Research Article

Self-Information Loss Compensation Learning for Machine-Generated Text Detection

Weikuan Wang and **Ao Feng**

Chengdu University of Information Technology, Chengdu, China

Correspondence should be addressed to Ao Feng; fengao@cuit.edu.cn

Received 6 November 2020; Revised 17 December 2020; Accepted 8 February 2021; Published 19 February 2021

Academic Editor: Hongchuan Yu

Copyright © 2021 Weikuan Wang and Ao Feng. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The technology of automatic text generation by machine has always been an important task in natural language processing, but the low-quality text generated by the machine seriously affects the user experience due to poor readability and fuzzy effective information. The machine-generated text detection method based on traditional machine learning relies on a large number of artificial features with detection rules. The general method of text classification based on deep learning tends to the orientation of text topics, but logical information between texts sequences is not well utilized. For this problem, we propose an end-to-end model which uses the text sequences self-information to compensate for the information loss in the modeling process, to learn the logical information between the text sequences for machine-generated text detection. This is a text classification task. We experiment on a Chinese question and answer the dataset collected from a biomedical social media, which includes human-written text and machine-generated text. The result shows that our method is effective and exceeds most baseline models.

1. Introduction

Have you ever encountered such a situation: when you ask a question on social media, the answers seem to be substantial and the topic is consistent with the question, but after reading carefully, you may find that the confusing logic makes it hard to read with the worthless contents. When searching for symptoms on the Internet, you cannot find the useful information quickly, and sometimes, you maybe panic or be misled to take medicine because of some answers. Such text is likely to be machine-generated text.

Machine-generated text [1–3] is the sentence or article automatically generates by machine with input information given by humans [4]. For different input information, machine-generated text can include text-to-text [5] generation, meaning-to-text generation, data-to-text [6] generation, and image-to-text [7] generation, etc.

The technology of automatic text generation can be applied to a question and answer system [8], machine translation [9], and automatic text summarization [10]. The development of this technology will realize more intelligent

and natural human-computer interaction. We look forward to the day when computers can write like humans, but generating high-quality text sequences is still a challenge. At present, even if the automatic text generation technology has achieved some internationally influential results [11], it lacks a standard evaluation system and cannot effectively and automatically check the quality of the generated text. Therefore, the text generated by the machine appears on the Internet without the valid inspection. Especially, in the current Internet economy era, many social media or Q&A platforms will reward publishers by calculating the number of article views or answers, which promotes the spread of machine-generated text on the Internet.

Traditional methods use classification or clustering [12] to detect machine-generated text, but this requires a lot of manual operations to extract data features, which can represent certain attributes of the data, and then build machine learning models based on these artificial features. In contrast, deep-learning methods can not only provide feature extraction but also complete classification tasks. General text classification methods include convolutional neural

networks [13] (CNN-based models), recurrent neural networks [14] (RNN-based models), and attention [15] mechanism-based models. These methods obtain the potential category information or emotional information contained in the text by extracting the text representation from the original text.

CNN sets up convolution filters of different sizes and numbers and directly acts on the word embedding matrix to obtain local features by setting the step size and sliding the window. A convolution filter of size n can capture n -gram information. The RNN-based model is a time series model designed to capture the semantic information of the context. LSTM, a variant of RNN, allows memory cells to selectively remember and forget information through a gating system to gain long-term dependence. It can capture global text information and is widely used in natural language processing.

In this paper, we propose a new method to capture the logical information of text sequences to detect machine-generated text with the self-information loss compensation mechanism. We believe that handwritten text has good readability, the logic between word order is in line with daily reading habits, and the sentences are continuous and natural with the same theme. The machine-generated text is based on the prior knowledge of the previous word to calculate the maximum probability of the next word to generate text in turn.

We define this task as a binary classification task, using CNN and BiLSTM to provide complementary information. The basic idea is to make the wrong logical information in the text sequence (such as inverted word order and incoherent sentences) follow the increasing length of text to propagate backward through the time series model.

The specific operation is to first encode the word-embedding matrix with correct potential logical information through a large number of handwritten texts and then use the handwritten text and machine-generated text as input information to model by BiLSTM. The hidden unit of BiLSTM can calculate the weighted combination of all words in the sentence to encode the bidirectional logical relationship of the text. Also, CNN extracts the most useful n -gram information for this relationship. At the same time, in order to eliminate the influence caused by the different positions of the wrong logical information in the text sequence, we perform max pooling on the calculated feature map to obtain the most valuable information.

In order to avoid important information being overwhelmed by other useless information, or discarded by max pooling, we use the global text information encoded by BiLSTM as the self-information of the text to compensate the information loss caused by max pooling. Finally, the obtained combined features are activated for classification. We use the biomedical question answering dataset extracted from a medical social platform to experiment, and the result shows that our model with the self-information loss compensation mechanism performs better than other baseline models in detecting machine-generated text.

2. Related Work

Liu et al. [16] proposed three different information-sharing mechanisms based on RNN to model specific tasks and texts

for multiclassification tasks. Yang et al. [17] proposed a hierarchical attention mechanism network for document classification. They proposed the attention mechanism at the sentence level and document level, so that the model could assign different weights to important contents when constructing documents, and at the same time, the gradient disappearance generated by RNN in capturing sequence information of documents could be alleviated. Zhao et al. [18] proposed a capsule network model based on three stable dynamic routes for text classification. Wang et al. [19] proposed embedding the word of label and input text in the same space, introduced the attention framework, and compared the input text sequence and label as attention to classify text. Bao et al. [20] proposed a SC-Net using semantic coherence for machine-generated spam text detection. In addition, some methods based on data mining and analysis [21–23], weighted word embedding [24–26], keyword extraction [27], and machine learning [28, 29] have been proposed.

3. Model

3.1. Task Definition. This task is a binary classification task with supervised learning. We use handwritten text and machine-generated text as the model input. The input sequence can be expressed as $S = \{s_1, \dots, s_n\}$, where s_n represents a sentence in the input sequence. We need to convert all the words in S into word index $ID = \{id_1, \dots, id_n\}$, where id_n is the index of the word in the word embedding matrix. The word sequence $W = \{w_1, \dots, w_n\}$ will be transformed from the ID sequence in the model. w_n is expressed as a high-dimensional feature vector of a word. At the same time, we need to provide a predefined label $Y = \{y_1, \dots, y_n\}$, where the value of y_n is one of (0, 1), where 0 means machine-generated text and 1 means handwritten text. For the sentence sequence S , the model needs to encode the logical relationship between s_i and s_{i+l} , $l \in (1, \text{length}(S)-1)$, for windows with different sizes, where s_i and s_{i+j} can be expressed as adjacent sentences or interval sentences. Similarly, for the word sequence W , the model needs to encode the logical relationship between w_i and w_{i+l} , $l = \text{size}(\text{window})-1$. This information can be an adjacent relationship or an interval relationship; size (window) is the size of the sliding window. Finally, the model needs to give the category number y of the input text sequence according to the captured combination features.

3.2. Model Description. According to Figure 1, the model has 5 layers. The first layer is used as the embedding layer where we use pretrained word2vec embedding. After obtaining the word embedding, we input it into the following time series model for encoding the logical relationship between word sequences. In order to capture the logical information in bidirectional of the sequence, we use a bidirectional model. The forward model uses $p(w_i|w_1, \dots, w_{i-1})$ as the conditional probability and predicts the current word through the previous word. The reverse model uses $p(w_i|w_{i+1}, \dots, w_N)$ as the conditional probability and predicts the current word through the following words. We use the following formula to maximize the value of the log likelihood function:

FIGURE 1: The overall structure of the model.

$$\sum_{i=1}^N (\log p(w_i|w_1, \dots, w_{i-1}) + \log p(w_i|w_{i+1}, \dots, w_N)). \quad (1)$$

We use LSTM as a time sequence unit to capture sequence logic information across time. In LSTM, memory cells span all LSTM units of the time series model and propagate historical information backwards. The input w_i of each unit and the information h_{i-1} transmitted by the previous unit are used to forget the useless information in the memory cell through the gating system and to update the memory of the newly generated useful information.

In our model, we believe that the potential logical relationship in the reverse sequence is as important as it is in the forward sequence. Therefore, we use BiLSTM to capture the bidirectional logical relationship of words. The hidden unit of the model encodes the potential feature information and then transmits it in the memory cell, and each hidden unit outputs a feature vector encoded by the current time step. BiLSTM will get feature vectors in bidirectional $\vec{h}_t$ and $\overleftarrow{h}_t$. The final output sequence is

$$\begin{aligned} \vec{H} &= \{\vec{h}_1, \vec{h}_2, \dots, \vec{h}_n\} \in R^{n \times \text{hidden_size}}, \\ \overleftarrow{H} &= \{\overleftarrow{h}_1, \overleftarrow{h}_2, \dots, \overleftarrow{h}_n\} \in R^{n \times \text{hidden_size}}, \end{aligned} \quad (2)$$

where n is the length of the sequence. $\overleftarrow{h}_n \in R^{1 \times \text{hidden_size}}$, where hidden_size is the dimension of the hidden unit of the LSTM artificially set. In the experiment, we used two methods of connection and addition for the bidirectional feature maps output by BiLSTM. We connect $\vec{h}_n$ and $\overleftarrow{h}_n$ in the hidden_size dimension to form a new feature map $H \in R^{n \times 2\text{hidden_size}}$.

$$H = \{\vec{h}_1 \oplus \overleftarrow{h}_1, \vec{h}_2 \oplus \overleftarrow{h}_2, \dots, \vec{h}_n \oplus \overleftarrow{h}_n\}. \quad (3)$$

The symbol $\oplus$ means connection, and another method is to add according to the corresponding position and $H' \in R^{n \times \text{hidden_size}}$.

$$H' = \{\vec{h}_1 + \overleftarrow{h}_1, \vec{h}_2 + \overleftarrow{h}_2, \dots, \vec{h}_n + \overleftarrow{h}_n\}. \quad (4)$$

The addition will neutralize part of the information of the feature, so that the main feature is covered by other useless information and is no longer prominent. So, we finally adopted the connection method. We represent

$$\begin{aligned} H &= \{\vec{h}_1 \oplus \overleftarrow{h}_1, \vec{h}_2 \oplus \overleftarrow{h}_2, \dots, \vec{h}_n \oplus \overleftarrow{h}_n\}, \\ H &= \{h_1, \dots, h_n\} \in R^{n \times 2\text{hidden_size}}. \end{aligned} \quad (5)$$

CNN is a cross-space model, which can mix and encode the n -gram information in the window. The position of the logical anomaly in the text sequence is random. When the logic anomaly occurs less or even once (the word order error only exists in one place), the n -gram information encoded by the CNN at this position can be successfully captured by the max pooling layer, as the most important local feature, and then, we connect the results to form a global feature map. However, when there are more local features that need to be captured, for example, the input sequence also contains word position errors, context logic inheritance errors, and topic consistency errors between sentences. We hope that these important information can be used by the model to improve model performance. But, max pooling will discard most of the useful information, which is not what we want. For this reason, we propose a self-information loss compensation mechanism.

The self-information loss compensation mechanism is a method that uses the unfiltered information of the text sequence encoded by the model to compensate for the loss of information in the max pooling layer. We believe that the

hidden state vector $\vec{h}_n$ and $\overleftarrow{h}_n$ output from the last unit of the BiLSTM layer encodes the global information of the text sequence, and we represent them as self-information. We carry out two operations on these two hidden state vectors: (1) we perform max pooling separately and then connect to get the vector $m_1 \in R^{1 \times 2}$, average m_1 to get a scalar, and connect it to the back of H (H is the vector obtained by max pooling after the CNN layer) to achieve global maximum feature compensation. (2) We perform average pooling separately and connect to get the vector $m_2 \in R^{1 \times 2}$, which integrates the word order information and then averages m_2 to get a scalar, which is used to add to each dimension of H to achieve the compensation of word order information. In the Figure 1, we mark this operation with a dashed box. We use $h \in R^{2 \times \text{hidden_size}}$ to represent $\{\vec{h}_n, \overleftarrow{h}_n\}$, and the transition formula is as follows:

$$\begin{aligned} C &= \{c_1, \dots, c_n, c_{n+1}\}, \\ c_i &= \max(f(W_c \odot h_{i:i+k-1})) + \text{comp}(h), \\ c_{n+1} &= \text{mean}(\max(h)) + \text{comp}(h), \\ \text{comp}(h) &= \text{mean}(\text{average}(h)), \end{aligned} \quad (6)$$

where n is convolution filter numbers; $i \in (1, n - k + 1)$. The symbol $+$ and $\odot$ stand for elementwise addition and multiplication. c_i is a scalar. f is a nonlinear function ReLU defined as $f(x) = \max\{0, x\}$, $x \in R$. c_{n+1} is also a scalar that connects directly behind c_n . $\text{comp}(h)$ is the average operation after h has been average pooled.

The last is the fully connected layer; in order to avoid overfitting, we apply the dropout before the fully connected layer. Finally, we get the encoded conditional probability $p_i(\{w_1, \dots, w_n\}|\theta) \in [0, 1]$, $i \in (0, n)$, where n is the number of samples. θ represents the parameter set in the model. For the binary classification task, label $y_i \in (0, 1)$, respectively, represents a negative or positive sample. Finally, we use p_i and y_i to calculate binary cross-entropy loss $L(\theta)$. The target of training is to optimize the binary cross entropy as follows:

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^n [y_i \ln p_i + (1 - y_i) \ln (1 - p_i)], \quad (7)$$

where y_i is the ground truth of sample i and n is the total number of samples in the dataset. We use the method by Kingma and Ba [30] to optimize the loss function. The Adam optimizer is an extension of SGD. It can replace the classic stochastic gradient descent method to update network weights more effectively. It is applied to nonconvex optimization problems and uses momentum and adaptive learning rate to speed up convergence. Finally, if $p(x)$ is less than the threshold θ , we deem x as machine-generated text.

4. Experiments

4.1. Dataset Description. In our model, we used a Chinese question and answer dataset from the biomedical social platform. In Table 1, we explain the dataset.

TABLE 1: Dataset description.

Category	6
Answers	792099
Selected sample	43200
Total token	5101408
Max token	4861
Average token	118
Token in inner set	42680

We divide it into positive samples and negative samples according to the ratio of 1:1 and divide the training set, validation set and test set according to the ratio of 10:1:1. For negative samples, we use three methods to construct a new dataset: random disturbance, translation tool to generate text, and machine writing tool to generate text. Under natural circumstances, handwritten text will also have the problem of reverse order, but such cases are relatively rare. We cannot guarantee whether there is a sample with the original word order error in the positive sample we use, so we process a small amount of data in the positive sample with random disturbances to simulate the behavior of artificial errors. The number of such disturbances is less than that of negative samples. The most important feature of machine-generated text is the error of word order and the logical connection of sentences. We have selected a few examples to say that the way we generate negative samples is effective.

As shown in the Table 2, as sample 1 shows, the machine-generated text contains errors in the drug entity, and such errors may cause significant health risks. The passive subject of the text generated by sample 2 should be a person rather than a fake medicine. In the n-gram information of sample 3, the words “diarrhea” and “harass” occur at the same time, which is an obvious language logic error. The text generated in sample 4 expresses the completely opposite meaning to the original text and may even cause panic in patients without medical knowledge.

4.2. Setting Description. We use the dataset mentioned above to train the model. We use word2vec pretrained in advance as the word embedding model. The model is pretrained on the Chinese Wikipedia dataset, which contains 223 million words. The language model of skip-gram with n-negative sampling is used for training and coding Word, Character Press, and n-gram information. For the input text, we use jieba for word segmentation and filter the repeated words to extract the lookup table used in this experiment. The embedding dimension is 300. The parameters of the time series model and the space model are initialized randomly.

As shown in Figure 2, the sequence length of the dataset is concentrated between 400 and 600. We set the maximum sequence length to 512, truncating the sequence that exceeds the length and fill in 0 to supplement the sequence that is insufficient. The size of the convolution filter is (2, 3, 4), and the number of each filter is 2. The model finally uses the accuracy and F1 score as the evaluation criteria, where the F1 score is defined as $F1 = 2 * (PR / (P + R))$. We set the dropout

TABLE 2: Samples of positive and negative.

Sample 1	
Original	Tannic acid protein yeast powder is recommended to be taken according to the instructions
MGT	It is recommended to take food yeast as directed
Sample 2	
Original	So do not be fooled by fake drugs
MGT	Fake drugs are tricked
Sample3	
Original	Use the medicine properly; otherwise, it will cause loose bowels again
MGT	Take the appropriate medication; otherwise, it will cause the diarrhea to harass
Sample4	
Original	It should never have left a hangover
MGT	You should never leave the hangover

to 0.5 and the learning rate to $5e^{-5}$. We train the model on the GPU to increase the training speed.

4.3. Comparison Model. We use the following text classification model as a baseline model for comparison experiments:

FastText [31]: averaging the input embedding to obtain the text representation

LSTM: the output of the hidden state can be used as a local text feature, and the output of the final state can be expressed as a global text feature

TextCNN [13]: a sliding window is used to capture the text n -gram information, max pooling is used to obtain the local maximum feature, and finally, the text representation is obtained through the fully connected layer

RCNN [32]: capturing the context information of the current location of the encoded local features to obtain text representation

LSTM + Attention [33]: the attention mechanism is used to calculate the contribution of each local vector, and the weighted sum is used to obtain the text representation

Compared with traditional machine learning models, the abovementioned neural networks based on deep learning do not need to manually extract features, and the model can learn features by itself. The abovementioned models can also be subdivided into CNN-based models, RNN-based models, and hybrid models.

5. Result and Analysis

The results of the experiment are shown in Table 3. We believe that longer text sequences contain more logical information, so the model should have better performance on long text sequences.

In the table, you can see that the test results of most models do improve as the text sequence grows, but the results of LSTM decrease as the text sequence grows. We know that LSTM has the characteristic of long-term dependence that historical information can be transmitted backwards with memory cells. According to the results,

single LSTM is more suitable for encoding semantic features. If the features are not captured, the logical anomaly information between text sequences will be covered by other information as the length of the text increases.

FastText directly encodes the global information of the text to achieve similar performance to the CNN-based model.

The CNN-based model can encode n -gram information across spaces, and the pooling layer can extract the logical information captured by the model, so as the text sequence grows, the performance is still stable and slightly improved. The mixed model is obviously better than the single model.

Comparing the LSTM with attention and RCNN models, the results have not been significantly improved with the growth of the sequence. We can think that CNN has a significant effect in the feature capture of this task. Our model is also based on CNN and RNN models and has achieved the best performance in text sequences of different lengths. Compared with RCNN, our model adds a self-information compensation mechanism to compensate for the information loss in the max pooling layer. From the results, it can be seen that this mechanism can indeed improve the performance of the model in this task.

From Table 3, we can find that when the text sequence length is set to 256, compared with the result of the sequence length of 128, except for the CNN model, the performance of other models has a small drop, while the CNN model has a sequence length of 512 in the result, and the performance is also slightly reduced. Only RCNN performance has been increasing steadily, but comparing the results of sequence lengths of 128 and 256, we can find that for sequences within this length range, the model only improves by 0.0003, which is still very small. We guess that there may be less distribution of logical anomaly information between the sequence length 128 and 256, so that the effective information encoded by the model in the early stage is affected by other useless information within this length range. The CNN-based model has a certain degree of anti-interference ability against such a situation, but the impact is still not avoided. When the text sequence length is 512, the result is reduced, which may be a delayed performance. This situation can be explained as that when the text sequence is too long, if sufficient features cannot be provided, then local features are more important than global features.

FIGURE 2: The distribution of the sample length.

TABLE 3: Comparison of results with the F1 score and accuracy.

Sequence length	64		128		256		512	
Models	F1 score	ACC	F1 score	ACC	F1 score	ACC	F1 score	ACC
FastText	0.7775	0.8131	0.7861	0.8200	0.7853	0.8169	0.7822	0.8125
TextCNN	0.7811	0.8067	0.7848	0.8131	0.7905	0.8183	0.7858	0.8128
LSTM	0.8414	0.8544	0.7907	0.8219	0.5923	0.6811	0.0044	0.5003
RCNN	0.8570	0.8650	0.8982	0.9033	0.8977	0.9036	0.9021	0.9061
LSTM + Attention	0.8547	0.8633	0.8841	0.8872	0.8770	0.8819	0.8844	0.8889
Our model	<u>0.8731</u>	<u>0.8792</u>	<u>0.9072</u>	<u>0.9103</u>	<u>0.9003</u>	<u>0.9050</u>	0.9131	0.9158

TABLE 4: Comparison of results with two initialization parameter methods.

Sequence length	64		128		256		512	
Our model	F1 score	ACC	F1 score	ACC	F1 score	ACC	F1 score	ACC
Random initialization parameters	0.8147	0.8283	0.8633	0.8683	0.8640	0.8683	0.8659	0.8708
Pretrained parameters	0.8731	0.8792	0.9072	0.9103	0.9003	0.9050	0.9131	0.9158

In Table 4, we set up a model in which the parameters of the embedding layer are randomly initialized for self-comparison. The result shows that using the pretrained word embedding matrix can improve the performance of the model. Compared with other models, our model still performs better than the single model even if it uses the word embedding matrix with random initialization parameters.

6. Conclusions

The hybrid model based on CNN and RNN can provide complementary information. Compared with the single CNN and RNN model, the hybrid model can achieve better performance in the task of machine-generated text detection. A time series model with a gating system can encode a long sequence of text, but the encoded information is not a single content. It is necessary to extract the required feature maps according to a specific task to avoid being covered by other useless information. Except for the most obvious features, not all other features are useless

information. Therefore, we apply the self-information compensation mechanism to the model to compensate for the information lost when extracting the features. The results show that the mechanism does perform better than the baseline [34].

Data Availability

The original dataset can be found at <https://github.com/Toyhom/Chinese-medical-dialogue-data>.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Sichuan Science and Technology Program (Project nos. 2020YFG0168 and 2019YFG0189) and in part by the Research Innovation Team

Fund from the Department of Education (Award No. 18TD0026), Sichuan Province.

References

- [1] N. Mostafazadeh, N. Chambers, X. He et al., "A corpus and evaluation framework for deeper understanding of commonsense stories," 2016, <https://arxiv.org/abs/1604.01696>.
- [2] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," 2018, <https://arxiv.org/abs/1805.04833>.
- [3] X. Feng, M. Liu, J. Liu, B. Qin, Y. Sun, and T. Liu, "Topic-to-essay generation with neural networks," in *Proceedings of IJCAI*, Stockholm, Sweden, 2018.
- [4] P. Jain, P. Agrawal, A. Mishra, M. Sukhwani, A. Laha, and K. Sankaranarayanan, "Story generation from sequence of independent short descriptions," in *Proceedings of SIGKDD 2017 Workshop on Machine Learning for Creativity*, Halifax, Canada, 2017.
- [5] C. Raffel, N. Shazeer, A. Roberts et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," 2019, <https://arxiv.org/abs/1910.10683>.
- [6] A. Moryossef, Y. Goldberg, and I. Dagan, "Step-by-step: separating planning from realization in neural data-to-text generation," 2019, <https://arxiv.org/abs/1904.03396>.
- [7] K. Li, Y. Zhang, K. Li et al., "Visual semantic reasoning for image-text matching," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4654–4662, Seoul, South Korea, 2019.
- [8] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," in *Proceedings of EMNLP 2017*, Copenhagen, Denmark, 2017.
- [9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of ICLR 2015*, San Diego, CA, USA, 2015.
- [10] Q. Wang, "ACL title and abstract dataset," in *Proceedings of ACL 2018*, Melbourne, Australia, 2018.
- [11] K. M. Yoo, Y. Shin, and S. Lee, "Data augmentation for spoken language understanding via joint variational generation," in *Proceedings of AAAI 2019*, Honolulu, HI, USA, 2019.
- [12] M. W. Berry, *Survey of Text Mining: Clustering, Classification, and Retrieval*, Springer, Berlin, Germany, 2003.
- [13] Y. Kim, "Convolutional neural networks for sentence classification," 2014, <https://arxiv.org/abs/1408.5882>.
- [14] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014, <https://arxiv.org/abs/1409.2329>.
- [15] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 5998–6008, Long Beach, CA, USA, 2017.
- [16] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016, <https://arxiv.org/abs/1605.05101>.
- [17] Z. Yang, D. Yang, C. Dyer et al., "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 1480–1489, San Diego, CA, USA, 2016.
- [18] W. Zhao, J. Ye, M. Yang et al., "Investigating capsule networks with dynamic routing for text classification," 2018, <https://arxiv.org/abs/1804.00538>.
- [19] G. Wang, C. Li, W. Wang et al., "Joint embedding of words and labels for text classification," 2018, <https://arxiv.org/abs/1805.04174>.
- [20] M. Bao, J. Li, J. Zhang et al., "Learning semantic coherence for machine generated spam text detection," in *Proceedings of the 2019 International Joint Conference on neural networks (IJCNN)*, pp. 1–8, IEEE, Budapest, Hungary, 2019.
- [21] A. Onan, "Sentiment analysis on massive open online course evaluations: a text mining and deep learning approach," *Computer Applications in Engineering Education*.
- [22] A. Onan, "Mining opinions from instructor evaluation reviews: a deep learning approach," *Computer Applications in Engineering Education*, vol. 28, no. 1, pp. 117–138, 2019.
- [23] A. Onan, "An ensemble scheme based on language function analysis and feature engineering for text genre classification," *Journal of Information Science*, vol. 44, no. 1, pp. 28–47, 2018.
- [24] A. Onan, "Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks," *Concurrency and Computation: Practice and Experience*, Article ID e5909, 2020.
- [25] A. Onan, "Weighted word embeddings and clustering-based identification of question topics in MOOC discussion forum posts," *Computer Applications in Engineering Education*, 2020.
- [26] A. Onan, "Sentiment analysis in turkish based on weighted word embeddings," in *Proceedings of the 28th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, Gaziantep, Turkey, 2020.
- [27] A. Onan, "Ensemble of keyword extraction methods and classifiers in text classification," *Expert Systems with Applications*, vol. 57, pp. 232–247, 2016.
- [28] A. Onan, "A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification," *Information Processing & Management*, vol. 53, no. 4, pp. 814–833, 2017.
- [29] A. Onan, "Two-stage topic extraction model for bibliometric data analysis based on word embeddings and clustering," *IEEE Access*, vol. 7, pp. 145614–145633, 2019.
- [30] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, <https://arxiv.org/abs/1412.6980>.
- [31] A. Joulin, E. Grave, P. Bojanowski et al., "Fasttext zip: compressing text classification models," 2016, <https://arxiv.org/abs/1612.03651>.
- [32] S. Lai, L. Xu, K. Liu et al., "Recurrent convolutional neural networks for text classification," in *Proceedings of the Twenty-ninth AAAI Conference on Artificial Intelligence*, Austin, TX, USA, 2015.
- [33] P. Zhou, W. Shi, J. Tian et al., "Attention-based bidirectional long short-term memory networks for relation classification," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 207–212, Berlin, Germany, 2016.
- [34] A. Onan, "A term weighted neural language model and stacked bidirectional LSTM based framework for sarcasm identification," *IEEE Access*, vol. 9, pp. 7701–7722, 2021.