Efficient Large Scale NLP Feature Engineering with Apache Spark

Armin Esmaeilzadeh
University of Nevada Las Vegas
Las Vegas, Nevada, USA
esmaeilz@unlv.nevada.edu

Maryam Heidari
George Mason University
Virginia, USA
mheidari@gmu.edu

Reyhaneh Abdolazimi
Syracuse University
rabdolaz@syr.edu

Parisa Hajibabaee
University of Massachusetts at Lowell
Lowell, USA
parisa_hajibabaee@student.uml.edu

Masoud Malekzadeh
University of Massachusetts at Lowell
Lowell, USA
masoud_malekzadeh@student.uml.edu

***Abstract*—Feature engineering is a computationally time-consuming process in the end-to-end machine learning pipeline. Large amounts of text data are being generated on many heterogeneous sources and platforms on the internet. The compute resources needed to extract valuable features from these big datasets are increasing significantly. In this research, we evaluate the runtime of the RDD and the SparkSQL APIs of the Apache Spark framework to extract text features from the corpus of english Wikipedia. As a result, we demonstrate the significant runtime performance of the SparkSQL compared to RDD API.**

***Index Terms*—natural language processing, NLP, machine learning, distributed systems, big data, Apache Spark**

I. Introduction

Online Text data has been growing significantly over the last two decades on the internet. [1] Natural language contents, both written and spoken, that are created on public social media platforms, news agencies or knowledge data stores such as Wikipedia are rich data sources for Natural Language Processing (NLP) tasks. [2], [3] [4]

The applications of NLP models can range from detecting fake bots on Twitter [5], [6] to advanced applications of AI in different scientific fields [7]–[13] [14]–[18] and to building question-answering bots for customer service software [19], [20]. There are also more critical applications such detecting misinformation on online content based on nlp models [21]–[26], classifying drug interactions in medical documents [27] and analyzing crime text reports. [28] All these applications require a syntactic and semantic understanding of natural language. [29]

In order to model natural language properties, a wide range of statistical and machine learning techniques have been utilized. [30]–[35] Historically, manual feature extraction from text documents by domain experts or capturing statistical properties of words in documents have been the main approaches. [36] But over the last decade, deep learning-based NLP models have substantially improved the accuracy and precision of many common NLP tasks compared to the previous generation of models. [37]

One of the main challenges of developing NLP models is prepossessing text data to construct clean datasets and extract effective features (feature engineering) to feed models. [38] These models could be served as online applications or batch jobs. [39] As the scale and volume of public datasets continue to grow, it is required to process text data more efficiently. [40]

Feature engineering in machine learning is the task of transforming the input feature vector into a new vector such that it increases the predictive power of the model. [41] A wide range of engineered features, such as log or polynomial transformations, have been proposed in machine

978-1-6654-8303-2/22/$31.00 ©2022 IEEE

2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) | 978-1-6654-8303-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCWC54503.2022.9720765

learning studies. [42] Most of these techniques are domain specific such as the use of graphlets in network analysis as described in [43] or detecting malware in mobile devices using a variety of machine learning techniques. [44] In NLP research, the word to vector representation features have been the most successful. But there are many cases in which engineered features such as Part of Speech Tagging, or Entity Extraction can be helpful when datasets are small and there is limited computational power available for training. [45]

Among big data processing platforms that have been developed over the last decade, Apache Spark has become one of the most capable and efficient data processing engines. [40] Similar to the MapReduce programming paradigm introduced by [46], Apache Spark supports map and reduce operations as distributed in-memory data transformations and can significantly reduce the runtime cost. [47] The Spark engine provides two data transformation APIs, namely RDD and SparkSQL, which we will explore their performance on text feature extraction in this experiment.

In Section II, we define feature engineering techniques in the literature and the features we will use in our experiment. In Section III, we review the big data platforms that we will utilize to conduct our experiment. In Section IV, we describe the details of our experiment which includes the configuration of the big data cluster, the dataset, and the feature extraction techniques. In Section IV, we show the job runtime results of the experiments. Finally, in Section VI, we review our experiment to outline an approach to extract text features from large datasets for NLP models.

II. Background

The early feature engineering techniques for NLP tasks is known as Bag of Words. In bag of words representations, the semantic and grammatical properties of language are ignored and only singleton words in a document are considered during the feature selection process. For example, the frequency of each word in a document could be used as a feature for the text document. These techniques are also known as Vector Space models. [36]

Another attempt to make the words in a document more statistically independent is using stemming algorithms. The stemming process will reduce a given word to its root or stem form. For instance, the words “driving”, and “driver” will be reduced to the root word “drive”. Therefore, the number of stem words are less than the number of all words in the document. [36]

Even for basic features, such as the frequency of each word in text documents, the required computation can be significant given terabytes of text data to process.

In this research, we consider the Part of Speech Tagging, which we will define in Section VI, and stem word frequency as text features to extract from a large text dataset in our experiment.

III. Big Data Platforms

Many technological trends have been developing over the last thirty years of the adaptation of the internet. Applications such as messaging, social media, news agencies, knowledge repositories, etc. are generating petabytes of data on a daily basis. [1]

The traditional data storage and processing technologies could not efficiently handle this vast amount of data. [40] At the same time, companies are aware of the value that the big data processing and analysis can bring to them. [48]

In the following sections, we review some of the most capable distributed data storage and processing technologies that can handle large scale datasets and that we will utilize in our experiment.

A. MapReduce

MapReduce was proposed as a programming model for processing large datasets in batch jobs. [46] The underlying implementation uses two functions, map and $reduce$, to process data. The map function, takes in a key-value pair as input and applies the mapping logic to each pair to generate a new set of key-value pairs. The $reduce$ function, groups all associated values for a given key and apply the reduce logic to generate the final aggregated result set. These two operations are shown below: [46]

$$map(k1, v1) \rightarrow list(k2, v2) \tag{1}$$

$$reduce(k2, list(v2)) \rightarrow list(v2) \quad (2)$$

The MapReduce framework can be set up and run on commodity hardware which is a major advantage compared to single machine databases that required expensive hardware. Moreover, the framework can scale up to use thousands of machines and handles failovers and data partitioning automatically. [46] The MapReduce paradigm has significant performance improvements compared to previous single machine approaches. [49]

B. Hadoop Distributed File System (HDFS)

HDFS is a distributed data storage file system that is heavily utilized by Hadoop and other big data platforms. [50] It can run on commodity hardware and scale to hundreds of nodes with support for automatic failovers. Its file system design allows us to store large-scale structured and unstructured datasets. HDFS is ideal for running large batch processing jobs that are not sensitive to delays in file retrieval. [50]

In our experiment, we utilize HDFS cluster to store datasets.

C. Yet Another Resource Negotiator (YARN)

Apache YARN is the second generation of a resource management system in the Hadoop environment. This new architecture decouples the MapReduce programming model from cluster and resource management tasks such as scheduling jobs. [51]

The YARN platform uses a distributed master-worker architecture and several components. The first component is a Resource Manager (RM) that is dedicated per cluster to track hardware resource usage for jobs, check the liveness of worker nodes, and enforce resource allocation specifications. [51]

The second component is an Application Master (AM) which coordinates the jobs execution sequence in the cluster given the resources. [51]

The third component is Node Manager (NM) which is the worker daemon in the YARN architecture and monitors the execution of job containers, manages job container dependencies, and provides required services to those containers. [51]

In our experiment, we utilize Apache YARN as the resource manager to run our batch jobs on top of Apache HDFS as the storage environment.

D. Apache Spark

Apache Spark is a distributed data processing framework developed by [47]. The ability of the Spark engine to retain intermediate datasets in memory used in different data transformations can greatly improve the performance of batch processing jobs compared to the MapReduce framework. [47]

Apache Spark defines a core internal data structure called Resilient Distributed Dataset (RDD). [52] The RDD is a new programming interface that allows developers to pull large datasets into memory and apply data transformations to them. The RDD data abstraction follows coarse-grain updates to datasets instead of fine-grained data transformations in previous database models. [52]

In this research, we utilize Apache Spark as the main compute engine to extract text features from the corpus.

IV. Experiment

In this experiment, we use the Apache HDFS as the storage layer for our big data cluster to store and retrieve datasets. We utilize Apache Yarn as the resource manager to provide the hardware resources required to run our batch jobs. And finally, Apache Spark is the main data processing engine we use to extract two features from datasets.

We perform the two feature extraction techniques using the RDD and SparkSQL APIs of Apache Spark with different parameters for utilizing hardware resources. The main variables are CPU and RAM

In the following section, we provide details on each step of the experiment.

A. Big Data Cluster

The big data cluster we have set up has ten servers. Nine of these servers are running the HDFS worker nodes to store data files and one master node is running the master Name Node to keep track of data blocks across the cluster. Apache Yarn cluster has a master Resource Manager node running on the same master node as the HDFS Name Node.

Other nodes are running the Node Manager daemon servers to define the available resources in each node for the Resource Manager.

The Apache Spark package is installed in each one of the servers. After submitting a job to the cluster, a master application container will be created to coordinate the distributed execution of the program.

The hardware specification of the big data cluster is shown in Figure 1.

Node	Total vCore	Total Memory (GB)	Total Data Storage (TB)	Network Speed (Gb)
Master	48	131.7	18.1	10
Worker	48	131.7	18.1	10
Worker	48	131.7	18.1	10
Worker	48	131.7	18.1	10
Worker	48	131.7	18.1	10
Worker	48	263.8	43.2	10
Worker	48	263.8	43.2	10
Worker	48	263.8	43.2	10
Worker	48	263.8	43.2	10
Worker	48	263.8	43.2	10

Fig. 1: Big Data Cluster Hardware Specification.

B. Dataset

The English Wikipedia is a rich data source for text analysis and NLP modeling tasks. The archived files for the Wikipedia articles are available at [53]. For our experiment, we use the XML archive file that was created on August 28, 2021. This version has a size of 88 Gigabytes and includes all the contents of the English Wikipedia up to the data the archive was created.

C. Part of Speech Tags

Part of speech (POS) tagging is the process of classifying the words in a document with a label describing their grammatical roles, such as a noun or verb, based on the definition and context of words. The POS of words is a helpful feature in a variety NLP tasks such as text classification or sentiment analysis. [45]

As an example, consider the sentence ”I see a bird”. The following table shows some of the suggested tags for each word in the sentence: [54]

Word	Part of Speech Tag
I	PPSS
see	VB
a	AT
bird	NN

In the given example, PPSS = pronoun; NP = proper noun; VB =verb; UH = interjection; IN = preposition; AT =article; NN = noun. [54]

In this experiment, we use the POS tagging functionality of the Natural Language Toolkit (NLTK) library in Python. We iterate through the sentences in each article of the Wikipedia files using the Spark engine and assign a label to words in those sentences. These tags can later be used as features in classification problems.

We perform the POS tagging process using both the RDD and SparkSQL APIs of the Spark engine and show the run-time result in Section V.

D. Stem Word Frequency

In Information Retrieval systems, stemming is a process to reduce a given word to its root or stem form. In the example shown below, all the words in a document have a similar meaning, and therefore, the suffixes such as -ING, -ED, -ION, -IONS could be removed and consider CONNECT as the stem form. [55]

CONNECTED
CONNECTING
CONNECTION
CONNECTIONS

In order to follow the most common practices in feature engineering for stem word frequencies, we remove the stop words from all the articles and perform stemming algorithm to reduce each word to its root form. [45] The stemming transformation will decrease the number of words we have to process and gives us a more statistically meaningful representation. One of the commonly used algorithms for stemming words was proposed by [55].

In this step, we use the NLTK library in Python to find matching stop words and remove them. The same library provides the porter stemming algorithm to extract the stem form of each word in an article.

We perform the aggregation operation to find the number of times a stem word has appeared across all the articles using both the RDD and SparkSQL APIs of Spark.

V. Run Time Evaluation

In the first experiment, we extracted the POS tags and stem word frequency of every article in the Wikipedia archived file. Figure 2 shows the runtime for the RDD and SparkSQL APIs to perform the feature engineering tasks.

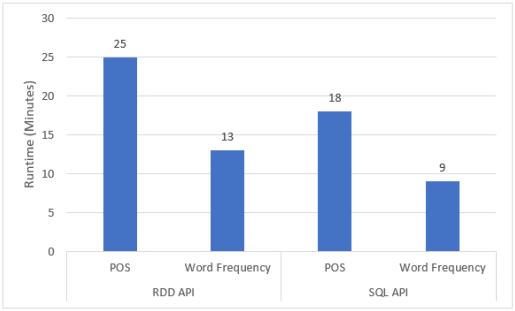

Fig. 2: Job Runtime.

As shown in the Figure 2, there is 28% decrease in the runtime between RDD and SparkSQL APIs when extracting POS. Additionally, there is a decrease of 30% using the SparkSQL API when extracting stem word frequency compared to RDD API.

The significant difference between the performance of the two APIs can be mainly attributed to the optimizations that the SparkSQL engine can provide. The RDD API executes the operations as defined by the user and does not perform optimizations on the job execution. On the other hand, the SparkSQL API provides the data access in a declarative paradigm such that the user defines what data transformation they need to perform, and the SparkSQL engine decides how to perform the computation. [56]

Once the SQL query is executed, the Spark engine builds an initial execution plan for the SQL query and its abstract syntax tree. Then, a number of query optimizations proceed to construct an efficient physical plan that will be executed. The Catalyst Optimizer in the SparkSQL engine is responsible to convert SparkSQL API requests to map and reduce operations on RDDs. [57]

In the second experiment, we used a variable number of Spark executors to run the batch jobs. Figure 2 shows that by using fifty Spark executors, the run-time decreases by 67% and 80% when extracting POS tags and stem word frequencies respectively compared to using three executors. Moreover, the SparkSQL API still outperforms the RDD API when we increase the number of nodes.

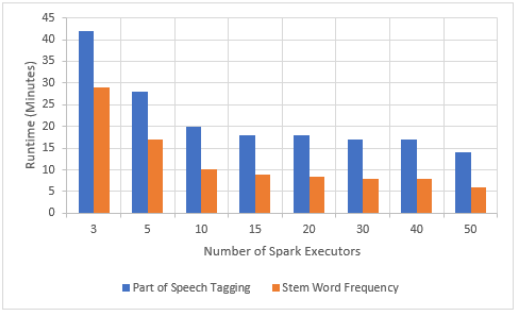

Fig. 3: Job Runtime - Variable Number of Nodes.

The large reduction in runtime when using fifty executors instead of three is attributed to higher parallelization of the jobs since the POS and stem word frequency of independent articles can be distributed and only a final layer of aggregation is needed for stem word frequencies.

VI. Conclusion

The radical increase in text data generation on the internet over the last two decades has brought a new set of data processing requirements. More academic institutions and companies have dedicated their resources to developing and maintaining big data platforms to tackle vast amounts of data. Additionally, the increase in natural language contents has given rise to many new NLP modeling techniques which can extract valuable features from raw text data. These feature engineering techniques, however, require a new set of tools and big data processing approaches to be efficient.

In this experiment, we set up and developed a big data processing environment and performed an empirical analysis to compare the performance of the two data transformation APIs on the Apache Spark platform. In Section V, we demonstrated that using the SparkSQL API on the Apache Spark engine can bring significant improvements to the time complexity of long-running batch jobs.

Following this approach on large datasets can also bring cost reduction while utilizing shared hardware and data processing environments.

ACKNOWLEDGMENT

This research paper is based on works that are supported by the National Science Foundation Grant No. 1625677.

REFERENCES

[1] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big data technologies: A survey," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018.

[2] M. Heidari and J. H. Jones, "Using bert to extract topic-independent sentiment features for social media bot detection," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2020, pp. 0542–0547.

[3] M. Heidari, J. H. Jones, and O. Uzuner, "Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter," in *2020 International Conference on Data Mining Workshops (ICDMW)*, 2020, pp. 480–487.

[4] M. Heidari and S. Rafatirad, "Semantic convolutional neural network model for safe business investment by using bert," in *2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2020, pp. 1–6.

[5] M. Heidari, "Nlp approach for social media bot detection(fake identity detection) to increase security and trust in online platforms," 2022.

[6] M. Heidari and J. H. J. Jones, "Fraud detection to increase customer trust in online shopping experience," 2022.

[7] P. Hajibabaee, F. Pourkamali-Anaraki, and M. Hariri-Ardebili, "Kernel matrix approximation on class-imbalanced data with an application to scientific simulation," *IEEE Access*, pp. 83 579–83 591, 2021.

[8] S. Zad, M. Heidari, J. H. J. Jones, and O. Uzuner, "Emotion detection of textual data: An interdisciplinary survey," in *2021 IEEE World AI IoT Congress (AIIoT)*, 2021, pp. 0255–0261.

[9] P. Hajibabaee, F. Pourkamali-Anaraki, and M. Hariri-Ardebili, "An empirical evaluation of the t-sne algorithm for data visualization in structural engineering," *arXiv preprint arXiv:2109.08795*, 2021.

[10] R. Abdolazimi, H. Naderi, and M. Sagharichian, "Connected components of big graphs in fixed mapreduce rounds," *Cluster Computing*, vol. 20, no. 3, pp. 2563–2574, 2017.

[11] R. Abdolazimi and R. Zafarani, "Noise-enhanced unsupervised link prediction," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2021, pp. 472–487.

[12] M. Heidari and J. H. J. Jones, "Misinformation detection model to prevent spread of the covid-19 virus during the pandemic," 2022.

[13] R. Abdolazimi, S. Jin, and R. Zafarani, "Noise-enhanced community detection," in *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, 2020, pp. 271–280.

[14] M. Heidari and S. Rafatirad, "Using transfer learning approach to implement convolutional neural network model to recommend airline tickets by using online reviews," in *2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization (SMA*, 2020, pp. 1–6.

[15] P. Hajibabaee, "Transit agencies performance assessment and implications," 2020.

[16] M. Heidari and J. H. J. Jones, "Offensive behaviour detection on social media platforms by using natural language processing models," 2022.

[17] P. Hajibabaee, F. Pourkamali-Anaraki, and M. Hariri-Ardebili, "An empirical evaluation of the t-sne algorithm for data visualization in structural engineering," in *2021 IEEE International Conference on Machine Learning and Applications*. IEEE, 2021.

[18] M. Torkjazi and H. Fazlollahtabar, "Applying a utility based fuzzy probabilistic α-cut method to optimize a constrained multi objective model," *Creative Mathematics and Informatics*, vol. 24, no. 2, pp. 221–230, 2015.

[19] M. Heidari and S. Rafatirad, "Bidirectional transformer based on online text-based information to implement convolutional neural network model for secure business investment," in *2020 IEEE International Symposium on Technology and Society (ISTAS)*, 2020, pp. 322–329.

[20] M. Heidari, J. H. J. Jones, and O. Uzuner, "An empirical study of machine learning algorithms for social media bot detection," in *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2021, pp. 1–5.

[21] M. Heidari, S. Zad, P. Hajibabaee, M. Malekzadeh, S. HekmatiAthar, O. Uzuner, and J. H. Jones, "Bert model for fake news detection based on social bot activities in the covid-19 pandemic," in *2021 IEEE 12th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2021, pp. 0103–0109.

[22] S. Zad, M. Heidari, J. H. Jones, and O. Uzuner, "A survey on concept-level sentiment analysis techniques of textual data," in *2021 IEEE World AI IoT Congress (AIIoT)*, 2021, pp. 0285–0291.

[23] H. Nazari, B. Abedin, P. Hajibabaee, and E. Fallah, "Identifying and ranking critical factors that determine meritocracy using ahp technique in automotive industry in iran," in *International Conference on Technology and Business Management March*, vol. 24, 2014, p. 26.

[24] M. Heidari, S. Zad, B. Berlin, and S. Rafatirad, "Ontology creation model based on attention mechanism for a specific business domain," in *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2021, pp. 1–5.

[25] M. Malekzadeh, T. Song, and J. Dutta, "Pet image denoising using unsupervised domain translation," in *2021 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC)*. IEEE, 2021.

[26] P. Hajibabaee, M. Malekzadeh, M. Heidari, S. Zad, O. Uzuner, and J. H. Jones, "An empirical study of the graphsage and word2vec algorithms for graph multiclass classification," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021, pp. 0515–0522.

[27] M. E. Z. N. Kambar, P. Nahed, J. R. F. Cacho, G. Lee, J. Cummings, and K. Taghva, "Clinical text classification of alzheimer's drugs' mechanism of action," in *Proceedings of Sixth International Congress on Information and Communication Technology*. Springer, 2022, pp. 513–521.

[28] S. Khorshidi, G. Mohler, and J. G. Carter, "Assessing gan-based approaches for generative modeling of crime

text reports," in *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2020, pp. 1–6.

[29] A. Esmaeilzadeh and K. Taghva, "Text classification using neural network language model (nnlm) and bert: An empirical comparison," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2021, pp. 175–189.

[30] S. Zad, M. Heidari, P. Hajibabaee, and M. Malekzadeh, "A survey of deep learning methods on semantic similarity and sentence modeling," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021, pp. 0466–0472.

[31] M. Malekzadeh, P. Hajibabaee, M. Heidari, and B. Berlin, "Review of deep learning methods for automated sleep staging," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022.

[32] M. Malekzadeh, P. Hajibabaee, M. Heidari, S. Zad, O. Uzuner, and J. H. Jones, "Review of graph neural network in text classification," in *2021 IEEE 12th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2021, pp. 0084–0091.

[33] J. Liu, M. Malekzadeh, N. Mirian, T. Song, C. Liu, and J. Dutta, "Artificial intelligence-based image enhancement in pet imaging: Noise reduction and resolution enhancement," *PET clinics*, vol. 16, no. 4, pp. 553–576, 2021.

[34] M. Heidari, S. Zad, and S. Rafatirad, "Ensemble of supervised and unsupervised learning models to predict a profitable business decision," in *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2021, pp. 1–6.

[35] D. Bamgboje, I. Christoulakis, I. Smanis, G. Chavan, R. Shah, M. Malekzadeh, I. Violaris, N. Giannakeas, M. Tsipouras, K. Kalafatakis *et al.*, "Continuous non-invasive glucose monitoring via contact lenses: Current approaches and future perspectives," *Biosensors*, vol. 11, no. 6, p. 189, 2021.

[36] S. Scott and S. Matwin, "Feature engineering for text classification," in *ICML*, vol. 99. Citeseer, 1999, pp. 379–388.

[37] S. Khorshidi, G. Mohler, and J. G. Carter, "Assessing gan-based approaches for generative modeling of crime text reports," in *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2020, pp. 1–6.

[38] P. Hajibabaee, M. Malekzadeh, M. Ahmadi, M. Heidari, A. Esmaeilzadeh, R. Abdolazimi, , and J. H. j. Jones, "Offensive language detection on social media based on text classification," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022.

[39] A. Esmaeilzadeh, "A test driven approach to develop web-based machine learning applications." UNLV Theses, Dissertations, Professional Papers, and Capstones, 2017. [Online]. Available: http://dx.doi.org/10.34917/11889688

[40] N. A. Ghani, S. Hamid, I. A. T. Hashem, and E. Ahmed, "Social media big data analytics: A survey," *Computers in Human Behavior*, vol. 101, pp. 417–428, 2019.

[41] C. Geigle, Q. Mei, and C. Zhai, "Feature engineering for text data," in *Feature Engineering for Machine Learning and Data Analytics*. CRC Press, 2018, pp. 15–54.

[42] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," in *SoutheastCon 2016*. IEEE, 2016, pp. 1–6.

[43] S. Khorshidi, M. Al Hasan, G. Mohler, and M. B. Short, "The role of graphlets in viral processes on networks," *Journal of Nonlinear Science*, vol. 30, no. 5, pp. 2309–2324, 2020.

[44] M. Esmail Zadeh Nojoo Kambar, A. Esmaeilzadeh, Y. Kim, and K. Taghva, "A survey on mobile malware detection methods using machine learning," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, virtual, USA, Jan. 2022.

[45] T. Wambsganss, C. Engel, and H. Fromm, "Improving explainability and accuracy through feature engineering: A taxonomy of features in nlp-based machine learning," in *Forty-Second International Conference on Information Systems*, 2021.

[46] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[47] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin *et al.*, "Apache spark: a unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[48] D. Robinson, Z. Zhang, and J. Tepper, "Hate speech detection on twitter: Feature engineering vs feature selection," in *European Semantic Web Conference*. Springer, 2018, pp. 46–49.

[49] R. Abdolazimi, M. Heidari, A. Esmaeilzadeh, and H. Naderi, "Mapreduce preprocess of big graphs for rapid connected components detection," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022.

[50] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*. Ieee, 2010, pp. 1–10.

[51] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013, pp. 1–16.

[52] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 15–28.

[53] [Online]. Available: https://dumps.wikimedia.org/dewiktionary/latest/

[54] K. W. Church, "A stochastic parts program and noun phrase parser for unrestricted text," in *International Conference on Acoustics, Speech, and Signal Processing,*. IEEE, 1989, pp. 695–698.

[55] M. F. Porter, "An algorithm for suffix stripping," *Program*, 1980.

[56] [Online]. Available: https://spark.apache.org/docs/latest/sql-ref.html

[57] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi *et al.*, "Spark sql: Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, 2015, pp. 1383–1394.