Text Classification Using a Word-Reduced Graph

Hiromu Nakajima
Major in Computer and Information Sciences
Graduate School of Science and Engineering,
Ibaraki University
Hitachi, Ibaraki, Japan
e-mail: 22nm738g@vc.ibaraki.ac.jp

Minoru Sasaki
Dept. of Computer and Information Sciences
Faculty of Engineering, Ibaraki University
Hitachi, Ibaraki, Japan
e-mail: minoru.sasaki.01@vc.ibaraki.ac.jp

Abstract— **Text classification, which determines the label of a document based on cues such as the co-occurrence of words and their frequency of occurrence, has been studied in various approaches to date. Conventional text classification methods using graph structure data express the relationship between words, the relationship between words and documents, and the relationship between documents in terms of the weights of edges between each node. They are then trained by inputting into a graph neural network. However, text classification methods using those graph-structured data require a very large amount of memory, and therefore, in some environments, they do not work properly or cannot handle large data. In this study, we propose a graph structure that is more compact than conventional methods by removing words that appear in only one document and are considered unnecessary for text classification. In addition to save memory, the proposed method can use a larger trained model by utilizing the saved memory. The results showed that the method succeeded in saving memory while maintaining the accuracy of the conventional method. By utilizing the saved memory, the proposed method succeeded in using larger trained models, and the classification accuracy of the proposed method was dramatically improved compared to the conventional method.**

Keywords- text classification; graph convolutional neural network; semi-supervised learning.

I. INTRODUCTION

Text classification is the task of estimating the appropriate label for a given document from a predefined set of labels. This text classification technique has been applied in the real world to automate the task of classifying documents by humans. Many researchers are interested in developing applications that take advantage of text classification techniques, such as spam classification, topic labeling, and sentiment analysis.

Recently, Graph Convolutional Neural networks (GCNs) [1], which can take advantage of data in graph structures, have been used to solve text classification tasks. TextGCN [2], VGCN-BERT [3], and BertGCN [4] are examples of text classification methods that utilize data from graph structures. In TextGCN [2], word and document nodes are represented on the same graph (heterogeneous graph), which is input into GCNs for learning. VGCN-BERT [3] constructs a graph based on the word embedding and word co-occurrence information in Bidirectional Encoder Representations from Transformers (BERT), and learns by inputting the graph into Vocabulary Graph Convolutional Network (VGCN). BertGCN [4] is a text classification method that combines the advantages of transductive learning of GCNs with the knowledge obtained from large-scale prior learning of BERT. The graphs produced by these graph-based text classification methods use relations between words and between words and documents, but do not use relations between documents, and are prone to topic drift. Therefore, in [5], we proposed a graph structure that uses relations between documents to solve this problem. The method of [5] boasts the best performance among existing methods for text classification on three datasets (20NG, R8, and Ohsumed). However, a new problem arises from the addition of relationships between documents to the graph, which increases the size of the graph and requires a lot of memory space. Therefore, we considered that compacting the size of the graph would reduce the memory requirement and allow the use of larger data and larger trained models.

The purpose of this study is twofold. The first is to successfully save memory by constructing a graph structure that removes words considered unnecessary in text classification to solve the problem of insufficient memory. The second is to improve classification accuracy over conventional methods by utilizing the reduced memory and using larger trained models. Specifically, words that appear in only one document are removed from the graph, reducing both the weights of edges between word nodes and the weights of edges between word nodes and document nodes, thereby saving memory. We believe that this will result in a graph that is more compact than the graphs created by conventional methods, saving memory and improving the accuracy of text classification by using a larger trained model.

This paper is organized as follows. In Section 2, we first describe graph neural networks used for text classification and existing research on text classification using graphs. After that, the structure of graphs created in conventional methods is described. In Section 3, we describe the graph structure of the proposed method and the processing after graph construction. In Section 4, we describe the experiments we conducted to evaluate the proposed method and show the experimental results. In Section 5, we discuss the experimental results presented in Section 4 and conclude in Section 6.

II. RELATED WORKS

A. *Text Classification Using Graph Neural Networks*

Graph Neural Network (GNN) [6] is a neural network that learns relationships between graph nodes via the edges that connect them. There are several types of GNNs depending on their form. Graph Convolutional Neural networks (GCNs) [1]

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

is a neural network that takes a graph as input and learns the relationship between nodes of interest and their neighbors through convolutional computation using weights assigned to the edges between the nodes. Graph Autoencoder (GAE) [7] is an extension of autoencoder, which extracts important features by dimensionality reduction of input data, to handle graph data as well. Graph Attention Network (GAT) [8] is a neural network that updates and learns node features by multiplying the weights of edges between nodes by Attention, a coefficient representing the importance of neighboring nodes. GNNs are used in a wide range of tasks in the field of machine learning, such as relation extraction, text generation, machine translation, and question answering, and have demonstrated high performance. The success of GNNs in these wide-ranging tasks has motivated researchers to study text classification methods based on GNNs, and in particular, many text classification methods based on GCNs have been proposed. In TextGCN [2], document and word nodes are represented on the same graph (heterogeneous graph), which is input into GCNs for training. In recent years, text classification methods that combine large-scale pre-trained models such as BERT with GCNs have also been studied extensively. VGCN-BERT [3] constructs a graph based on word co-occurrence information and BERT's word embedding, and inputs the graph into GCNs for learning. In BertGCN [4], a heterogeneous graph of words and documents is constructed based on word co-occurrence information and BERT's document embedding, and the graph is input into GCNs for learning. In [5] we propose a graph structure that exploits relationships between documents.

The detailed description of BertGCN and the graph structure proposed in [5] is given in the Section II-B.

B. *Graph Structure of Conventional Methods*

BertGCN is a text classification method that combines the knowledge of BERT obtained by large-scale pre-training utilizing large unlabeled data with the transductive learning of GCNs, and was proposed by Lin et al. in July 2021. In BertGCN, each document is input into BERT, the document vector is extracted from its output, and it is input into GCNs as an initial representation of document nodes together with a heterogeneous graph of documents and words for training.

Lin et al. distinguish the names of the training models according to the pre-trained model of BERT and the type of GNN used. The correspondence between pre-trained models and model names is shown in Table I. In this study, RoBERTaGCN, a model employing roberta-base and GCNs, is the target of improvement.

TABLE I. NAMES OF THE MODELS.

Pre-Trained Model	GNN	Name of Model
bert-base	GCN	BertGCN
roberta-base	GCN	RoBERTaGCN
bert-base	GAT	BertGAT
roberta-base	GAT	RoBERTaGAT

RoBERTaGCN defines weights between nodes on heterogeneous graphs of words and documents as in (1).

$$A_{i,j} = \begin{cases} PPMI(i,j), & i,j \text{ are words and } i \neq j \\ TF-IDF(i,j), & i \text{ is document}, j \text{ is word} \\ 1, & i = j \\ 0, & otherwise \end{cases} \quad (1)$$

Positive point-wise mutual information (PPMI) is used to weight edges between word nodes. PPMI is a measure of the degree of association between two events and can be viewed as word co-occurrence in natural language processing. Term frequency-inverse document frequency (TF-IDF) values are used for the weights of edges between word nodes and document nodes; TF-IDF values are larger for words that occur more frequently in a document but less frequently in other documents, i.e., words that characterize that document.

In [5], weights between nodes on heterogeneous graphs of words and documents are defined as in (2).

$$A_{i,j} = \begin{cases} COS\_SIM(i,j), & i,j \text{ are documents and } i \neq j \\ PPMI(i,j), & i,j \text{ are words and } i \neq j \\ TF-IDF(i,j), & i \text{ is document}, j \text{ is word} \\ 1, & i = j \\ 0, & otherwise \end{cases} \quad (2)$$

As shown in (1), RoBERTaGCN considered relations between words and relations between words and documents in the form of weights of edges between nodes, but did not consider relations between documents. Therefore, we have improved RoBERTaGCN to consider the relationship between documents by expressing the relationship between documents in the form of weights of edges between document nodes. $COS\_SIM(i,j)$ in (2) is the weight of edges between document nodes and represents the cosine similarity. Each document is tokenized and input into BERT to obtain a embedding for each document. The Cosine similarity is calculated between the obtained vectors, and if the Cosine similarity exceeds a predefined threshold value, the Cosine similarity is added as weights of edges between corresponding document nodes.

III. METHOD

This section describes the details of the proposed method. Figure 1 shows a schematic diagram of the proposed method. First, a heterogeneous graph of words and documents is constructed from documents. Next, the graph information (weight matrix and initial node feature matrix) is input into the GCN, and the document vector is input into the feed-forward neural network. Finally, a linear interpolation of the two predictions is computed and the result is used as the final prediction.

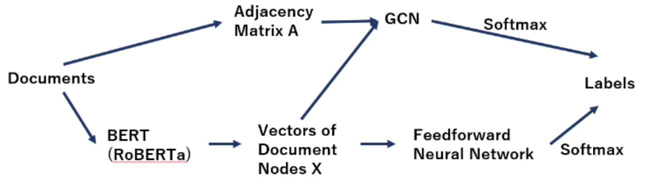

Figure 1. Schematic Diagram of the Proposed Method.

A. Build Heterogeneous Graph

First, a heterogeneous graph containing word and document nodes is constructed. The weights of edges between nodes i and j are defined as in (3). df indicates the number of documents in which the word appears. The difference between (2) and (3) is that the words that appear in only one document are removed. This reduces both the weights of edges between word nodes and the weights of edges between word nodes and document nodes, thus saving memory. PPMI is used for the weights of edges between word nodes, and TF-IDF values are used for the weights of edges between word and document nodes.

$$A_{i,j} = \begin{cases} COS\_SIM(i,j), & i,j \text{ are documents and } i \neq j \\ PPMI(i,j), & i,j \text{ are words and } i \neq j \\ & df(i) > 1, df(j) > 1 \\ TF-IDF(i,j), & i \text{ is document}, j \text{ is word} \\ & df(j) > 1 \\ 1, & i = j \\ 0, & otherwise \end{cases} \tag{3}$$

The process from Section B to E below is based on RoBERTaGCN [4].

B. Creating the Initial Node Feature Matrix

Next, we create the initial node feature matrix to be input into the GCNs. We use BERT to obtain the document embeddings and treat them as the input representations of the document nodes. The embedded representation X_{doc} of a document node is represented by $X_{doc} \in \mathbb{R}^{n_{doc} \times d}$, where n_{doc} is the number of documents and d is the number of embedding dimensions. Overall, the initial node feature matrix is given by (4).

$$X = \begin{pmatrix} X_{doc} \\ 0 \end{pmatrix}_{(n_{doc}+n_{word}) \times d} \tag{4}$$

C. Input into GCN and Learning by GCN

The weights of the edges between nodes and the initial node feature matrix are input into GCNs for training. The output feature matrix $L^{(i)}$ of layer i is computed by (5).

$$L^{(i)} = \rho\left(\tilde{A} L^{(i-1)} W^{(i)}\right) \tag{5}$$

ρ is the activation function and $\tilde{A}$ is the normalized adjacency matrix. $W^i \in \mathbb{R}^{d_{i-1} \times d_i}$ is the weight matrix at layer i. $L^{(0)}$ is X, the input feature matrix of the model. The dimension of the final layer of W is (number of embedded dimensions) × (number of output classes). The output of the GCNs is treated as the final representation of the document node, and its output is input into the softmax function for classification. The prediction by the output of the GCNs is given by (6). g represents the GCNs model. The cross-entropy loss in labeled document nodes is used to cooperatively optimize the parameters of BERT and GCNs.

$$Z_{GCN} = softmax\left(g(X, A)\right) \tag{6}$$

D. Input into Feedforward Neural Network and Learning by Feedforward Neural Network

Optimizing GCNs with an auxiliary classifier that directly handles BERT-embedded representations leads to faster convergence and improved performance. Specifically, a document embedded representation X is input into a Feedforward Neural Network. The output is then fed directly into a softmax function with a weight matrix W to create an auxiliary classifier with BERT. The prediction by the auxiliary classifier is given by (7).

$$Z_{BERT} = softmax(WX) \tag{7}$$

E. Interpolation of Predictions with BERT and GCN

A linear interpolation is computed with Z_{GCN}, the prediction from RoBERTaGCN, and Z_{BERT}, the prediction from BERT, and the result of the linear interpolation is adopted as the final prediction. The result of the linear interpolation is given by (8).

$$Z = \lambda Z_{GCN} + (1 - \lambda) Z_{BERT} \tag{8}$$

λ controls the trade-off between the two predictions. $\lambda = 1$ means using the full RoBERTaGCN model, while $\lambda = 0$ means using only the BERT module. When $\lambda \in (0, 1)$, the predictions from both models can be balanced, making the RoBERTaGCN model more optimal. Experiments by Lin et al. using the graph structure in (1) show that $\lambda = 0.7$ is the optimal value of λ.

IV. EXPERIMENTS

In this study, two experiments were conducted.

Experiment 1: Experiment to confirm the effectiveness of the graphs of the proposed method.

In Experiment 1, the classification performance of the proposed method using compact graphs was compared with other methods. The parameter λ, which controls the balance between predictions from BERT and predictions from GCNs, was fixed at 0.7. Preliminary experiments were conducted on the validation data, and the optimal values of the threshold of the cosine similarity for each dataset are shown in Table II. We used the values in Table II as our threshold values. The trained model used was roberta-base. Accuracy was used to evaluate the experiment. Positive is the label of the correct answer, negative is the label of the incorrect answer, and negative is all the remaining labels except the correct label.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

TABLE II. OPTIMAL VALUE FOR COSINE SIMILARITY THRESHOLD.

Dataset	Optimal Threshold Value
20NG	0.99
R8	0.975
R52	0.96
Ohsumed	0.965
MR	0.97

Experiment 2: Experiment to check classification accuracy when changing to a larger trained model.

In Experiment 2, we take advantage of the memory savings and check the accuracy of the proposed method by applying a larger trained model. Specifically, the learned model is changed from roberta-base to roberta-large. λ and cosine similarity values are set to the same values as in Experiment 1.

A. Data Set

We evaluated the performance of the proposed method by conducting experiments using the five data sets shown in Table III. We used the same data used in RoBERTaGCN. Each dataset was already divided into training and test data, which we used as is. The ratio of training data to test data is about 6:4 for 20NG, about 7:3 for R8 and R52, about 4.5:5.5 for Ohsumed, and about 6.5:3.5 for MR.

TABLE III. INFORMATION OF EACH DATA SET.

Dataset	Number of Documents	Average of Words
20NG	18846	206.4
R8	7674	65.7
R52	9100	69.8
Ohsumed	7400	129.1
MR	10662	20.3

・20-Newsgroups (20NG)

20NG is a dataset in which each document is categorized into 20 news categories, and the total number of documents is 18846. In our experiments, we used 11314 documents as training data and 7532 documents as test data.

・R8, R52

Both R8 and R52 are subsets of the dataset provided by Reuters (total number is 21578). R8 has 8 categories and R52 has 52 categories. The total number of documents in R8 is 7674, and we used 5485 documents as training data and 2189 documents as test data. The total number of documents in R52 is 9100, and we used 6532 documents as training data and 2568 documents as test data.

・Ohsumed

This is a dataset of medical literature provided by the U.S. National Library of Medicine, and total number of documents is 13929. Every document has one or more than two related disease categories from among the 23 disease categories. In the experiment, we used documents that had only one relevant disease category, and the number of documents is 7400. We used 3357 documents as training data and 4043 documents as test data.

・Movie Review (MR)

This is a dataset of movie reviews and is used for sentiment classification (negative-positive classification). The total number of documents was 10662. We used 7108 documents as training data and 3554 documents as test data.

B. Experimental Environment

The experiments were conducted using Google Colaboratory Pro+, an execution environment for Python and other programming languages provided by Google. The details of the specifications of Google Colaboratory Pro+ are shown in Table IV.

TABLE IV. DETAILS OF THE SPECIFICATIONS OF GOOGLE COLABORATORY PRO+.

GPU	Tesla V100（SXM2）/ A100（SXM2）
Memory	12.69GB（standard）/ 51.01GB（CPU / GPU (high memory)）/ 35.25GB（TPU (high memory)）
Disk	225.89GB（CPU / TPU）/ 166.83GB（GPU）

C. Result of Experiment

TABLE V. CLASSIFICATION PERFORMANCE OF THE PROPOSED METHOD.

	20NG	R8	R52	Ohsumed	MR
Text GCN	86.34	97.07	93.56	68.36	76.74
Simplified GCN	88.50	-	-	68.50	-
LEAM	81.91	93.31	91.84	58.58	76.95
SWEM	85.16	95.32	92.94	63.12	76.65
TF-IDF +LR	83.19	93.74	86.95	54.66	74.59
LSTM	65.71	93.68	85.54	41.13	75.06
fastText	79.38	96.13	92.81	57.70	75.14
BERT	85.30	97.80	96.40	70.50	85.70
RoBERTa	83.80	97.80	96.20	70.70	89.40
RoBERTa GCN	89.15	98.58	94.08	72.94	88.66
[5]	89.82	**98.81**	94.16	74.13	89.00
Proposed method (base)	**90.02**	98.58	**96.88**	73.53	89.65
Proposed method (large)	89.95	98.58	96.81	**76.08**	**91.50**

Table V compares the classification performance of the proposed method with the conventional methods. [5] shows the classification performance when using the graph structure in (2). proposed method (base) is the result of Experiment 1, and proposed method (large) is the result of Experiment 2.

Comparing the results of the Proposed method (base) with the other methods, the accuracy of 20NG, R52, and MR improved. The accuracy of the other datasets also maintains a high level. Even with a compact graph in which words that appear only in one document are removed, the classification performance remains high. Therefore, it can be said that the proposed method succeeds in saving memory.

Comparing the results of the Proposed method (large) with the other methods, the accuracy is significantly improved for Ohsumed and MR. The classification performance of Ohsumed was 76.08%, 1.95% higher than that of [5], and that of MR was 91.50%, 1.85% higher than that of the Proposed method (base).

V. Discussion

Table VI shows the number of word types that appear in each dataset and the number of words that are removed in the graph structure of (3). Table VII shows the number of PPMI edges added in the original graph structure and the number of PPMI edges removed in the graph structure of (3). Table VIII shows the number of TF-IDF edges added in the original graph structure and the number of TF-IDF edges removed in the graph structure of (3). Since TF-IDF edges are added between word and document nodes, the number of edges removed is the same as the number of words removed. From these three tables, it can be seen that the graph of the proposed method reduces the number of edges by 1 to 20%. Experimental results show that the classification performance of the proposed method maintains performance of the method using the original graph structure. Therefore, it can be said that the proposed method succeeds in saving memory because it reduces the number of edges on the graph while maintaining the accuracy.

We believe that the reason why the accuracy was maintained even with a compact graph is because the words to be removed were limited to words that appear only in a single document. Words that appear in only one document do not propagate document topic information through the word node, and thus text classification performance is maintained even if those words are removed.

This study also confirmed the document classification performance when the trained model was changed to a larger one, taking advantage of the memory savings. When the learned model was changed from roberta-base to roberta-large, the accuracy improved significantly. It is thought that the change to roberta-large improved the accuracy because it was able to acquire embedded representations that better reflect the characteristics of the documents.

TABLE VI. Number of Words Removed.

Dataset	Number of Words	Number of Words Removed
20NG	42757	755
R8	7688	225
R52	8892	245
Ohsumed	14157	851
MR	18764	8687

TABLE VII. Number of PPMI Edges Removed.

Dataset	Number of PPMI Edges	Number of Edges Removed
20NG	22413246	127662
R8	2841760	32954
R52	3574162	36138
Ohsumed	6867490	129938
MR	1504598	314950

TABLE VIII. Number of TF-IDF Edges Removed.

Dataset	Number of TF-IDF Edges	Number of Edges Removed
20NG	2276720	755
R8	323670	225
R52	407084	245
Ohsumed	588958	851
MR	196826	8687

VI. Conclusion and Future Work

To solve the memory-consuming problem of conventional text classification methods based on graph structures, this paper proposes the text classification method using compact graphs in which words that appear only in one document are removed. Experiments confirmed that the proposed method can maintain the accuracy of the conventional method while saving a lot of memory. Experiments also showed that the acuracy of text classification improves when the learned model is changed to a larger one, taking advantage of the saved memory.

Future work includes comparing the accuracy with the proposed method when other features are used instead of cosine similarity and optimizing the parameter λ for each data.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks." In ICLR, 2017.

[2] L. Yao, C. Mao and Y. Luo, "Graph convolutional networks for text classification." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 7370-7377, 2019.

[3] Z. Lu, P. Du and J. Y. Nie, "Vgcn-bert: augmenting bert with graph embedding for text classification." In European Conference on Information Retrieval, pp. 369-382, 2020.

[4] Y. Lin, Y. Meng, X. Sun, Q. Han, K. Kuang, J. Li and F. Wu, "BertGCN: Transductive Text Classification by Combining GCN and BERT" In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 1456–1462, 2021.

[5] H. Nakajima and M. Sasaki, "Text Classification Using a Graph Based on Relationships Between Documents." In Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation, pp. 119–125, Manila, Philippines. De La Salle University. 2022.

[6] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The graph neural network model," IEEE Transactions on Neural Networks, vol.20, no.1, pp.61-80, 2008.

[7] T. N Kipf and M. Welling, "Variational graph auto-encoders." arXiv preprint arXiv:1611.07308. 2016b.

[8] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph attention networks." arXiv preprint arXiv:1710.10903. 2017.

[9] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani and K. S. An, "Graph convolutional encoders for syntax-aware neural machine translation." In Proceedings of the 2017 Conference on Empirical Methods in

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

Natural Language Processing, pp. 1957-1967, Copenhagen, Denmark. Association for Computational Linguistics. 2017.

[10] L. Huang, D. Ma, S. Li, X. Zhang and H. Wang, “Text level graph neural network for text classification.” 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. 3444–3450, 2019.

[11] R. K. Bakshi, N. Kaur, R. Kaur and G. Kaur, “Opinion mining and sentiment analysis.” In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 452-455, IEEE. 2016.

[12] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu and J. Gao, “Deep Learning Based Text Classification: A Comprehensive Revie.” ACM Computing Surveys, vol.54, Issue 3, no.62, pp.1-40, 2021.

[13] X. Liu, X. You, X. Zhang, J. Wu and P. Lv, “Tensor graph convolutional networks for text classification” arXiv:2001.05313v1. pp.8409-8416, 2020.

[14] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao and L. Carin, “Joint embedding of words and labels for text classification.” In ACL, pp.2321–2331, 2018.

[15] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao and L. Carin, “Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms.” In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (vol. 1: Long Papers), pp. 440–450, 2018.

[16] P. Liu, X. Qiu and X. Huang, “Recurrent neural network for text classification with multi-task learning.” In IJCAI, pp.2873–2879, AAAI Press. 2016.

[17] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov, “Bag of tricks for efficient text classification.” In EACL, pp.427–431, Association for Computational Linguistics. 2017.

[18] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” Proceedings of NAACL-HLT 2019, pp. 4171–4186, 2019.

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” arxiv:1907.11692v1. 2020.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org