Large-Scale News Classification using BERT Language Model: Spark NLP Approach

Kuncahyo Setyo Nugroho
Department of Informatics Engineering, Faculty of Computer Science, Brawijaya University, Indonesia
ksnugroho26@gmail.com

Anantha Yullian Sukmadewa
Department of Informatics Engineering, Faculty of Computer Science, Brawijaya University, Indonesia
ananthayullian@gmail.com

Novanto Yudistira
Department of Informatics Engineering, Faculty of Computer Science, Brawijaya University, Indonesia
yudistira@ub.ac.id

ABSTRACT

The rise of big data analytics on top of NLP increasing the computational burden for text processing at scale. The problems faced in NLP are very high dimensional text, so it takes a high computation resource. The MapReduce allows parallelization of large computations and can improve the efficiency of text processing. This research aims to study the effect of big data processing on NLP tasks based on a deep learning approach. We classify a big text of news topics with fine-tuning BERT used pre-trained models. Five pre-trained models with a different number of parameters were used in this study. To measure the efficiency of this method, we compared the performance of the BERT with the pipelines from Spark NLP. The result shows that BERT without Spark NLP gives higher accuracy compared to BERT with Spark NLP. The accuracy average and training time of all model's using BERT is 0.9187 and 35 minutes while using BERT with Spark NLP pipeline is 0.8444 and 9 minutes. The bigger model will take more computation resources and need a longer time to complete the tasks. However, the accuracy of BERT with Spark NLP only decreased by an average of 5.7%, while the training time was reduced significantly by 62.9% compared to BERT without Spark NLP.

CCS CONCEPTS

• **Computing methodologies** → Artificial intelligence; Natural language processing; Parallel computing methodologies; Parallel algorithms; MapReduce algorithms.

KEYWORDS

Large-scale text classification, distributed NLP architectures, BERT language model, Spark NLP

ACM Reference Format:
Kuncahyo Setyo Nugroho, Anantha Yullian Sukmadewa, and Novanto Yudistira. 2021. Large-Scale News Classification using BERT Language Model: Spark NLP Approach. In *6th International Conference on Sustainable Information Engineering and Technology 2021 (SIET '21), September 13, 14, 2021, Malang, Indonesia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3479645.3479658

1 INTRODUCTION

Natural language processing (NLP) is a subfield of artificial intelligence (AI) that can study human and computer interactions through natural languages, such as the meaning of words, phrases, sentences, and syntactic and semantic processing. NPL researches were using rule-based methods to acknowledge and rationing a text in the earlier times. Experts manually create these rules for various NLP tasks [37]. It is complicated to manage the rules if the number of rules is enormous. Therefore, this approach is considered obsolete by researchers [9]. Internet development causes data to be collected easily so that a statistical learning approach is possible to resolve NLP tasks. This approach is known as machine learning. With feature engineering, machine learning brings significant improvements to many NLP tasks [44]. Meanwhile, deep learning approaches were introduced to NLP in 2012 after successful image recognition [14] and speech recognition [43]. Deep learning outperformed the other approaches with surprisingly better results.

In NLP, language modeling (LM) provides a context that differentiates similar words and phrases according to the context in which they appear. The NLP framework based on deep learning for language modeling has entered a new chapter. This is characterized by many deep learning architectures and models from which to solve NLP tasks is constantly evolving. Previously successful architecture is bidirectional LSTM (bi-LSTM) based on recurrent neural network (RNN), where the model can read the context from left to right and from right to left [48]. The main limitations of bi-LSTM are sequential, which makes the parallel training process very difficult. The Transformer architecture accomplishes this by replacing the LSTM cells with an "attention" mechanism [41]. With this attention, the model can see the entire sequence of context as a whole, making it easier to practice in parallel. The Transformer has made great progress on many different NLP benchmarks. There are many transformer-based language models, including BERT [13], RoBERTa [24] GPT-2 [5] and XLNet [45].

The rise of "big data" analytics on top of NLP has led to an increasing need to ease the computational burden that processes text at scale [3]. The amount of unstructured textual data has led to increased interest in information extraction technology from academia and industry. One of the problems faced in NLP is that text has very high dimensions [42]. It takes computation capable of processing high-dimensional textual data quickly. Input data is distributed across multiple machine clusters to complete within a reasonable time. The MapReduce allows easy parallelization of large computations and uses re-execution as the primary mechanism for fault tolerance [12]. Previous research has used this concept to perform sentiment analysis tasks. The results obtained are that MapReduce can improve the efficiency of processing large amounts of text even though the performance obtained is similar to traditional sentiment analysis [17].

This research aims to study big data processing on NLP tasks based on a deep learning approach. We classify large amounts of news topics using BERT based on transformer architecture. Training BERT from scratch requires a huge dataset and takes much time to train. Therefore, we use the existing pre-trained models [13, 40]. To demonstrate the efficiency of this method, we conducted extensive experiments to study our proposed approach. We use Spark NLP built on top of Apache Spark as a library that can scale the entire classification process in a distributed environment [22]. We compared the performance of the base method model with the classifier pipelines from Spark NLP. Apart from observing the model's accuracy, we also look at the computation time and computation resources used during the training and testing process.

2 RELATED WORK

Big data comes with an unstructured format, mainly textual data, called big text [38]. Social media has the most contribution to a big text. In addition, other online sources such as online news portals, blogs, health records, government data provide rich textual data for research. Despite the abundance of data sources, this field has attracted less attention from academia. In this section, we present literature studies carried out in the fields of deep learning for text classification and big data framework for large-scale text processing. We reviewed prior work to understand its limitations so that we can use them to refine our research.

Deep learning gives us big potential in the NLP field [11]. Many studies have contributed to text classification tasks using deep neural networks. Some successful architectures include convolutional neural network (CNN) based models, for example, VD-CNN [10] and DP-CNN [20], recurrent neural network (RNN) based models, for example, SANN [23], and attention-based models, for example, HAN [46] and DiSAN [36]. These models use pre-trained word embedding [28, 31] to improve performance in downstream tasks. Although many impressive results have been achieved, the dependent problem carries many limitations for enhancing the model's performance. Even with the development of contextualized word vectors such as CoVe [26] and ELMo [33], the model architecture still needs to be assigned in particular. Pre-training language models and fine-tuning of downstream tasks have made breakthroughs in NLP. Howard and Ruder proposed ULMFiT [19], whereas Radford et al. proposed OpenAI GPT [34] using a multi-layer transformer architecture to learn language representations of large-scale text. To solve unidirectional language representation from OpenAI GPT, Devlin et al. proposed BERT [13] using deep bidirectional representations. Compared to the previous model, BERT does not require a specific architecture for each downstream task, so this model has achieved great success in many NLP downstream tasks [2].

Hadoop is a MapReduce platform used for distributed processing. One of the Hadoop framework's major problems is that it transforms any computation as a MapReduce job [3]. In NLP, this would require re-implementation of each NLP pipeline, so it is ineffective. Apache Spark addresses this problem by extending Hadoop ecosystem with a parallel computational programming model, including resilient distributed datasets (RDDs) and learning algorithms [47]. Next, Xiangrui et al. introduced Apache Spark MLib [29] as a machine learning library running on Spark [27]. Research from Jian et al. analyzed the Spark framework by running a machine learning instance using MLib and highlighting Spark's advantages [15]. Spark is also used as a distributed framework for solving NLP tasks such as sentiment analysis [4, 32], and document classification [35]. Their research results show that Spark has a speed advantage in large text processing. As deep learning models have successfully in NLP, there is a need to implement pre-trained models and scale large data with distributed use cases. John Snow Labs [30] developed Spark NLP as a library built on top of Apache Spark and Apache MLib that provides an NLP pipeline and pre-trained models [22]. The library offers the ability to train, customize and save models so they can be run on clusters, other machines, or stored.

3 METHODOLOGY

3.1 Dataset

We use a corpus of news articles from the AG dataset [16]. It contains 1 million news articles that have been gathered from more than 2000 new sources from ComeToMyHead news search engine. This dataset includes 120,000 training samples and 7,600 test samples. We only use the description as a sample and category as the label. Each sample is a short text divided into four labels.

3.2 BERT

BERT stands for Bidirectional Encoder Representations from Transformers is a deep learning architecture that can be used for downstream NLP tasks. The architecture consists of a stacked encoder layer from the Transformer [41]. BERT takes a distinctive approach to learning. Bidirectional means that BERT learns from the left and right sides of the token during learning. A bidirectional method is essential to understand the meaning of language. There are two main steps in BERT: pre-training and fine-tuning [13]. During pre-training, BERT is trained in a large unlabeled corpus with two unsupervised tasks: masked language model (MLM) and next sentence prediction (NSP) to produce a pre-trained model. For fine-tuning, the model is initialized with the pre-trained parameters, and all the parameters are fine-tuned using labeled data for specific tasks such as classification.

We can assume the pre-trained model as a black box with $H = 768$ shaped vectors for each input token in a sequence. Sequences can be one sentence or a pair of sentences separated by a [SEP] token and begin with a [CLS] token. For classification task, we added an output layer to model and fine-tuned all parameters from end to end. In practice, we only use the output from the [CLS] token as the representation of the whole sequence. Thus, the entire fine-tuning BERT architecture for the classification task is shown in Figure 1. A simple SoftMax classifier is added to the top of the model to predict the probability of label c shown in Equation 1.

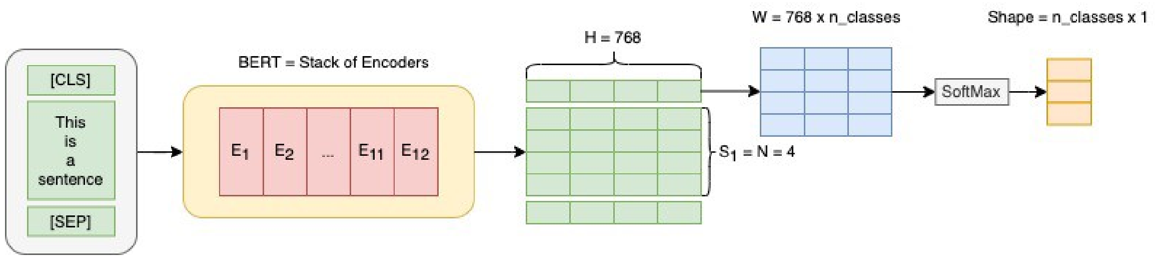

Figure 1: Fine-tuning BERT architecture for the classification task. We just use the [CLS] output token for classification along with some added Linear and SoftMax layers.

Table 1: Pre-trained BERT models are used. We only focus on six models: Tiny (L=2, H=128), Mini (L=4, H=256), Small (L=4, H=512), Medium (L=8, H=512), and Base (L=12, H=768).

	H=128	H=256	H=512	H=768
L=2	BERT-Tiny	-	-	-
L=4	-	BERT-Mini	BERT-Small	-
L=8	-	-	BERT-Medium	-
L=12	-	-	-	BERT-Base

Table 2: The number of parameters on the pre-trained BERT model.

Model	Parameters (Millions)
BERT-Tiny	4.4
BERT-Mini	11.3
BERT-Small	29.1
BERT-Medium	41.7
BERT-Base	110.1
BERT-Large	340

Where W is the task-specific parameter matrix. We fine-tune all the parameters from BERT as well as W jointly by maximizing the log-probability of the correct label.

$$p(c|h) = softmax(Wh) \tag{1}$$

In this study, we use five pre-trained models as shown in Table 1. In the original paper, L represents the numbers of transformer layers (stacked encoder), H represents numbers of hidden embedding size, and A represents numbers of attention heads [13]. Smaller model architecture is using less parameters to train and can be used in limited computation resources. The number of parameters in every pretrained model shown in Table 2.

The model hyperparameters for fine-tuning are the same as in the pre-training, except for the batch size, learning rate, and a number of training epochs. The optimal hyperparameter values are task-specific. In the original BERT paper [13], the range of values for the batch size was 16 or 32, while the number of epochs was 2, 3, or 4. We used a batch size of 32 to ensure that the GPU memory is fully utilized. Then, we use 4 epochs to pass the complete dataset. We do not use a learning rate that is too large or too small. When the learning rate is too large, gradient descent can inadvertently increase training errors. When the learning rate is too small, training is not only slower because it can also be permanently stuck with high training errors. Therefore, we use a learning rate of 1e-4 and the warm-up proportion is 0.1. We also use Adam optimizer with $\beta 1 = 0.9$ and $\beta 2 = 0.999$.

3.3 Spark NLP

Due to the popularity of NLP in recent years, many NLP library have been developed, such as Natural Language Toolkit (NLTK) [7], SpaCy [39], TextBlob [6], Gensim [18], FastText [8, 21], and Stanford Core NLP [25]. Some of these are only optimized to work on a single node machine and not designed for distributed environments or parallel computing. In addition, recent deep learning models such as BERT have made significant changes to NLP because they can be fine-tuned and reused without major computational effort. A new library, Spark NLP, was introduced to meet the need for scalable, high-performance, and high-accuracy text processing. Spark NLP is an open-source library built on top of Apache Spark and Spark ML [17]. Apache Spark is a component of the Hadoop ecosystem, a favorite big data platform because of its ability to process streaming data.

In this study, we used text processing and word embedding from the BERT pre-trained model to build a text classification model in Spark NLP. Each stage in the Spark NLP is implemented in a pipeline as a sequence, as shown in Figure 2. Each resulting output is directed to the next stage as input. This means that the DataFrame (DF) input will be changed as it passes through each stage. First, DF is fed to DocumentAssembler() to generate document fields as starting points in Spark NLP. Then the document column is inserted into SentenceDectector() to be split into an array of sentences and generate a sentence column. The sentence column is inserted into Tokenizer() to generate a word token for the entire sentence and

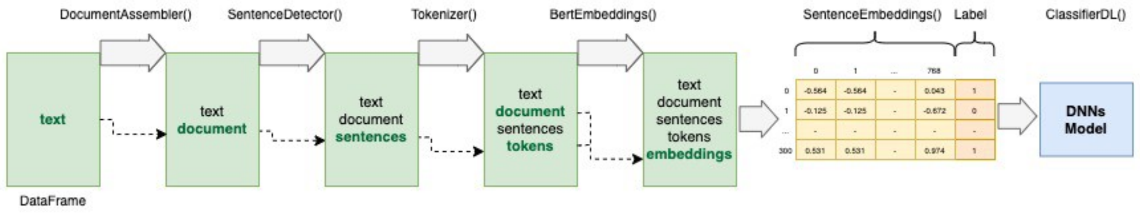

Figure 2: Spark NLP pipeline as a sequence for text classification. Each annotator applied adds a new column to a DataFrame that is fed into the pipeline.

generate the token column. A token column is fed to BertEmbeddings() to convert the token into a vector representation. We use the BERT pre-trained model as explained in the previous section. Finally, we use Sentence Embeddings() to train a model.

To create a classifier in Spark NLP, we use ClassifierDL. ClassifierDL is a multi-class text classifier in Spark NLP, and it uses various text embeddings as an input for text classifications. The ClassifierDL uses a deep learning model (DNNs) built inside TensorFlow [1]. The classification process is carried out after going through the text processing stages before. Spark NLP will write the training logs to annotator_logs folder in our directory.

4 RESULT AND DISCUSSION

Our experiment compares basic BERT without Spark NLP and uses pipelines in Spark NLP to classify large text data. We run all models in Google Colab[1] containing 1 GPU Tesla P100 16 GB and 27.4 GB RAM. For the experiment environment, we use Python version 3.8, Spark NLP version 2.7.5, Apache Spark version 2.3.0, OpenJDK version 1.8.0_292, and TensorFlow version 2.4.1. We measure the resource needed when running all five pre-trained models in Table 1. In addition, we also measure the accuracy performance of the model on the testing data. We observe the GPU and RAM resources used during the training process. To track and observe during the model training process, we use the Weights & Biases[2] library running in our environment.

The first experiment is performed using the BERT model without Spark NLP. The first computing resources test results are shown in Figure 3. The BERT-Large model cannot run in our environment because it needs a higher specification than the one GPU we use in our experiment. According to the comparison shown in Figure 3, BERT-Base takes the most resources and needs a longer time to complete the training, and BERT-Tiny requires the least number of resources and completes the training the fastest. The result shows the bigger the model, the greater the GPU usage and memory allocation needed. Moreover, the bigger the model, the longer time it takes to complete model training.

In the next experiment, we used the pipeline from Spark NLP and added the embedding from the pre-trained BERT model to generate the word embedding. Model development using Spark NLP pipeline is relatively easy and fast compared to the BERT model from scratch. The results of testing computation resources when using Spark NLP are shown in Figure 4. The result is similar to the previous experiment, the bigger the model, the greater the GPU usage and memory allocation are needed. And the bigger the model is, the longer time to complete model training. This is because large models have a larger number of parameters to fine-tune. From the computation resource comparison shown in Figure 3 and Figure 4, we can see that BERT without Spark NLP needs the least number of resources and can complete the training much faster than BERT without Spark NLP.

BERT without Spark NLP and BERT with Spark NLP gives different results, as shown in Table 3. The highest accuracy of BERT without Spark NLP is 0.9253 by BERT-Base. This is not much different from the BERT-Small of 0.9213. Meanwhile, the lowest accuracy when using BERT-tiny is 0.9104. Thus, we did not see a significant increase in accuracy across the pre-trained BERT models without the Spark NLP with an average accuracy of 0.9187. The fastest training time using BERT-Tiny is 11 minutes. Meanwhile, the longest training time when using BERT-Base shows more than 1 hour. The average training time using BERT without spark NLP is 25 minutes. Similar to the previous experiment, the lowest accuracy was when using BERT-tiny at 0.8444, while the highest accuracy was obtained using BERT-base at 0.8665. The different results showed that accuracy continues to increase when using all pre-trained models using Spark NLP pipeline. Except for the BERT-Base medium, smaller than the BERT-Small. The average accuracy obtained for BERT with spark NLP is 0.8665. The average training time using BERT with spark NLP is 9 minutes.

All experiments show that BERT without Spark NLP gives a higher accuracy rate than BERT with Spark NLP on all pre-trained models. However, BERT with Spark NLP has an advantage of efficiency. As shown in Table 3, BERT with Spark NLP gives good accuracy, but it takes less time to complete the task. There was a significant decrease in computation time when using BERT with Spark NLP by 62.9% but decreasing accuracy by 5.7%. Even though using Spark NLP, the RAM resources used are much higher, we can see the efficiency of this method.

5 CONCLUSION

The BERT model is a good model to do large-scale NLP tasks such as news classification. The larger the model, will gives higher accuracy, but it will take more time to complete the task. The bigger the dataset we use to train and test the model, the more time it takes to complete the task. Using Spark NLP gives us advantages when using a BERT-Large model and processing large amounts of data. This study found that using BERT with Spark NLP is more efficient than using BERT without Spark NLP. Using BERT with Spark NLP, the

[1]https://colab.research.google.com.
[2]https://wandb.ai.

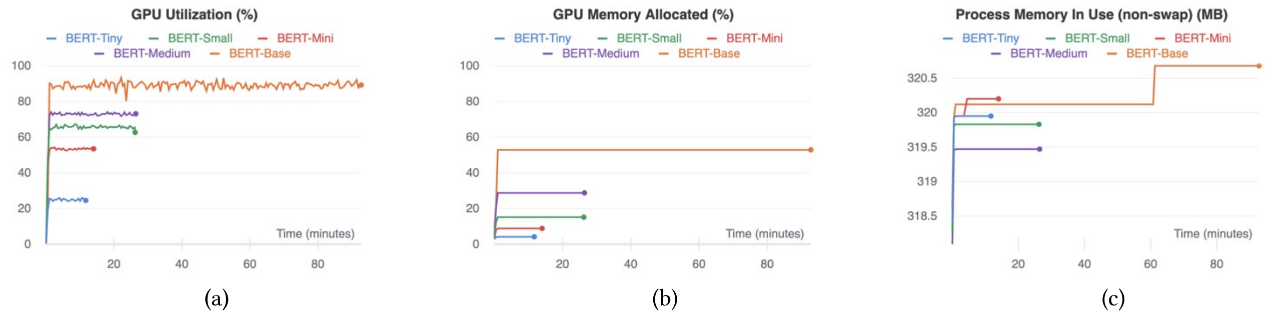

Figure 3: The computational resources used during the training use the BERT without Spark NLP pipeline. (a) GPU utilization, (b) GPU memory allocated, and (c) process memory in use.

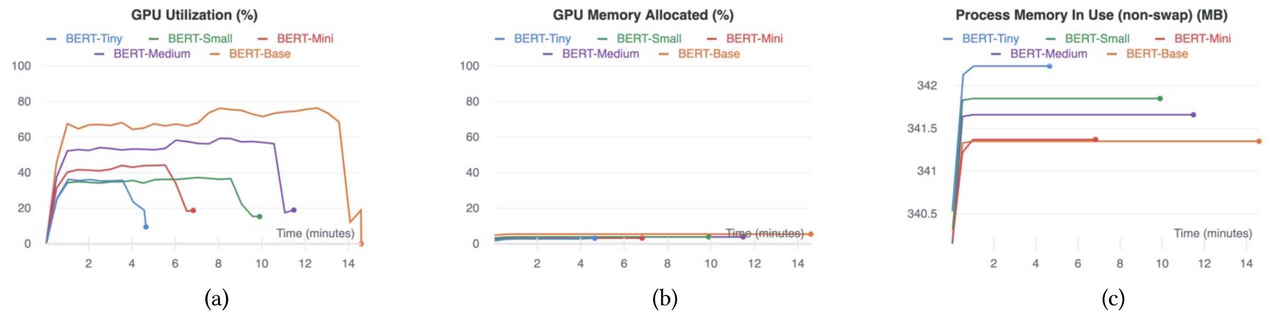

Figure 4: The computational resources used during the training use BERT with Spark NLP pipeline. (a) GPU utilization, (b) GPU memory allocated, and (c) process memory in use.

Table 3: Comparison of accuracy and computation time during the training process between the BERT without Spark NLP and BERT with Spark NLP pipelines. We also calculate the reduction in accuracy and computation time (in percent) to determine the effectiveness of the proposed pipeline.

	BERT		BERT + Spark NLP		Decrease in Accuracy (%)	Decrease in Time (%)
	Accuracy	Wall Time	Accuracy	Wall Time		
BERT-Tiny	0.9104	00:11:41	0.8444	00:04:36	7.2	60.6
BERT-Mini	0.9168	00:13:59	0.8567	00:06:49	6.6	51.3
BERT-Small	0.9213	00:26:13	0.8714	00:10:17	5.4	60.8
BERT-Medium	0.9199	00:26:24	0.8710	00:11:28	5.3	56.6
BERT-Base	0.9253	01:38:12	0.8893	00:14:35	3.9	85.1
Average	0.9187	00:35:18	0.8665	00:09:33	5.7	62.9

drop accuracy average is 5.7%, and the training time drop average is 62.9% compared to BERT without Spark NLP. This study shows a significant reduction in computational time when fine-tuning BERT using the concept of parallelization computing using Spark NLP. We plan to expand and improve our framework by exploring more architectures and pre-trained models to improve classification performance and computational resources. Furthermore, we wanted to explore the effects of text preprocessing before training.

REFERENCES

[1] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. (March 2016). Retrieved August 11, 2021 from https://arxiv.org/abs/1603.04467v2
[2] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. DocBERT: BERT for Document Classification. (April 2019). Retrieved from https://arxiv.org/abs/1904.08398
[3] Rodrigo Agerri, Xabier Artola, Zuhaitz Beloki, German Rigau, and Aitor Soroa. 2015. Big data for Natural Language Processing: A streaming approach. Knowledge-Based Systems 79, (May 2015), 36–42. DOI:https://doi.org/10.1016/j.knosys.2014.11.007
[4] Samar Al-Saqqa, Ghazi Al-Naymat, and Arafat Awajan. 2018. A Large-Scale Sentiment Data Classification for Online Reviews Under Apache Spark. Procedia Computer Science 141, (2018), 183–189. DOI:https://doi.org/10.1016/j.procs.2018.10.166

[5] Ilya Sutskever Alec Radford , Jeffrey Wu , Rewon Child , David Luan , Dario Amodei. 2019. Language Models are Unsupervised Multitask Learners. OpenAI Blog (2019). Retrieved from https://openai.com/blog/better-language-models

[6] Shen Ao. 2018. Sentiment Analysis Based on Financial Tweets and Market Information. In 2018 International Conference on Audio, Language and Image Processing (ICALIP), IEEE, 321–326. DOI:https://doi.org/10.1109/ICALIP.2018.8455771

[7] Steven Bird. 2006. NLTK. In Proceedings of the COLING/ACL on Interactive Presentation Sessions, Association for Computational Linguistics, 69–72. DOI:https://doi.org/10.3115/1225403.1225421

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics 5, (2017), 135–146. DOI:https://doi.org/10.1162/tacl_a_00051

[9] Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! Long live rule-based information extraction systems! Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing October (2013), 827–832. Retrieved from https://aclanthology.org/D13-1079

[10] Alexis Conneau, Holger Schwenk, Yann Le Cun, and Löc Barrault. 2017. Very deep convolutional networks for text classification. Proceedings of 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 1, 2001 (2017), 1107–1116. DOI:https://doi.org/10.18653/v1/e17-1104

[11] Jin Dai and Chong Chen. 2020. Text classification system of academic papers based on hybrid Bert-BiGRU model. Proceedings of 2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2020 2, (2020), 40–44. DOI:https://doi.org/10.1109/IHMSC49165.2020.10088

[12] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. Association for Computing Machinery 51, January 2008 (2008), 107–113. DOI:https://doi.org/10.1145/1327452.1327492

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (October 2019). Retrieved from http://arxiv.org/abs/1810.04805

[14] L. Fei-Fei, J. Deng, and K. Li. 2010. ImageNet: Constructing a large-scale image database. Journal of Vision 9, 8 (2010), 1037–1037. DOI:https://doi.org/10.1167/9.8.1037

[15] Jian Fu, Junwei Sun, and Kaiyuan Wang. 2017. SPARK-A Big Data Processing Platform for Machine Learning. Proceedings - 2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration, ICIICII 2016 (2017), 48–51. DOI:https://doi.org/10.1109/ICIICII.2016.0023

[16] Antonio Gulli. AG's corpus of news articles. AG's corpus of news articles. Retrieved May 20, 2021 from http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[17] Ilkyu Ha, Bonghyun Back, and Byoungchul Ahn. 2015. MapReduce functions to analyze sentiment information from social big data. International Journal of Distributed Sensor Networks 2015, (2015). DOI:https://doi.org/10.1155/2015/417502

[18] Mofiz Mojib Haider, Md Arman Hossin, Hasibur Rashid Mahi, and Hossain Arif. 2020. Automatic Text Summarization Using Gensim Word2Vec and K-Means Clustering Algorithm. In 2020 IEEE Region 10 Symposium (TENSYMP), IEEE, 283–286. DOI:https://doi.org/10.1109/TENSYMP50017.2020.9230670

[19] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (2018), 328–339. DOI:https://doi.org/10.18653/v1/P18-103

[20] Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. Proceedings of 55th Annual Meeting of the Association for Computational Linguistics 1, (2017), 562–570. DOI:https://doi.org/10.18653/v1/P17-1052

[21] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference 2, (2017), 427–431. DOI:https://doi.org/10.18653/v1/e17-2068

[22] Veysel Kocaman and David Talby. 2021. Spark NLP: Natural Language Understanding at Scale. Software Impacts 8, January (2021), 100058. DOI:https://doi.org/10.1016/j.simpa.2021.100058

[23] Filippos Kokkinos and Alexandros Potamianos. 2017. Structural attention neural networks for improved sentiment analysis. In Proceedings of 5th Conference of the European Chapter of the Association for Computational Linguistics, 586–591. Retrieved from https://aclanthology.org/E17-2093

[24] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. (July 2019). Retrieved from http://arxiv.org/abs/1907.11692

[25] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. (2014), 55–60. DOI:https://doi.org/10.3115/v1/p14-5010

[26] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In Proceedings of the 31st International Conference on Neural Information Processing Systems, 6295–6306. Retrieved from https://dl.acm.org/doi/10.5555/3295222.3295377

[27] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. 2016. MLlib: Machine learning in Apache Spark. Journal of Machine Learning Research 17, (2016), 1–7. Retrieved from http://jmlr.org/papers/v17/15-237.html

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. Proceedings of 1st International Conference on Learning Representations, ICLR 2013 (January 2013). Retrieved from http://arxiv.org/abs/1301.3781

[29] MLib. 2018. Apache Spark MLib. Retrieved July 11, 2021 from https://spark.apache.org/mllib

[30] Spark NLP. 2021. John Snow Labs - Spark NLP. Retrieved July 11, 2021 from https://nlp.johnsnowlabs.com

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 1532–1543. DOI:https://doi.org/10.3115/v1/D14-1162

[32] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 1532–1543. DOI:https://doi.org/10.3115/v1/D14-1162

[33] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference 1, (2018), 2227–2237. DOI:https://doi.org/10.18653/v1/n18-1202

[34] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. OpenAI (2018). Retrieved from https://openai.com/blog/language-unsupervised

[35] Piotr Semberecki and Henryk Maciejewski. 2016. Artificial Intelligence and Soft Computing. In International Conference on Artificial Intelligence and Soft Computing, 621–630. DOI:https://doi.org/10.1007/978-3-319-39378-0_53

[36] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. (September 2017). Retrieved from http://arxiv.org/abs/1709.04696

[37] Grigori Sidorov, Anubhav Gupta, Martin Tozer, Dolors Catala, Angels Catena, and Sandrine Fuentes. 2013. Rule-based system for automatic grammar correction using syntactic n-grams for english Language Learning (L2). In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task, 96–101. Retrieved from https://aclanthology.org/W13-3613

[38] Marina Sokolova. 2018. Big Text advantages and challenges: classification perspective. International Journal of Data Science and Analytics 5, 1 (2018), 1–10. DOI:https://doi.org/10.1007/s41060-017-0087-5

[39] SpaCy. 2016. Spacy: Industrial-strength natural language processing in python. Retrieved from https://spacy.io

[40] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. (August 2019). Retrieved from http://arxiv.org/abs/1908.08962

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., 6000–6010. Retrieved from https://dl.acm.org/doi/10.5555/3295222.3295349

[42] Jiaying Wang, Yaxin Li, Jing Shan, Jinling Bao, Chuanyu Zong, and Liang Zhao. 2019. Large-Scale Text Classification Using Scope-Based Convolutional Neural Network: A Deep Learning Approach. IEEE Access 7, (2019), 171548–171558. DOI:https://doi.org/10.1109/ACCESS.2019.2955924

[43] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke. 2017. The Microsoft 2017 Conversational Speech Recognition System. (August 2017). Retrieved from http://arxiv.org/abs/1708.06073

[44] Yan Xu, Kai Hong, Junichi Tsujii, and Eric I.Chao Chang. 2012. Feature engineering combined with machine learning and rule-based methods for structured information extraction from narrative clinical discharge summaries. Journal of the American Medical Informatics Association 19, 5 (2012), 824–832. DOI:https://doi.org/10.1136/amiajnl-2011-000776

[45] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. XLNet: Generalized Autoregressive Pretraining for Language Understanding. (January 2020). Retrieved from http://arxiv.org/abs/1906.08237

[46] Zichao Yang, Diyi Yang, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1480–1489. DOI:https://doi.org/10.18653/v1/N16-1174

[47] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster computing with working sets. In 2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010 (HotCloud'10), 10. DOI:https://doi.org/10.5555/1863103.1863113

[48] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. (2016). Retrieved from http://arxiv.org/abs/1611.06639