A Comparative Study on Embedding Models for Keyword Extraction Using KeyBERT Method

Bayan Issa* [1], Muhammed Basheer Jasser [2], Hui Na Chua [3], Muzaffar Hamzah [4]
[1] Faculty of Informatics Engineering, University of Aleppo, Syria
[2, 3] Department of Computing and Information Systems, Sunway University, Sunway City, Selangor, Malaysia
[4] Faculty of Computing and Informatics, Universiti Malaysia Sabah, Kota Kinabalu 88450, Sabah, Malaysia
Email: bayan.issa.b@gmail.com [1], basheerj@sunway.edu.my[2], huinac@sunway.edu.my [3], muzaffar@ums.edu.my [4]
Corresponding Author: Bayan Issa (bayan.issa.b@gmail.com)

2023 IEEE 13th International Conference on System Engineering and Technology (ICSET) | 979-8-3503-4089-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSET59111.2023.10295108

Abstract— **KeyBERT is a method for keywords/keyphrases extraction, which has three steps. The first step is selecting candidate keywords from a text using sklearn library, the second step is the embedding operation of the text and its candidate keywords; this operation is done by BERT to get a numerical representation that represents the meanings. The third step is calculating the cosine similarity between individual keywords vectors and document vector. In this paper, we focus on the second step of KeyBERT (embedding step). Although KeyBERT has a lot of supported models for the embedding operation, there are no extensive previous comparative studies to analyze and study the effect of using different supported models in KeyBERT. We introduce a comparative study of two commonly used groups of models; the first group is sentence_transformers pretrained models, supported via the sentence-transformers library, and the second group includes the Longformer model, supported via the Hugginface Transformers library. We conduct the comparative study of models on benchmark datasets, which contain English text documents of multi-domains with different text lengths. Based on the study, we found that the Paraphrase-mpnet-base-v2 model provides the best results among all other models in keyword extraction in terms of effectiveness (f1-score, recall, precision, MAP) on all datasets, with higher efficiency (time) on short text compared with using it on long text; accordingly, we recommend using it in that context. On the other hand, the Longformer model is the most efficient/fastest in keyword extraction among all other models on all datasets and this superiority has been evident, especially in long text; accordingly, we recommend using it in that context.**

Keywords—Keyword Extraction, NLP, Pretrained Model, Embedding Models

I. INTRODUCTION

Text information has spread massively, especially when social media has become very common recently. We have more textual documents that are generated on a daily basis. For that, the need to summarize and extract the main ideas from this huge number of text documents is becoming more important. Many methods have been introduced by Academia and Industry to extract keywords and they are categorized into four categories: heuristics approaches, supervised approaches, unsupervised approaches, and Neural network approaches.

Heuristics approaches, such as in [1,2,3], work on selecting keywords from text gradually or pruning of words, which are not considered to be keywords, using some key-heuristics such as Part Of Speech (POS), lexical information, lemmatization, and named entity recognitions.

Supervised approaches, such as in [4,5,6] use classification methods like SVM, Naïve-Bayes, and random forest.

Unsupervised approaches are divided into three categories: statistical approaches such as in these studies [7,8]. graph based approach such as in these studies [9,10,11] and deep learning approach such as in [12,13,14]; we focus on this approach in our study on KeyBERT.

Neural network approaches used Multi-Layer Perceptron (MLP) [15] and Recurrent Neural Network (RNN)[16].

In our study, we focus on KeyBERT, which is considered as an unsupervised deep learning approach, and we compare among performances of several embedding models of KeyBERT in extracting meaningful keywords. Although some comparative studies [17,18,19,20] have been done to compare between KeyBERT and other methods, there is no extensive previous comparative studies to analyze and study the effect of using different supported models in KeyBERT.

The contribution of our work is as follows:

- To introduce an extensive comparative study on embedding models, which are supported by KeyBERT to extract keywords.
- To include effective evaluation measurements such as the keywords appearing order (MAP).
- To observe the text length effect on the accuracy of keyword extraction.
- To provide a recommendation for employing specific models according to tackled text length.

The following paper is structured as follows: Section II presents a background on KeyBERT, Section III introduces a background on the sentence-transformers models, Section IV presents a background on Longformer model, Section V contains a comparison among models in KeyBERT. In Section VI, we discuss our results and give recommendations. Section VII concludes the work.

II. KEYBERT METHOD

KeyBERT, as shown in Figure 1, has three steps: the selection of candidate keywords, the embedding operation and finally the calculation of the cosine similarity.

The first step is selecting a list of candidate keywords or keyphrases from a document using Scikit-Learns, which is superior to other libraries in two features: firstly, it allows to adjust the keyphrase length; when it is adjusted to one, KeyBERT provides keywords. The other feature is to quickly remove stop words.

Scikit-Learns has 'n_gram_range' parameter to adjust the candidate keyphrase length. For example, if we would set it to (2,2), then each candidate phrase would include two keywords.

In this study, we extract keywords which mean we adjust this parameter to (1,1).

The second step is the embedding operation. KeyBERT represents the document and the candidate keywords/keyphrases as numerical data. It uses BERT [21] because of it provides great results for similarity and paraphrasing tasks. Also, KeyBERT supports many other models for generating the BERT embeddings integrated in many libraries such as Flair, Hugginface Transformers, and spaCy.

Our study uses the sentence-transformers package, which has quick and high-quality embeddings, and Longformer model [22] supported by Hugginface Transformers library. The Longformer model performs effectively and efficiently with long documents. It depends on splitting long document to windows that move through the text to calculate the numerical representation of document.

The third step is calculating the cosine similarity. KeyBERT selects the most similar keywords/keyphrases to the document. To achieve that, it uses cosine similarity between individual keywords vectors and document vector.

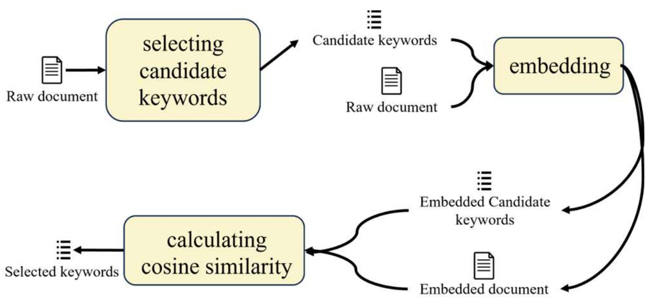

Figure 1, KeyBERT steps

III. SENTENCE-TRANSFORMERS MODELS

Sentence-transformers models are a set of new state-of-the-art performance models that is designed especially for sentence-pair regression tasks like semantic textual similarity (STS), which forms the core of keywords extraction task.

In [23], the researchers modify the pretrained BERT into siamese and triplet network structures. That modification made the BERT's embeddings more semantically meaningful, so that it could perform better in STS tasks. Also, that helped to reduce the finding time of similarity from 65 hours with BERT / RoBERTa to about 5 seconds with SBERT while maintaining the BERT accuracy.

We use two models from sentence-transformers models: Paraphrase-mpnet-base-v2 and Paraphrase-distilroberta-base-v2.

Paraphrase-mpnet-base-v2: its Base Model is mpnet-base that is made by Microsoft. Its size is 420 MB. Paraphrase-mpnet-base-v2 is pretrained on 12 datasets and it achieves high performance on 14 datasets for Sentence Embeddings task.

Paraphrase-distilroberta-base-v2: its Base Model is distilroberta-base, which is designed for fine-tuning proposes on tasks like token classification, which is related to keywords extraction task. It is a distiled model with a size of 290MB. Paraphrase-distilroberta-base-v2 is pretrained on 12 datasets and it achieves high performance on 14 datasets for Sentence Embeddings task.

IV. LONGFORMER MODEL

Longformer model is a transformer-based model, which is designed to deal with long documents because other Transformer-based models need quadratic time and quadratic memory for long sequence processing. However, the Longformer model scales linearly when sequence becomes longer and it performs state-of-the-art results on text8 and enwik8 datasets. The Longformer model is finetuned on many downstream tasks achieving state-of-the-art results on WikiHop and TriviaQA datasets. Longformer Self Attention is different from standard Self Attention:

- Standard Self Attention (full attention)

Transformer-based models, which depend on Standard Self Attention, have $O(n^2)$ time and memory complexity where n is the input sequence length, as shown in Figure 2 (a).

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Longformer Self Attention

Longformer has three mechanisms to solve the previous problem of time and memory complexity, which are: Sliding Window, Dilated Sliding Window, and Global Attention

Sliding Window mechanism makes a fixed-size window for each token, as shown in Figure 2 (b). This window has size w and it surrounds each token with $(1/2)\times w$ before it and $(1/2)\times w$ after it. This lowers the complexity to $O(n\times w)$, when the input sequence length is n.

Dilated Sliding Window: The Longformer employs this mechanism to increase the receptive field further, as in Figure 2 (c); the receptive field is $l \times d \times w$, where d and w are fixed for all layers. Increasing the receptive field increases the complexity. For that, The Longformer reduces dilation configurations for some heads to focus on the local context, which enhances the model performance.

Global Attention: it is a sliding window attention with a global attention for some tokens, as in Figure 2(d).

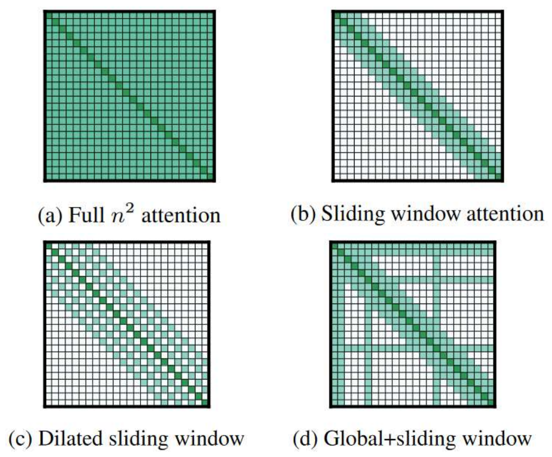
(a) Full n^2 attention
(b) Sliding window attention
(c) Dilated sliding window
(d) Global+sliding window

Figure 2, Comparing the full self-attention pattern and the configuration of attention patterns in Longformer model [22]

V. Comparison among models in KeyBert

A. Experimental Setup

In this study, we employ five metrics: f1_score, precision, recall, MAP (Mean Average Precision), and Execution time to evaluate the models' performances. MAP is employed because it considers the order of keywords list. The other metrics are obviously required in our study.

We conducted this study on five datasets, which are: SemEval2017, Inspect, 500N-KPCrowd-v1.1, Schutz2008, and SemEval2010. We choose these datasets because they include sequences of documents ordered according to their length.

Inspect and 500N-KPCrowd-v1.1 include documents whose lengths are under 1000 tokens in each document.

Schutz2008 includes documents whose lengths are between 1000 to 25000 tokens in each document.

SemEval2010 includes documents whose lengths are between 4000 to 14000 tokens in each document.

Those datasets contain key phrases and words. KeyBert requires the user to specify the length of the extracted phrases (through the n-gram parameter); and we set it in this experiment to 1 then KeyBert provides keywords and not phrases. Therefore, we evaluate the models via two different kinds of comparisons: a comparison based on keywords and a comparison based on Keyphrases

In the comparison that is based on keywords, we divide the real key phrases (in the dataset) into words, and then compare them with the keywords extracted by the model.

In the comparison that is based on Keyphrases, we keep the key phrases (in the dataset) unchanged and compare them with the extracted keywords by the model.

Our study is executed by google Colaboratory with its default hardware specifications. The default CPU for google Colaboratory is an Intel(R) Xeon(R) CPU @ 2.20GHz and 13GB of RAM.

B. Paraphrase-mpnet-base-v2 results

In TABLE-1, we present the performance of the *Paraphrase-mpnet-base-v2* model on four datasets, which is measured by: f1_score, precision, recall, and MAP. We do comparison here as based on Keyphrases.

TABLE-1. Paraphrase-mpnet-base-v2 Keyphrases evaluation

dataset\ Metric	f1_score	precision	recall	MAP
SemEval2017	0.1212	0.1208	0.12159	0.35973
Inspect	0.06894	0.06764	0.07046	0.2415
500N-KPCrowd-v1.1	0.28418	0.2833	0.28507	0.48451
SemEval2010	0.04548	0.04532	0.04566	0.17767

In TABLE-2, we present the performance of the *Paraphrase-mpnet-base-v2* on three datasets, which is measured by: f1_score, precision, recall, and MAP. We do a comparison here based on Keywords. We notice that the longer the dataset contains texts, the worse the model performs.

TABLE-2. Paraphrase-mpnet-base-v2 Keywords evaluation

dataset\ Metric	f1_score	precision	recall	MAP
Inspect	0.4816	0.43006	0.5524	0.69049
500N-KPCrowd-v1.1	0.4538	0.42264	0.49226	0.6345
SemEval2010	0.18099	0.16041	0.2099	0.3945

C. Paraphrase-distilroberta-base-v2 results

In TABLE-3, we present the performance of the *Paraphrase-distilroberta-base-v2* on four datasets, which is measured by: f1_score, precision, recall, and MAP. We do the comparison here based on Keyphrases.

TABLE-3. Paraphrase-distilroberta-base-v2 Keyphrases evaluation

dataset\ Metric	f1_score	precision	recall	MAP
SemEval2017	0.11612	0.11580	0.11647	0.34681
Inspect	0.0642	0.063	0.0656	0.2202
500N-KPCrowd-v1.1	0.2764	0.2756	0.27732	0.47
SemEval2010	0.0366	0.0365	0.0368	0.1523

In TABLE-4, we present the performance of the *Paraphrase-distilroberta-base-v2* on four datasets, which is measured by: f1_score, precision, recall, and MAP. We do the comparison here based on Keywords. We notice that the longer the dataset contains texts, the worse the model performs.

TABLE-4. PARAPHRASE-DISTILROBERTA-BASE-V2 KEYWORDS EVALUATION

dataset\Metric	f1_score	precision	recall	MAP
Inspect	0.47468	0.42370	0.54465	0.67028
500N-KPCrowd-v1.1	0.44901	0.418128	0.4870	0.61445
Schutz2008	0.14747	0.1291	0.172	0.3163
SemEval2010	0.16138	0.1427	0.18777	0.4099

D. *Longformer results*

In TABLE-5, we present performance of the *Longformer* model on four datasets, which is measured by: f1_score, precision, recall, and MAP. We do the comparison in this case based on Keyphrases.

TABLE-5. LONGFORMER KEYPHRASES EVALUATION

dataset\Metric	f1_score	precision	recall	MAP
SemEval2017	0.10495	0.104651	0.10527	0.28227
Inspect	0.216	0.0631	0.0645	0.216
500N-KPCrowd-v1.1	0.28139	0.280531	0.28229	0.4803
SemEval2010	0.03496	0.03482	0.03510	0.12245

In TABLE-6, we present the *Longformer* model on four datasets, which is measured by: f1_score, precision, recall, and MAP. We do the comparison in this case based on Keywords. We notice that the longer the dataset contains texts, the worse the model performs.

TABLE-6. LONGFORMER KEYWORDS EVALUATION

dataset\Metric	f1_score	precision	recall	MAP
Inspect	0.47267	0.4218	0.5423	0.6782
500N-KPCrowd-v1.1	0.45004	0.41939	0.48867	0.62098
Schutz2008	0.18063	0.15798	0.2120	0.2847
SemEval2010	0.16309	0.14455	0.18909	0.3407

E. *Comparison Among the models' performances*

In TABLE-7, we compare among the models' performances depending on the f1-score.

TABLE-7, COMPARISON AMONG THE MODELS' PERFORMANCES (F1-SCORE)

model\dataset	Inspect	500N-KPCrowd-v1.1	Schutz2008	SemEval2010
Longformer	0.47267	0.45004	**0.18063**	0.16309
paraphrase-distilroberta-base-v2	0.47468	0.44901	0.14747	0.16138
paraphrase-mpnet-base-v2	**0.4816**	**0.4538**	-	**0.18099**

In TABLE-8, we Compare among the models' execution times (average-execution-time in second).

TABLE-8, COMPARISON BETWEEN THE MODELS' EXECUTION TIMES (AVERAGE-EXECUTION-TIME BY SECOND)

model\dataset	500N-KPCrowd-v1.1	Schutz2008	SemEval2010
Longformer	**1.01838**	**6.1746**	**6.0923**
paraphrase-distilroberta-base-v2	1.81648	11.3625	11.4302
paraphrase-mpnet-base-v2	2.8383	-	22.5632

VI. DISCUSSION AND RECOMMENDATIONS

The results, which are achieved by the *Paraphrase-mpnet-base-v2* model in Table-7, are better than those of other models (except on Schutz2008 dataset). Therefore, we put more emphasis on evaluating this model's performance on different text length and different numbers of tokens.

The f1-score, achieved by the *Paraphrase-mpnet-base-v2* model reaches its highest peak (0.85) at 50 tokens, as shown in Figure 3. However, as in Figure 4 (a), (b), (c), (d), and (e), the f1-score gradually decreases, as the length of the text, represented by the number of tokens, increases. As in Figure 5, the accuracy of the extracted keywords decreases into 0.4 at 1000 tokens.

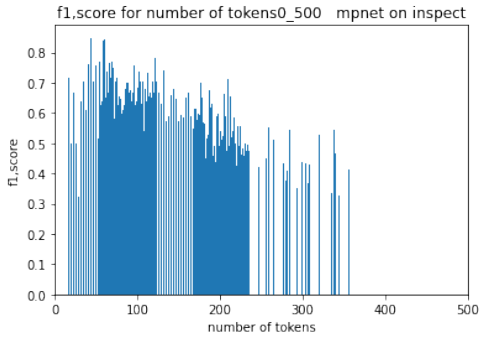

Figure 3, f1-score for Paraphrase-mpnet-base-v2 that is evaluated on Inspect for texts length under 500 tokens

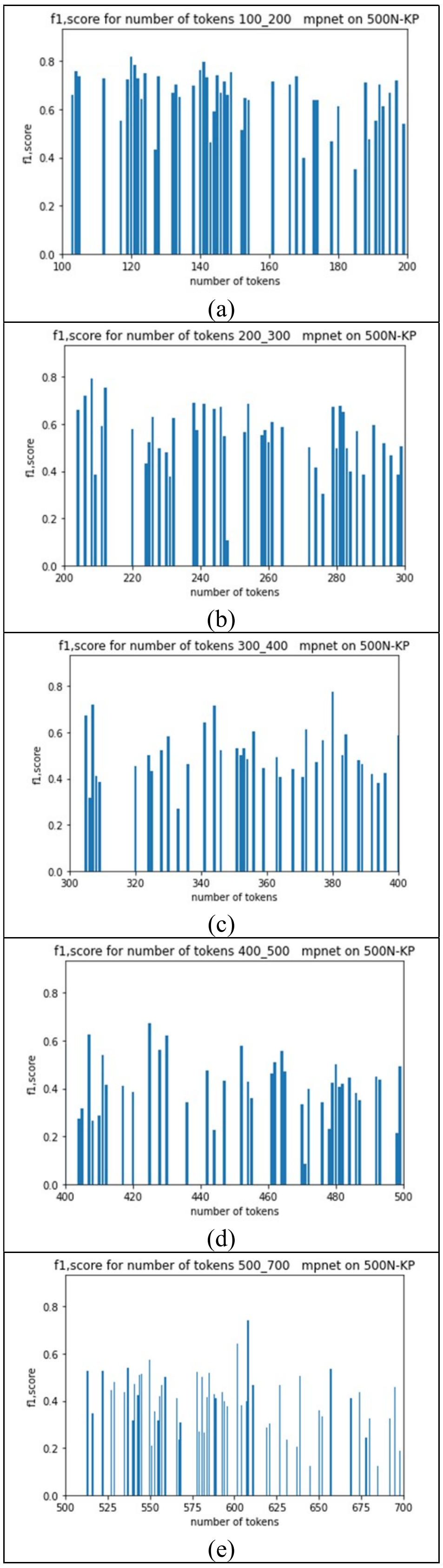

Figure 4, f1-score for Paraphrase-mpnet-base-v2 that is evaluated on 500N-KPCrowd-v1.1 for texts length from 100 to 700 tokens

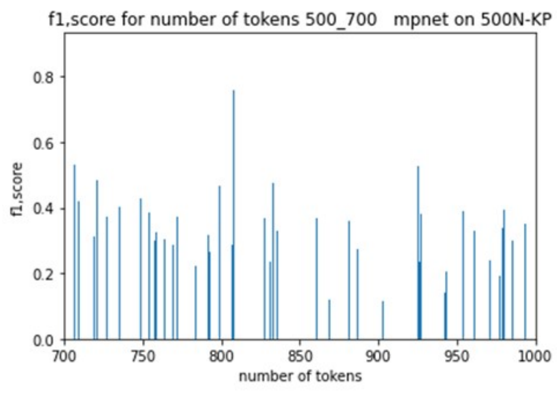

Figure 5, f1-score for Paraphrase-mpnet-base-v2 that is evaluated on 500N-KPCrowd-v1.1 for texts length from 700 to 1000 tokens

For comparing the model *Paraphrase-mpnet-base-v2* performances on larger text lengths, we applied it on Schutz2008 and semeval2010 that include documents of larger text and numbers of tokens. We notice from Table 7 that the f1-score decreases on Schutz2008 and semeval2010. Further, it caused memory crash when it is applied on Schutz2008 dataset, which has text length up to 25000 tokens.

From Table 7, we notice that f1-score values of all models decrease when we use datasets of longer text. However, *Paraphrase-mpnet-base-v2* has the best results but it needs so much time to process the long text, as shown in Table 8, especially when it couldn't deal with very long text in Schutz2008 dataset and caused memory crash. On the other hand, *Paraphrase-distilroberta-base-v2* could deal with very long texts because it is a distilled model as it needs less memory than *paraphrase-mpnet-base-v2*. However, *Paraphrase-distilroberta-base-v2* results (f1-score) on the longer datasets were worse than the *Longformer* and required more processing time than the *Longformer* model.

Based on all the above discussion, we recommend employing the *Paraphrase-mpnet-base-v2* model for keyword extraction from short text (less than 1000 token). Also, we recommend employing the *Longformer* model for extract keywords from long text (more than 1000 token).

VII. CONCLUSION

Natural language processing (NLP) makes it possible for machines to deal with human languages. Since the invention of NLP domain, the need to pull structured data from text emerged. In recent years, NLP performs several tasks such as text classification, sentiment analysis, automatic translation, and question answering. One of the most important NLP tasks is keyword extraction, which is tackled and emphasized by several research works. KeyBert is considered as easy-to-use and powerful method for keyword extraction, which has three steps: firstly, the selection of candidate keywords from a text, secondly, the embedding operation of the text and its candidate keywords, and thirdly, the calculation of the cosine similarity between individual keywords vectors and document vector. In this study, we focus on the second step (embedding operation). In KeyBert, this step depends on BERT model embeddings. However, KeyBERT supports many models for generating the BERT embeddings from many libraries such as Flair, Hugginface Transformers, and spaCy. Our study uses the sentence-transformers package and Longformer from Hugginface Transformers.

We conducted this study on five datasets, which include sequences of documents ordered according to their length. These datasets are: SemEval2017, Inspect, 500N-KPCrowd-v1.1, Schutz2008, and SemEval2010. Our study employs five metrics: f1_score, precision, recall, MAP (Mean Average Precision), and Execution time to evaluate the models' performances. MAP is employed because it considers the order of keywords list.

Based on the study, we found that the *Paraphrase-mpnet-base-v2* has the best results but it needs so much time to process the long text. On the other hand, *Paraphrase-distilroberta-base-v2* could deal with very long texts. However, *Paraphrase-distilroberta-base-v2* results (f1-score) on the longer datasets were worse than the *Longformer* and required more processing time than the *Longformer* model. So, we recommend employing the *Paraphrase-mpnet-base-v2* model for keyword extraction from short text (less than 1000 token). Also, we recommend employing the *Longformer* model for extract keywords from long text (more than 1000 token).

For future work, we shall extend our study to consider comparing more models on more datasets in terms of both their effectiveness and efficiency. Also, we may consider comparing the models' performances on specific domains such as in agriculture and computer science.

ACKNOWLEDGMENT

We are grateful to Remocolla Ltd. Japan (http://www.remocolla.co.jp/) and the CEO Dr. Tarek Fatyani for supporting the research experiments.

REFERENCES

[1] T. Pay and S. Lucci. "Automatic keyword extraction: An ensemble method". In *2017 IEEE international conference on big data (big data)*, 2017, pp. 4816-4818.

[2] T. Weerasooriya, N. Perera and S.R. Liyanage. "KeyXtract Twitter model – An essential keywords extraction model for Twitter designed using NLP tools", ArXiv170802912 Cs, 2017.

[3] D. Newman, N. Koilada, J. H. Lau and T. Baldwin. "Bayesian text segmentation for index term identification and keyphrase extraction." In *COLING 2012*, 2012, pp. 2077– 2092.

[4] S. Yang, W. Lu, D. Yang, X. Li, C. Wu, and B. Wei. "KeyphraseDS: Automatic generation of survey by exploiting keyphrase information." *Neurocomputing*, 224, pp.58-70, 2017.

[5] F. Xie, X. Wu and X. Zhu. "Efficient sequential pattern mining with wildcards for keyphrase extraction." *Knowledge-Based Systems*, 115, pp.27-39, 2017

[6] J. R. Thomas, S. K. Bharti and K. S. Babu. "Automatic keyword extraction for text summarization in e-newspapers". In *the international conference on informatics and analytics* , 2016, pp. 1-8.

[7] S. Rose, D. Engel, N. Cramer and W. Cowley, "Automatic keyword extraction from individual documents", *Text mining: applications and theory*, pp.1-20, 2010

[8] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes and A. Jatowt, "Yake! keyword extraction from single documents using multiple local features", *Information Sciences*, 509, pp.257-289, 2020.

[9] R. Mihalcea and P. Tarau, Textrank: Bringing order into text, In *the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404-411.

[10] A. Bougouin, F. Boudin and B. Daille, "TopicRank: Graph-based topic ranking for keyphrase extraction", In *International joint conference on natural language processing (IJCNLP),* 2013, pp. 543-551.

[11] C. Florescu and C. Caragea, "Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents", in *the 55th annual meeting of the association for computational linguistics*, 2017, (volume 1: long papers), pp. 1105-1115.

[12] Y. Sun, H. Qiu, Y. Zheng, Z. Wang and C. Zhang. "SIFRank: a new baseline for unsupervised keyphrase extraction based on pre-trained language model". *IEEE Access*, 8, pp.10896-10906, 2020.

[13] D. Mahata, J. Kuriakose, R. Shah and R. Zimmermann, "Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings", in *the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, Volume 2 (Short Papers), pp. 634-639.

[14] M. Grootendorst, Keybert: Minimal keyword extraction with bert, 2020, http://dx.doi.org/10.5281/zenodo.4461265.

[15] T. Jo and J.H. Lee. "Latent keyphrase extraction using deep belief networks. *International Journal of Fuzzy Logic and Intelligent Systems*, 15(3), pp.153-158, 2015.

[16] Q. Zhang, Y. Wang, Y. Gong and X.J. Huang. "Keyphrase extraction using deep recurrent neural networks on Twitter". In *the 2016 conference on empirical methods in natural language processing*, 2016, pp. 836-845.

[17] N. Giarelis, N. Kanakaris and N. Karacapilidis, " A comparative assessment of state-of-the-art methods for multilingual unsupervised keyphrase extraction". In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, 2021, pp. 635-645. Cham: Springer International Publishing.

[18] M.Q. Khan, A. Shahid, M.I. Uddin, M. Roman, A. Alharbi, W. Alosaimi, J. Almalki and S.M. Alshahrani, "Impact analysis of keyword extraction using contextual word embedding." *PeerJ Computer Science*, 8, e967, 2022.

[19] J. Piskorski, N. Stefanovitch, G. Jacquet and A. Podavini,"Exploring linguistically-lightweight keyword extraction techniques for indexing news articles in a multilingual set-up." In *the EACL Hackashop on News Media Content Analysis and Automated Report Generation*, 2021, pp. 35-44.

[20] A. D'Agostino, "Keyword Extraction—A Benchmark of 7 Algorithms in Python I compared 7 relevant algorithms in a keyword extraction task on a corpus of 2000 documents.", 2023. [Online]. Available: https://towardsdatascience.com/keyword-extraction-a-benchmark-of-7-algorithms-in-python-8a905326d93f

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

[22] I. Beltagy, M.E. Peters and A. Cohan. "Longformer: The long-document transformer" arXiv preprint arXiv:2004.05150, 2020.

[23] N. Reimers, & I. Gurevych, (2019). "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084.