Chen, J., Yao, X., Sangaiah, A.K., Jiang, Y., Zhao, C.: Clickbait convolutional neural network. Symmetry **10**(5), 138 (2018)
26. Ajao, O., Bhowmik, D., Zargari, S.: Fake news identification on twitter with hybrid CNN and rnn models. In: 9th International Conference on Social Media and Society, pp. 226–230 (2018)
27. Asghar, M.Z., Habib, A., Habib, A., Khan, A., Ali, R., Khattak, A.: Exploring deep neural networks for rumor detection. J. Ambient. Intell. Humaniz. Comput. **12**(4), 4315–4333 (2019). https://doi.org/10.1007/s12652-019-01527-4

28. Fang, Y., Gao, J., Huang, C., Peng, H., Wu, R.: Self multi-head attention-based convolutional neural networks for fake news detection. PLoS ONE **14**(9), e0222713 (2019)
29. Ghanem, B., Rosso, P., Rangel, F.: An emotional analysis of false information in social media and news articles. ACM Trans. Internet Technol. **20**(2), 19 (2020)
30. Jwa, H., Oh, D., Park, K., Kang, J.M., Lim, H.: exBAKE: Automatic fake news detection model based on bidirectional encoder representations from transformers (BERT). Appl. Sci. **9**(19), 4062 (2019)
31. Kaliyar, R.K., Goswami, A., Narang, P., Sinha, S.: FNDNet – A deep convolutional neural network for fake news detection. Cogn. Syst. Res. **61**, 32–44 (2020)
32. Liu, Y., Wu, Y.B.: FNED: A deep network for fake news early detection on social media. ACM Trans. Inf. Syst. **38**(3), 25 (2020)
33. Umer, M., Imtiaz, Z., Ullah, S., Mehmood, A., Choi, G.S., On, B.: Fake news stance detection using deep learning architecture (CNN-LSTM). Access **8**, 156695–156706 (2020)
34. Yao, H., Jia, X., Kumar, V., Li, Z.: Learning with small data. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 3539–3540 (2020)
35. Molina, M.Á., Asencio-Cortés, G., Riquelme, J.C., Martínez-Álvarez, F.: A preliminary study on deep transfer learning applied to image classification for small datasets. In: 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020), pp. 741–750 (2021)
36. Moreno-Barea, F.J., Jerez, J.M., Franco, L.: Improving classification accuracy using da-ta augmentation on small data sets. Expert Syst. Appl. **161**, 113696 (2020)
37. Sun, X., He, J.: A novel approach to generate a large scale of supervised data for short text sentiment analysis. Multimedia Tools Appl. **79**(9–10), 5439–5459 (2018). https://doi.org/10.1007/s11042-018-5748-4
38. Park, D., Ahn, C.W.: Self-supervised contextual data augmentation for natural language processing. Symmetry **11**(11), 1393 (2019)
39. Fadaee, M., Bisazza, A., Monz, C.: Data augmentation for low-resource neural machine translation. arXiv:1705.00440 (2017)
40. Xie, Z., Wang, S.I., Li, J., Lévy, D., Nie, A., Jurafsky, D., Ng, A.Y.: Data noising as smoothing in neural network language models. arXiv:1703.02573 (2017)
41. Kobayashi, S.: Contextual augmentation: Data augmentation by words with paradigmatic relations. arXiv:1805.06201 (2018)
42. Gao, F., et al.: Soft contextual data augmentation for neural machine translation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 5539–5544 (2019).
43. Fagni, T., Falchi, F., Gambini, M., Martella, A., Tesconi, M.: TweepFake: About detecting deepfake tweets. PLOS ONE **16**(5), e0251415 (2021)
44. Fox, C.: A stop list for general text. ACM SIGIR forum **24**(1–2), 19–21 (1989)
45. Jiang, J., Zhai, C.: An empirical study of tokenization strategies for biomedical information retrieval. Inf. Retrieval **10**, 341–363 (2007)
46. Harman, D.: How effective is suffixing? J. Am. Soc. Inf. Sci. **42**(1), 7–15 (1991)
47. Li, Y., Li, X., Yang, Y., Dong, R.: A diverse data augmentation strategy for low-resource neural machine translation. Information **11**(5), 255 (2020)
48. Wei, J.W., Zou, K.: EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 6382–6388 (2019)
49. Nasir, I.M., et al.: Pearson correlation-based feature selection for document classification using balanced training. Sensors **20**(23), 6793 (2020)
50. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)

51. Liu, Y., et al.: RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 (2019)
52. Cho, K., et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078 (2014)
53. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
54. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 (2014).
55. Ragab, M.G., et al.: A novel one-dimensional cnn with exponential adaptive gradients for air pollution index prediction. Sustainability **12**, 10090 (2020)
56. Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., Vollgraf, R.: FLAIR: an Easy-to-Use Framework for State-of-the-Art NLP. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, pp. 54–59 (2019)
57. Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D.: Hyperopt: a python library for model selection and hyperparameter optimization. Comput. Sci. Discov. **8**(1), 014008 (2015)