Textual Analysis of the EKU Student Handbook 2022 using Word Count, Term Frequency - Inverse Document Frequency and Latent Dirichlet Allocation

Hamza Khattak
Dept. of Computer Science
Eastern Kenutcky University
Richmond, Kentucky
muhammad_khattak@mymail.eku.edu

Enock Kipchumba
Dept. of Computer Science
Eastern Kenutcky University
Richmond, Kentucky
enock_kipchumba@mymail.eku.edu

Abstract – The rise of Big Data has sparked transformation across various industries to gather, process and leverage the information so as to make informed decisions based on real-time insights and analytics.
The aim of our research project is to demonstrate transformative tools that could help real world industries to extract meaningful information and insights through data analysis. This study illustrates how MapReduce can be applied to text processing such as Word Count, TF-IDF and LDA and demonstrates its adaptability in extraction of meaningful information from the textual data in the context of the Hadoop ecosystem.

I. Introduction

In recent years, there has been an increase in digital information, leading to an exponential growth in volume and variety of data. The data generated through social, online interactions, financial transactions, commerce, and such are stored and analyzed by experts to understand it. This endeavor has prompted the necessity for advanced analytical tools to procure meaningful insights. Which is what our project is attempting to achieve here. The primary objective of our project is to take a journey through the textual nature of our data, glean pertinent information and explore the three informative techniques: Word Count, TF-IDF and LDA. These techniques offer a glimpse into the confined landscape of data to extract insights, visible and hidden meaning from the text.

We aim to reach out goals by employing Hadoop, an ecosystem which is ideally designed to process and store large amounts of data across distributed clusters of computers. Hadoop's characteristics such scalability, fault tolerance, flexibility, parallel-processing, and cost-effectiveness make it an unparalleled solution to the challenges brought by explosive growth of information [1]. The technique applied here is MapReduce, a mechanism which plays a major role in handling extensive datasets, making it convenient for big data analysis.

MapReduce through its leverage of distribution of Hadoop clusters it is able to process data in parallel across multiple nodes which makes it great for handling large datasets. This project will use the Eastern Kentucky University student handbook from 2020, and apply text processing such as word count, TF-IDF and LDA to extract profound insights.

II. System Specifications

To perform the textual analysis, we had to employ the Hadoop Distributed framework and take advantage of MapReduce. While our Hadoop system is using a single node cluster on a Manjaro Linux operating system hosted in the Oracle VirtualBox, we can extrapolate the result to real life scenarios outside our academic environment and consider its widespread application of this idea.

We used Python programming language to develop preprocessing, mapping, and reducer steps for each technique. We installed some of the python built-in libraries for natural language processing such as SpaCy and scikit-learn, which offer feature extraction using Countvectorizer and LatentDirichletAllocation for topic modeling. The project was created on an 11th generation Intel Core

i5 with 8 processors and 16 GB RAM, out of which 4 processors and 8 GB RAM were dedicated to the virtual box.

III. Dataset

The dataset is the Eastern Kentucky University Student Handbook 2020 for Word Count, TF-IDF and LDA and the Eastern Kentucky University Student Handbook for 2022 as a secondary source of data for TF-IDF. The handbook provides a structured description of guiding principles and policies, academic information, student assistance, support and student conduct and community standards for its populace [2]. The handbook comes in the form of pdf format, which was converted into a text file using pdfminer's pdf2txt.py program [3].

IV. Data Processing

Data preprocessing is essential for our project since our analysis requires structured data in a specific format for each step of MapReduce framework. Our original data was noisy, messy, and unstructured, rendering us unable to process it for the models. In terms of the preprocessing steps, after converting the file type from pdf to txt, we cleaned the text data by removing irrelevant characters, whitespace, special characters, and symbols such as '#@',.-&^(' which are irrelevant to the analysis. We also removed URLs, QR codes, numbers, and we converted all text data to lowercase to ensure the data is uniform and not case sensitive. We also removed most frequently used words or commonly known as 'stopwords', words such as 'a', 'the', 'and', 'of' and so on, which have no significant meaning in the overall picture of the data and are only used for human communication.

We also performed lemmatization for LDA only. Lemmatization is a common natural language processing technique which is used to reduce words to their root form or lemmas [4]. In our case, words such as 'accompany, accompanied, accompanies' are lemmatized to 'accompany'. For TF-IDF we applied vectorization, which is the process of transforming each word into a vector where the document is converted into a matrix [5].

V. Word Count

Word Count is a process numerating the frequency of each word in the text or document. It involves the calculation of the occurrence of each key-value pair present in the text file. After data preprocessing, we perform a mapper function to the data text file. There is one mapper and one reducer for our word count program. After preprocessing, the mapper takes in the input text file and outputs the set of key-value pairs of the words. These key-value pairs are each associated word count of 1 hence counting the number of words. These <key, value> pairs emitted by the mapper represent intermediate output.

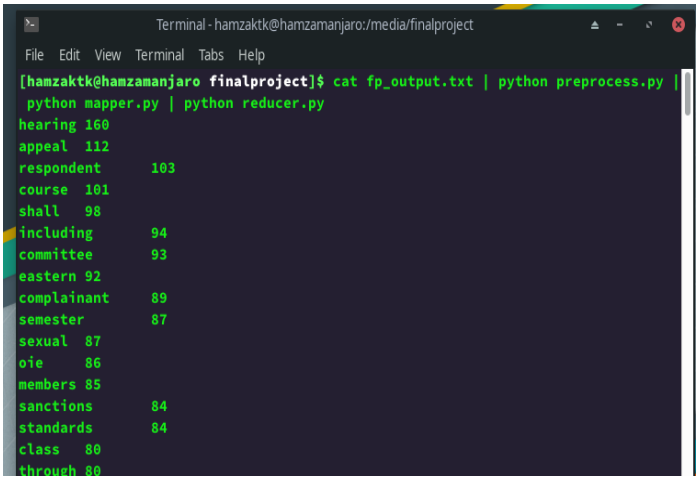

Figure 1: Output of running word count on EKU student Handbook on the local system

In the Reduce phase, the reducer takes in the key-value pairs and counts values based on the keys from the mapper. The reducer aggregates the word counts by summing up the count of words with the same key. The counted words are sorted in descending order to ensure that the word count reflects the convenience of the analysis, in which we are trying to find the words with the highest frequency count from the handbook, output shown in figure 1 and the output shown in figure 2 is streaming job of word count on Hadoop.

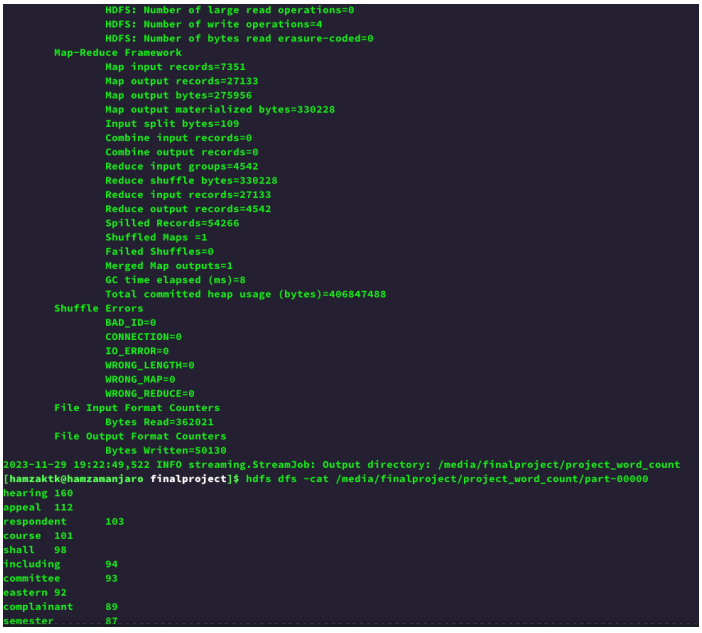

Figure 2: Output of running word count on EKU student Handbook on HDFS (Hadoop)

VI. Term Frequency – Inverse Document Frequency

Term Frequency (TF) is the frequency of a term in a text or document. It involves the count of specific words in relation to the total number of words in a text or document [15]. It is calculated by the occurrences of a term in a text which is word count then dividing the word count with the numbers of words in the text file or document. The term frequency of the *ith* term in the *jth* document, denoted by TF(ti, dj) is given by

$$TF(ti,dj) = tcij/ max(dj)tc$$

where the tcij is the term count and max(dj) is the maximum term count in the jth document [6]. The higher the TF of a word, the higher the weight of the document.

Inverse Document Frequency (IDF) highlights the number of documents containing the term. This evaluates the uniqueness of the word across the number of documents. The IDF for the ith term is given by [7],

$$IDF(ti) = \log D/dfi$$

where D is the number of documents in the corpus and dfi is the number of documents containing the ith term, which is the document frequency [8]. The higher the IDF, the more significant and unique the word across the collection of the documents.

The combination of term frequency and inverse document frequency, called TF-IDF, illustrates the significance and weight of a word across the collection of documents. This helps to assess the importance of words in relation to the collection of documents. The formula for the ith term is given by [9],

$$TF\text{-}IDF(ti,dj) = TF(ti,dj) * IDF(ti)$$

The TF-IDF reducer reads the input form the mapper, which is the word, count, and the value which TF-IDF of each word. The reducer sums up the TF-IDF values and increments the document count. It then averages the TF-IDF values of the word using the sum of values and document count. We made the reducer sorts the TF-IDF scores in descending order instead of using a specified 'sort' function in the command line. The word with high TF-IDF score indicates the word is more significant and relevant to a specific document in relation to the entire collection of documents [10].

Unlike the wordcount MapReduce, the TF-IDF uses a script to run the program since m documents were required for TF-IDF and running the pipeline multiple times was ineffective. Figure 3 shows the output of our TF-IDF output for m = 2 and figure 4 shows our workflow for TF-IDF.

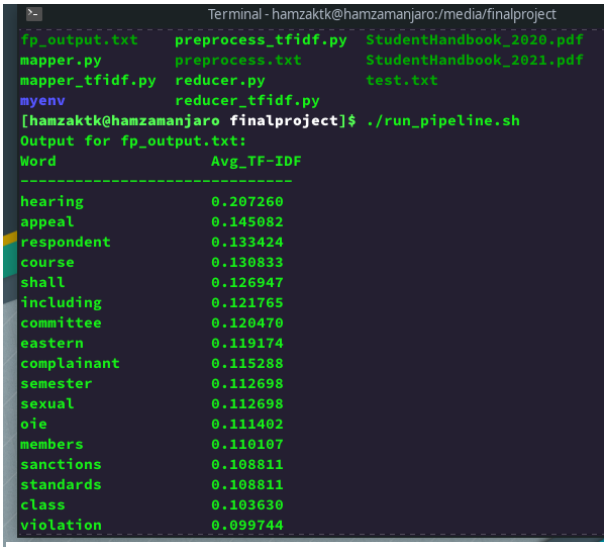

Figure 3: Output of TF-IDF after running our run_pipeline.sh script

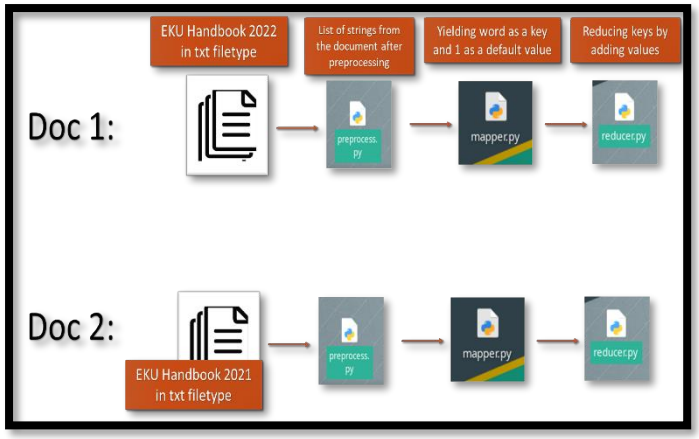

Figure 4: Workflow of our TF-IDF MapReduce

VII. LDA FOR TOPIC MODELING

The Latent Dirichlet Allocation is a generative probabilistic model that identifies the topics presenting the document and maps significant words to the topics [11]. The goal for using LDA in our project is to reveal any hidden topics in our document. LDA assumes that a document is a mixture of topics and every topic is a mixture of terms [12]. With this basic assumption , here is the mathematical formula of the LDA:

$$p(\theta, z, w \mid \alpha, \beta) = p(\theta \mid \alpha) \, N \prod_{n=1} p(z_n \mid \theta) p(w_n \mid z_n, \beta$$

Which in simple terms denotes the joint probability of the topic mixture θ , set of topics z, and set of words in a document w [13]. Here α is a k-dimensional vector and β is a matrix of word probabilities [14].

In order to employ LDA in Python in Manjaro Linux, we activated a separate Python version than the default version in Manjaro. We used SpaCy library to load the 'en_core_web_sm' to lemmatize the words in our document, which was part of our mapper file. In our reducer the Scikit-Learn library is used to import the CountVectorizer and the LatentDirichletAllocation. While the LDA model does not name a model, for our project we are assuming a main topic as the output without outputting multiple topics only for demonstration purposes.

Figure 5: LDA topic modeling output

The reducer outputs <key, value> format in <Topic, Content>, given in figure 5. Here the topic is assumed to be the key and the content would be the values. Although this might not be the most effective use of the MapReduce because LDA may come after the MapReduce technique is applied, our project shows that this is a possibility, and more testing may be needed to validate its efficacy.

VIII. RESULT AND TESTING

For the project, we used a student handbook 2022 dataset that was obtained from the Eastern Kentucky University website. These datasets underwent enormous and comprehensive analysis incorporating different sets of techniques to extract meaningful insights. Initially, we conducted data preprocessing which involved cleaning, transforming, and normalizing the dataset to ensure its relevance for accurate and meaningful results.

Following this, we performed basic word count analysis to clear up on the frequency of each word within the datasets. Word counting offered sensual understanding of textual data through its volume and offered the text make up. The results were quite obvious that the common words such as prepositions are at the top which is why we added stopwords removal to our processing steps and re-performed our wordcount. In figure 6 we search the term 'hearing' in the EKU student handbook, which according to our world count MapReduce output should show up in the handbook for 160 times as shown in figure 1.

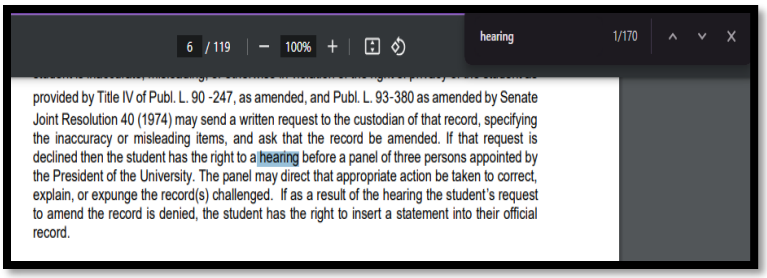

Figure 6: Searching up term 'hearing' in the EKU student handbook of 2020-2021. The search shows up 170 occurrences, including 11 'hearings' leaving us with 159 'hearing' occurrences.

Henceforth, we conducted TF-IDF analysis to allow us to weigh the significance of each word not only to the individual documents but also across a number of documents. This technique enabled us to understand the importance of specific words in a broader context.

Moreover, we conducted Latent Dirichlet Allocation (LDA) to determine the latent structures and underlying topics present in our datasets. The results not only uncovered the patterns and trends in distribution of the key terms but provided broader understanding of exploratory data analysis. In the future, we recommend that we may incorporate this technique with machine learning techniques, thus elevating its potential analysis to beyond term frequencies and latency and also addressing some challenges encountered during the analysis.

IX. References

[1] Team, DataFlair. “10 Features of Hadoop That Made It the Most Popular.” DataFlair, August 25, 2021. https://data-flair.training/blogs/features-of-hadoop-and-design-principles/.

[2] Student handbook - Eastern Kentucky University. Accessed December 2, 2023. https://studentsuccess.eku.edu/sites/studentsuccess.eku.edu/files/files/StudentHandbook_2020.pdf.

[3] “Extract Text from a PDF Using the Commandline.” Extract text from a PDF using the commandline - pdfminer.six __VERSION__ documentation. Accessed December 1, 2023. https://pdfminersix.readthedocs.io/en/latest/tutorial/commandline.html.

[4] Sasikumar, Vivek. “Natural Language Processing - Topic Modelling (Including Latent Dirichlet Allocation-Lda &...” Medium, March 1, 2019. https://medium.com/@vivekvscool/natural-language-processing-topic-modelling-including-latent-dirichlet-allocation-lda-860e5a3d377f.

[5] Sasikumar, Vivek. “Natural Language Processing - Topic Modelling (Including Latent Dirichlet Allocation-Lda &...” Medium, March 1, 2019. https://medium.com/@vivekvscool/natural-language-processing-topic-modelling-including-latent-dirichlet-allocation-lda-860e5a3d377f.

[6] Shehzad, Farhan. “Binned Term Count: An Alternative to Term Frequency for Text Categorization.” Mathematics, December 15, 2022. https://www.academia.edu/92762660/Binned_Term_Count_An_Alternative_to_Term_Frequency_for_Text_Categorization.

[7] Shehzad, Farhan. “Binned Term Count: An Alternative to Term Frequency for Text Categorization.” Mathematics, December 15, 2022. https://www.academia.edu/92762660/Binned_Term_Count_An_Alternative_to_Term_Frequency_for_Text_Categorization.

[8] Shehzad, Farhan. “Binned Term Count: An Alternative to Term Frequency for Text Categorization.” Mathematics, December 15, 2022. https://www.academia.edu/92762660/Binned_Term_Count_An_Alternative_to_Term_Frequency_for_Text_Categorization.

[9] Shehzad, Farhan. “Binned Term Count: An Alternative to Term Frequency for Text Categorization.” Mathematics, December 15, 2022. https://www.academia.edu/92762660/Binned_Term_Count_An_Alternative_to_Term_Frequency_for_Text_Categorization.

[10] Person. “Understanding TF-IDF for Machine Learning.” Capital One, October 6, 2021. https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

[11] Blei, David M, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation - Journal of Machine Learning Research, January 2003. https://jmlr.org/papers/volume3/blei03a/blei03a.pdf.

[12] “Using Topic Modelling to Increase Business Results.” Qualtrics, February 17, 2022. https://www.qualtrics.com/uk/experience-management/research/topic-modelling/?rid=ip&prevsite=en&newsite=uk&geo=GB&geomatch=uk.

[13] [14] Blei, David M, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation - Journal of Machine Learning Research, January 2003. https://jmlr.org/papers/volume3/blei03a/blei03a.pdf.

[15] Muhammad Ibnu Alfarizi, Lailis Syafaah, Merinda Lestandy. Emotional Text Classification Using TF-IDF (Term Frequency-Inverse Document Frequency) And LSTM (Long Short-Term Memory). Jurnal Informatika. 2022;10(2):225-232. doi:10.30595/juita.v10i2.13262