Extraction of keywords in the abstract of scientific papers using BERT

M Ikhsan¹, R F Rachmadi², E M Yuniarno³
¹Department of Electrical Engineering
²Department of Electrical Engineering and Department of Computer Engineering
³Department of Computer Engineering
¹,²,³Institut Teknologi Sepuluh Nopember Surabaya

E-mail: mikhsan.18071@mhs.its.ac.id

Abstract. Keywords are used to facilitate searches, inaccurate keyword determination, which can result in searches for journals that do not match the intended title. To overcome this, it is necessary to implement a keyword search in the abstract using deep learning methods from a journal to produce the right keywords that will distinguish journals from one another. The purpose of this study is to implement a keyword search system in abstracts using the Bidirectional Encoder Representations from Transformers (BERT) method as an abstract word search system that produces keyword answers that have context according to the desired keyword results. The results of this study get optimal results in the epoch 500, and from these results using a dataset with 444 training data and 10 test data for testing.

1. Introduction
Keywords are an integral part of writing scientific articles. Keywords can be taken from the thesaurus according to their respective scientific fields, written in one line with a number varying from 3 to 6 words and sorted from specific to general. Keywords can inform the subject matter of a scientific article. In addition, keywords have an important role in article indexing. Article indexing can help make it easier to find an article through computer scanning on the internet, for example, if someone wants to search for an article, a keyword search can provide results in the form of articles that are relevant to the given keywords. Therefore, the selection of keywords must be done well, so that they can effectively describe the whole subject in a scientific article. In this era of machine learning, choosing effective keywords is possible and easy to do.

Machine learning has developed very rapidly and has been widely applied in various fields of research. One of them is in various text mining processes, such as word identification, word classification, entity extraction, word analysis, and other aspects of the text [1]. In recent years, there have been many studies that have developed text mining algorithms, one of which is the BERT (Bidirectional Encoder Representations from Transformers) algorithm [2]. BERT is a new technique in NLP (natural language processing) created by researchers from Google AI Language. In its creation, Google AI Language can present more modern results in a wide variety of problems in NLP, including Questionnaires, Natural Language Inference (MNLI), and others.

Based on the success of the research above, this study implements the BERT method in searching for abstracted keywords in scientific articles. The dataset used in this research is in the form of
abstracts in scientific journals obtained from the semaval 2010 dataset and added with data from competent journals. Using the method of BERT, this research is expected to provide the best results of research using the NLP method, especially in the case of keyword searches.

2. Related Works
The approach describes related works for aspect extraction using deep learning algorithms for keywords and BERT methods based on study literature. In this case, the approach to describe this problem is using a deep learning algorithm. In deep learning, many methods can be developed in more depth. In its journey, the method can be trained to be able to classify the desired case. Deep learning classification is very dependent on big data to get optimal performance. Bidirectional Encoder Representations from Transformers (BERT) are capable and effective on a large number of datasets but will lack significant performance on a small number of trained datasets. In the study of M. R. Yanuar [3], adapting BERT to handle aspect extraction ABSA's aspect-based sentiment analysis assignment for Indonesian, especially in the tourist area domain. In M. Ramina et.al. [4], work implements a topic level with a level where the topic can be an idea, concept, or term that the user wants to know. Their research used NLP capabilities of the bidirectional encoder representation transformers (BERT) language model for generating a topic level.

In previous work of J. Yadav et.al [5] researched to calculate bullying comments for proposed solutions on social media based on the BERT model as a pre-trained BERT model with a single linear neural network layer on top as a classifier, which improves over the existing results. Furthermore, in Y. Iwasaki et.al [6] work developed an automatic abstractive text summarization algorithm. Utilizing BERT to evaluating the model and also highlight future issues that could arise from the text generated as a result of training in the Japanese corpus. K. Ki et.al [7] proposed KoTAB (Korean Template-Based), a Korean template-based arithmetic solver with BERT, which is modified from TAB (Template-based Arithmetic Solver with BERT) framework that is used to solve English math word problems. KoTAB also solved Korean math word problems using two datasets, CC_Ko and IL_Ko, which were translated from English to Korean. He used four pre-trained BERT models that were used to fine-tune KoTAB; BERT-based, BERT-multilingual, Korean BERT WordPiece, and Korean BERT Morphology.

X. Zhao et.al [8] The combination model performance by combining BERT with several conversation models. The model investigates a better solution to using BERT: fixed weights or fine-tuning. Meanwhile, this paper describes new baseline methods based on BERT for the conversation generation task. We evaluate the proposed BERT-based models and baseline models on two popular dialog datasets: Ubuntu Dialog and Cornell Movie Dialog. The experiment results prove the effectiveness of incorporating BERT into conversation generation models.

3. Material and Methods
3.1. Dataset
In this study using the semaval 2010 dataset in its development the dataset is added by taking from other competent journal datasets, additional collections of other journals obtained from IEEE, in the dataset, there are 2 different file types, the first is a collection of journal abstracts. Scientific data and the second data is in the form of keywords from the abstract, the total scientific journal data used is 564 journals, which will be divided into 444 training (400 for train data, 44 for validation) and 10 testing data, for testing data obtained from IEEE journal.

3.2. BERT
BERT is a pre-trained language model which comprises a set of transformer encoders which represents the text at word and sentence level with the help of unsupervised training techniques like masked language modeling and next sentence prediction. Being a pre-trained model, BERT trained on 3300M words. BERT uses a transformer encoder with an attention mechanism to learn contextual relations between words. Transformer [9] in their native form consists of the encoder as well as
decoder where the encoder learns the text input, and the decoder is tuned to conduct a specific task. As a language model, understanding the input text is the only important factor. Due to this reason, only transformer encoders are used in BERT. Rather than reading the text input sequentially like various directional models [9], the transformer encoder reads the entire sequence of words at once.

The BERT base model can be fine-tuned for text classification by simply adding a softmax classification layer on top, so it will predict the class of a given text sequence. It can be seen in Equal q.

\[ p(c|H) = \text{softmax}(WH) \]  

\[ Z = \text{softmax} \left( Q \times \frac{K^T}{\sqrt{d_k}} \right) V \]

Where Q is query matrix, K is a key matrix, V is a value matrix, \( d_k \) is the dimension of the key vector, and Z is the attention score matrix of a single head. The attention score matrix is generated for every attention head, where score matrices are combined, and then reduce every input dimension of the feed-forward layer.

This functionality of the transformer helps generate context by calculating the relevance of each word concerning the presence of other words in the sentence. The level of contextual understanding is directly proportional to several transformer encoder layers. Fig. 1 shows BERT’s architecture. BERT-Base, Uncased is used for the purpose which has 12 transformer layers, 768 hidden state, 12 attention heads, 110M parameters.

![Figure 1. Attention-based model system design.](image)

3.2.1. Fine Tunning BERT.
Understanding the input text is the only important factor in the BERT model because of the transformation of the encoder. Furthermore, the need in this work requires a transformer decoder. So, it will help this work to convert contextual embedding. The encoder is a pre-trained BERT language model, it takes less training time than a decoder.
To make BERT adapt to a target task, it needs several factors, such as the preprocessing of long text since the maximum sequence length of BERT is 512, layer selection factor, and overfitting problem. A better optimizer with an appropriate learning rate will be desired. The lower layer of the BERT model may contain some information. So it can be fine-tuned with different learning rates. Following Howard and Ruder (2018) method, it split the parameters $\theta$ into $[\theta_1^l, \ldots, \theta_t^l]$ where $\theta_t^l$ contains the parameters of the $l$-th layer of BERT. Then the parameters are updated as:

$$
\theta_t^l = \theta_{t-\xi}^l - \eta^l \nabla_{\theta_t^l} J(\theta)
$$

(3)

Where $\eta^l$ represents the learning rate of the $l$-th layer. And then set the base learning rate to $\eta_b$ and use $\eta_{t-\xi}^l = \xi^\xi \eta_b^l$, where $\xi$ is a decay factor and less than or equal to 1. But when $\xi<1$, the lower layer has a slower learning rate than the higher layer. If $\xi=1$ all layers have the same learning rate, so it will be equivalent to the regular stochastic gradient descent (SGD).

3.2.2. Further Pre-Tunning.
The BERT model is pre-trained for the general domain corpus. For the role of classifying text in a particular domain, such as movie reviews, the distribution of data may be different from BERT. Therefore, we can further pre-train BERT with masked language model and next sentence prediction tasks for domain-specific results. Three additional pre-training approaches are carried out:

1. Within-task pretraining, in which BERT is further pre-trained on the training data of a target task.
2. Pre-training in-domain, in which pre-training data is derived from the same domain of the aim mission. For example, there are many different sentiment classification tasks that have a similar distribution of data. We will further pre-train BERT on the cumulative training data for these tasks.
3. Cross-domain pretraining in which pre-training data is obtained from both the same and separate domains for the target task. We will discuss these diverse approaches to further pretraining.

3.2.3. Fine Tuning BERT.
Multitask Learning is also an important approach to knowledge exchange from a variety of various supervised tasks. Like [10], BERT is also used for text recognition in a multi-task learning system. Both functions are exchanged between the BERT layers and the embedding layer. The only layer that
does not shift is the final classification layer, which means that each work has a private classifier layer.
Experimental analysis is being done in Sec. 4.

**4. Result and Discussion**

In the experiment, classification testing was carried out using the BERT method and for this test using the 2010 SemEval dataset with the entire dataset used in this study were 444 training (400 for training data, 44 for validation) and 10 test data for testing.

Performance in the classification is evaluated using performance metrics based on Train loss, validation accuracy, validation loss, and F1-score. The model with input data from the validation dataset. For validation accuracy, the calculated accuracy value of the validation dataset and prediction of the model with data input from the validation dataset and F1-score are used to calculate the average accuracy and recall.

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (4)
\]
\[
\text{Precision} = \frac{TP}{TP + FP} \quad (5)
\]
\[
F1 - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)
\]

In building a strong word classifier, large amounts of data are needed to get good work results in the training process of this study using 500 epochs. The training results are shown in Table 1:

| epoch | validation Accuracy | validation loss | F1-score | Train loss |
|-------|--------------------|-----------------|----------|-----------|
| 500   | 0.96081            | 0.34035         | 0.58504  | 0.000868  |

To find out the toughness of the model, the training result model at 500 epochs was tested on 10 testing data, and calculated using equation (7). The test results show that the model obtains an average accuracy of 55.01%. The accuracy of the test results of each test data shown in Figures 5 and Figure 6 counted using this formula:

\[
\text{Accuracy Test BERT} = \frac{\text{Success Rate}}{\text{count ground truth}} \times 100\% \quad (7)
\]

The following is an example of the test results on one of the testing data "Achieving Reliable Sentiment Analysis in the Software Engineering Domain using BERT".
### Table 2. Testing Result Each Testing Data

| NO | Testing abstract | Accuracy |
|----|------------------|----------|
| 1  | 10.1109/ICSME46990.2020.00025 | 66.7% |
| 2  | 10.1109/ACCESS.2020.3030468 | 50.0% |
| 3  | 10.1109/TALE48000.2019.9225993 | 50.0% |
| 4  | 10.1109/ICSME46990.2020.00025 | 66.7% |
| 5  | 10.1109/IColICT49345.2020.9166407 | 50.0% |
| 6  | 10.1109/BIBM47256.2019.8983370 | 60.0% |
| 7  | 10.1109/ICAIBD.2019.8837002 | 66.7% |
| 8  | 10.1109/IUCC/DSCI/SmartCNS.00132 | 40.0% |
| 9  | 10.1109/BisData47090.2019.9006511 | 50.0% |
| 10 | 10.1109/ITNEC48623.2020.9085059 | 50.0% |
|     | Average accuracy | 55.01% |

Accuracy Test BERT = \( \frac{2}{5} \times 100\% = 66.7\% \)

The success rate is obtained from the calculation of the success rate and divided by the total number of ground truth keywords and multiplied by 100%. Example of Bert's Test Results from one of the scientific journal test data with a test accuracy of 66.7%. The ground truth shown in Figure 4 and Figure 5 is the result of testing using models trained for 500 epochs.

**Figure 4. Ground Truth**

**Input text**

Researchers have shown that sentiment analysis of software artifacts can potentially improve various software engineering tools, including API and library recommendation systems, code suggestion tools, and tools for improving communication among software developers. However, sentiment analysis techniques applied to software artifacts still have not yet yielded very high accuracy. Recent adaptations of sentiment analysis tools to the software domain have reported some improvements, but the f-measures for the positive and negative sentences still remain in the 0.4-0.64 range, which deters their practical usefulness for software engineering tools. In this paper, we explore the potential effectiveness of customizing BERT, a language representation model, which has recently achieved very good results on various Natural Language Processing tasks on English texts, for the task of sentiment analysis of software artifacts. We describe our application of BERT to analyzing sentiments of sentences in Stack Overflow posts and compare the impact of a BERT sentiment classifier to state-of-the-art sentiment analysis techniques when used on a domain-specific data set created from Stack Overflow posts. We also investigate how the performance of sentiment analysis changes when using a much (3 times) larger data set than previous studies. Our results show that the BERT classifier achieves reliable performance for sentiment analysis of software engineering texts. BERT combined with the larger data set achieves an overall f-measure of 0.87, with the f-measures for the negative and positive sentences reaching 0.91 and 0.78 respectively, a significant improvement over the state-of-the-art.

**Output text**

sentiment 0

analysis 1

bert 0
Researchers have shown that sentiment analysis of software artifacts can potentially improve various software engineering tools, including API and library recommendation systems, code suggestion tools, and tools for improving communication among software developers. However, sentiment analysis techniques applied to software artifacts still have not yet yielded very high accuracy. Recent adaptations of sentiment analysis tools to the software domain have reported some improvements, but the f-measures for the positive and negative sentences still remain in the 0.4-0.64 range, which deters their practical usefulness for software engineering tools. In this paper, we explore the potential effectiveness of customizing BERT, a language representation model, which has recently achieved very good results on various Natural Language Processing tasks on English texts, for the task of sentiment analysis of software artifacts. We describe our application of BERT to analyzing sentiments of sentences in Stack Overflow posts and compare the impact of a BERT sentiment classifier to state-of-the-art.

5. Conclusion
In this paper, an extensive experiment was conducted to investigate the approach to BERT, the task of text classification. There are several experimental findings: 1) The top layer of BERT is more useful for text classification; 2) With an appropriate layer-wise decreasing learning rate, BERT can overcome the catastrophic forgetting problem; 3) In a training experiment, we can conclude that the greater the data and the smaller the training loss; 4) From the results of the training trials on the epoch test 500, satisfactory test results are obtained in the training results by getting a validation Accuracy of 96.08%. 5) From the epoch 500 extract scientific journals and get a model from the training results. These results were tested with 10 abstract data. From the text results, calculations are made to get the results of the success rate.

6. References
[1] Tang M, Gandhi P, Kabir M A, Zou C, Blakey J and Luo X 2019 Progress Notes Classification and Keyword Extraction using Attention-based Deep Learning Models with BERT. arXiv [Online]. Available: http://arxiv.org/abs/1910.05786
[2] Devlin J, Chang M W, Lee K and Toutanova K 2019 BERT: Pretraining of deep bidirectional transformers for language understanding. NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf 1 pp 4171 - 4186
[3] Yanuar M R and Shiramatsu S 2020 Aspect extraction for tourist spot review in Indonesian language using BERT. Int. Conf. Artif. Intell. Inf. Commun. ICAIIC pp 298 - 302 doi: 10.1109/ICAIIIC48513.2020.9065263
[4] Ramina M, Darnay N, Ludbe C and Dhruv A 2020 Topic level summary generation using BERT induced abstractive summarization model. Proc. Int. Conf. Intell. Comput. Control Syst. ICICCS 2020. ICICCS pp 747 - 752 doi: 10.1109/ICICCS48265.2020.9120997
[5] Yadav J, Kumar D and Chauhan D 2020 Cyberbullying detection using pre-trained BERT model. Proc. Int. Conf. Electron. Sustain. Commun. Syst. ICESC 2020. ICESC pp 1096 - 1100 doi: 10.1109/ICESC8915.2020.9155700
[6] Iwasaki Y, Yamashita A, Konno Y and Matsubayashi K 2019 Japanese abstractive text summarization using BERT. Proc. - 2019 Int. Conf. Technol. Appl. Intell. TAAI 2019 pp 1096 - 1100 doi: 10.1109/TAAI48200.2019.8959920
[7] Ki K, Lee D and Gweon G 2020 KoTAB: Korean template-based arithmetic solver with BERT. Proc. - 2020 IEEE Int. Conf. Big Data Smart Comput. BigComp 2020 pp 279 - 282 doi: 10.1109/BigComp48618.2020.00-61
[8] Zhao X, Zhang Y, Guo W and Yuan X 2019 BERT for open-domain conversation modeling.
IEEE 5th Int. Conf. Comput. Commun. ICCC 2019 pp 1532 -1536 doi: 10.1109/ICCC47050.2019.9064414

[9] Sutskever I, Vinyals O and Le Q V 2014 Sequence to sequence learning with neural networks. Adv. Neural Inf. Process. Syst 4 January 2014 pp 3104 - 3112

[10] Liu X, He P, Chen W and Gao J 2020 Multi-task deep neural networks for natural language understanding. ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf pp 4487 - 4496