Uppsala University

Data Engineering I

1TD069

March 15, 2024

Investigation on Authors in Reddit Dataset

Authors:
Chenglong Li
Koushik Shrivatsav Balasubramanian
Naresha Krishna Gudalur Koundampalayam Rajendran
Tsion Samuel Tegegn

Contents

1 Background

Reddit is a prominent online community where registered users can post their thoughts and opinions on a wide range of subjects, allowing them to share their perspectives with others from all over the world. The platform is organized into user-created boards called subreddits, which are specialized forums that enable users to engage in discussions about topics that interest them. Currently, there are around 130,000 active subreddits on the site, covering a vast array of topics ranging from science and technology to art and entertainment. One of the unique features of Reddit is the ability for users to upvote or downvote posts and comments, thereby influencing their visibility and ranking on the site. Upvotes signal agreement or appreciation for a post or comment, while downvotes signal disagreement or disapproval. The point system, which is based on these votes, helps users discover the most popular or interesting content on the site. Another noteworthy feature of Reddit is the controversiality and distinguished metrics. The controversiality score is a measure of how polarizing a comment is, calculated based on the balance between upvotes and downvotes. This score is used to highlight comments that elicit a mixed or polarized response from the community. Distinguished comments, on the other hand, are often made by moderators or individuals with special roles within a subreddit. These comments are highlighted to draw attention to important information, announcements, or authoritative statements.

The Webis-TLDR-17 Corpus[1] is a dataset that has been created to facilitate abstractive summarization using deep learning techniques. This dataset contains around 4 million content-summary pairs from the Reddit dataset, providing a valuable resource for researchers and practitioners interested in natural language processing and summarization tasks[2]. The datasets are provided by Bauhaus-Universität Weimar and Leipzig University as part of The Web Technology and Information Systems Network.

Text mining or text analytics is the process of deriving high-quality information from text. This process usually involves structuring input text, such as removing non-essential features and adding new features to obtain structured data. This data is then evaluated to draw conclusions. Text analysis includes information retrieval and dictionary analysis to study the frequency distribution of words, pattern recognition, tagging, and information extraction. Data mining techniques include link and association analysis, visualization, and predictive analysis. The Webis-TLDR-17 Corpus has played a vital role in advancing the field of text information retrieval, providing researchers with a large-scale dataset of text features and metadata that can be used to develop and evaluate text information algorithms.[3] This resource has been used for various analyses, including automatic text summarization techniques and public opinion studies. The availability of this dataset has enabled researchers to make significant strides in the field of natural language processing, thereby enhancing our understanding of text information algorithms and their potential applications.

2 Data Format

The Webis-TLDR-17 Corpus data set is stored in a JSON (JavaScript Object Notation) file. JSON is a lightweight data exchange format that is based on a subset of JavaScript. It is a programming language-independent text format used for storing and representing data. JSON is popular because of its ease of readability and writing, which makes it extremely beneficial to programmers. It is also compact, which enables faster and more efficient data sharing. JSON is versatile and can work well with a variety of programming languages, making it highly suitable for sharing data across different computer systems. Its flexible design allows it to handle a wide range of information types, including lists and complex structures, making it a simple format for organizing different types of data.

Table 1: Schema of a JSON object in the Webis-TLDR-17 Corpus

Field	**Description**
author	The author of the Reddit post
body	The original body text of the post
normalizedBody	The normalized body text of the post
content	The content of the post
content_len	The length of the content
summary	A summary of the post
summary_len	The length of the summary
id	The ID of the post
subreddit	The subreddit where the post was published
subreddit_id	The ID of the subreddit
title	The title of the Reddit post

JSON's hierarchical structure facilitates the representation of complex data relationships, making it ideal for managing information that has interrelated components. This format's simple syntax makes it easy to read and write, which helps programmers with coding and fixing problems. However, it is important to note that JSON's textual representation may result in larger file sizes compared to more compressed formats like CSV. Additionally, JSON may not be as efficient for tabular data. There are alternatives to JSON, such as XML or CSV. XML offers similar hierarchical structuring capabilities but is generally less human-readable than JSON. CSV, on the other hand, is a format that stores data in tabular form with columns separated by commas and rows by line breaks. After processing data, the CSV format is used for storing the generated data. CSV is a simple text file format where each value is separated by a comma and each line represents one piece of data. Its structure is clear and easy to understand, convenient to use in different data structures, especially suitable for our experiments.

CSV, as a format stored by rows, is slow to query and inefficient to store. There are other formats, such as Parquet, which is stored by columns, that can improve storage and memory efficiency. However, this enhancement is not necessary for us.CSV's row-by-row format has its drawbacks, but it is more flexible when the data structure changes, and it is easy to read, so it is still appropriate for our needs.

3 Computational Experiments

3.1 Distributed System

A distributed system was created using six virtual machines, with one serving as the Spark master. This master node manages the other machines and hosts the code. The other machines perform tasks assigned by the master. Apache Spark was set up on this cluster, with the driver application running on the master node and four slaves acting as worker nodes [4]. An HDFS cluster was set up in a different virtual machine in a pseudo-distributed mode, hosting both the namenode and datanode on the same computer [5]. Four distinct virtual machines with different configurations were created to demonstrate vertical and horizontal scalability.

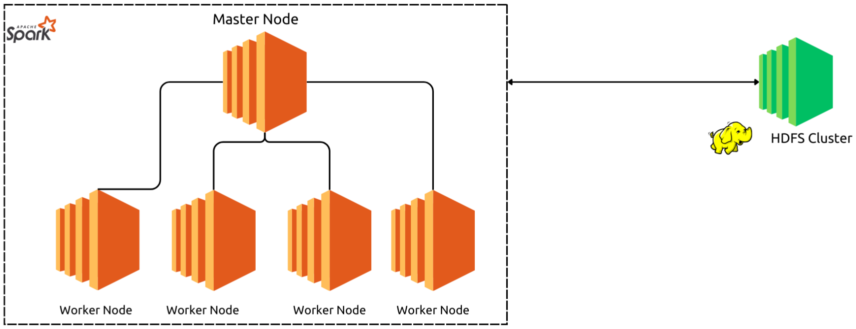

Figure 1: Distributed System Architecture

3.2 Scalability

Scalability is crucial for systems to efficiently handle increasing data volumes. Two main approaches to achieve scalability are vertical scalability (scaling up) and horizontal scalability (scaling out).[6]

1. Vertical Scalability (Scaling Up): Vertical scalability is a method that increases a system's capabilities by adding more resources like CPU and RAM. This approach enhances efficiency as data volumes increase, but it also increases the risk of hardware failures due to reliance on a single machine's resources.

2. Horizontal Scalability (Scaling Out): Horizontal scaling is a method of expanding a system by adding more instances to run processes in parallel, reducing the burden on individual servers. This fault-tolerant approach ensures continuous system operation in case of server or worker failure, enhancing efficiency and reliability.

3.3 Strong and Weak Scaling

Weak scaling and strong scaling are crucial concepts in high-performance computing, assessing efficiency and scalability. Strong scaling examines how execution time decreases with increased resources without changing problem size, while weak scaling examines changes in execution time with increasing problem size while maintaining a fixed work per process.

3.3.1 Strong Scaling Experiment

The dataset size of 19.6 GB was achieved through horizontal scaling of four worker nodes with different cores (two 2 cores,4 cores,8 cores). The computational runtime was after addition of each worker node was computed three times, and it was observed that the addition of nodes significantly decreased the runtime due to the distributed load from the master node to other worker nodes.

Table 2: Performance measurements for different core counts (Strong Scaling)

Cores	Experiment 1	Experiment 2	Experiment 3	Average
2	7.4 mins	6.1 mins	5.9 mins	6.45 mins
4	3.1 mins	4.4 mins	2.4 mins	3.30 mins
8	2.5 mins	4.1 mins	3.4 mins	3.33 mins
12	2.4 mins	2.3 mins	2.6 mins	2.43 mins
16	2.5 mins	2.1 mins	1.95 mins	2.18 mins

Table 2 presents the times for three experiments conducted with different numbers of processing cores: 2, 4, 8, 12, and 16. For each core count, the times for each experiment are listed, along with an average time. The times range from 7.4 minutes with 2 cores to 1.95 minutes with 16 cores, showing an overall decrease in time with the increase in core count, which suggests strong scaling performance.

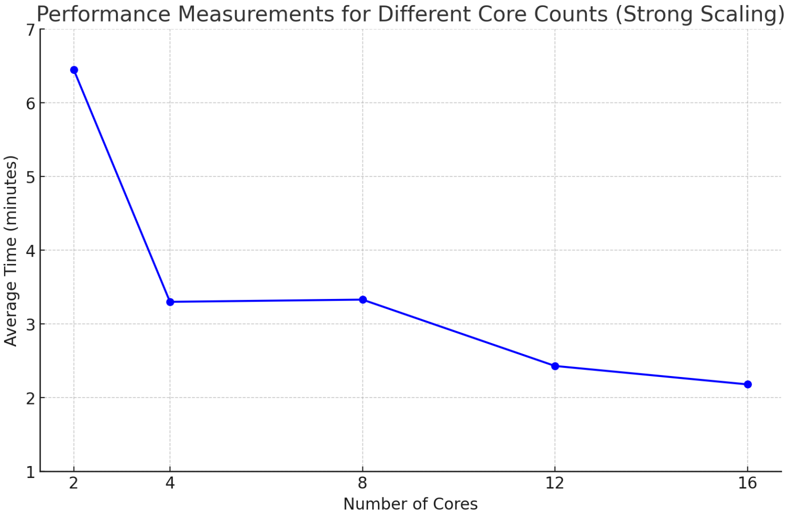

Figure 2: Performance measurements for different core counts (Strong Scaling)

Figure 2 visualizes the average times from Table 2. The x-axis represents the number of cores, and the y-axis represents the average time in minutes. The line graph shows a downward trend, indicating that as the number of cores increases, the average time for computations decreases.

3.3.2 Weak Scaling Experiment

The weak scaling was achieved by altering the dataset size and adding worker nodes from different cores. The dataset sizes were 2.28 GB, 4.57 GB, 9.14 GB, and 19.6 GB. The runtime for each datasize and worker node was run three times, and the average was calculated. The results and graph show an increase in runtime with the data size and worker nodes.

Table 3: Performance measurements for different core counts (Weak Scaling)

Cores	Size GB	Experiment 1	Experiment 2	Experiment 3	Average
2	2.28 GB	37.8 sec	28.8 sec	28.8 sec	31.8 sec
4	4.57 GB	51 sec	57 sec	55 sec	54.33 sec
8	9.14 GB	1.3 min	1.61 mins	1.68 mins	1.53 mins
16	19.6 GB	2.5 mins	2.1 mins	1.95 mins	2.18 mins

Table 3 lists the results of experiments conducted with varying numbers of processing cores: 2, 4, 8, and 16. It shows the size of the data processed, times taken for three separate experiments, and the average time for each core count. As the number of cores increases, the size of the data being processed also increases. The experiment times and averages are provided, with times being shorter for larger core counts, indicating that with more cores, larger datasets can be processed more quickly.

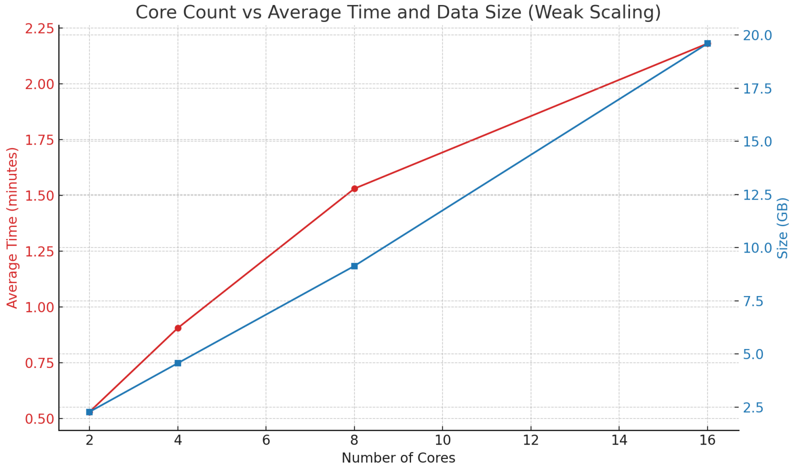

Figure 3: Core Count vs Average Time and Data Size (Weak Scaling)

Figure 3 is a dual-axis line graph that visualizes the data from Table 3. The primary y-axis (left side) and the corresponding red line show the average time in minutes needed for data processing as the core count increases. The secondary y-axis (right side) and the blue line represent the size of the data in GB. The x-axis represents the number of cores. The graph shows a positive correlation between the number of cores and data size, and a general downward trend in average processing time, indicating improved performance with higher core counts. The relationship shown suggests that as the workload increases with more cores, the system can handle it more efficiently, demonstrating the system's weak scaling capabilities.

3.4 Results

In our experiment, we initially computed the top 20 authors based on their content length, as can be seen in Table 4. Later, we filtered the dataset based on content length and grouped the entries by author name. Table 5 displays the average content length for each author.

Following this, we investigated the subreddits where authors frequently post. We filtered the data by author and counted the occurrences of each author's contributions. We then grouped the data by author again, aggregating the associated subreddits into a list. Finally, we expanded the list of subreddits for each author into individual rows. The resulting analysis is presented in the table below.

Table 4: Content Length by Author

Author	Content Length
Didimeister	9952
PmMeUrGrammerMistake	7259
iluhdatmass	7249
Atlas2088	7215
nlofe	7197
doigotoaustralia	7076
POVsocks	7052
doigotoaustralia (2nd entry)	7027
fakie_cakes	7014
supermanthrows1212	6948
redditmakesmegiggle	6829
thedeejus	6790
jackou2	6730
interfail	6713
olicope (1st entry)	6703
olicope (2nd entry)	6694
PM_ME_NUDE_BITS	6642
sorryfriend	6627
Candle_Jack_	6597
Vincethatwaspromised	6595

Table 5: Average Content Length by Author

Author	Avg(Content Length)
hollaback_girl	97.52
burncycle	120.79
Gonziago	225.00
scubamikey	181.50
The_ESC_artist	165.00
leethestud	180.00
vaughnm1	646.67
occamsrzr	124.75
twisted_spoon	398.50
malkouri	188.29
cucchiaio	378.86
MedeaDemonblood	443.18
Purrmaster	148.00
HiddenInSorrow	332.00
OIP	112.10
arcbeetle	158.33
Fearlessleader85	289.89
panthanator	337.00
ChoiceD	4.00
SpartacvsZA	806.00

Table 6: Subreddits by Author

Author	**Collect List(Subreddit)**
—JustMe—	worldnews
—annon—	TrueChristian, rapecounseling, AskReddit
–Beetlejuice–	HappyWars
–Chaos	DotA2
–MG–	DunderMifflin, clevelandcavs
–Ping–	IAmA, movies
–TT–	40k
–Unidan–	Nexus5
-10-	conspiracy, law
-3k-	Guildwars2
-AbracadaveR-	nosleep
-Ace_Rockolla-	wakfu
-Ahab-	AskReddit, AdviceAnimals

4 Discussion and Conclusion

4.1 Discussion over the results

The dataset analysis results provide detailed insights into the behaviors of authors on Reddit. As a platform for expressing knowledge, opinions and various information, authors rely heavily on posts and comments. We narrowed our focus to the top 10 authors, taking into account both post and comment content length. From there, we delved deeper into each author's average content length and also examined the subreddits in which they contributed. Our end goal was to gain a better understanding of authors based on their content length, average content length, and corresponding subreddits.

4.2 Discussion over the experiments

The study focuses on setting up different VMs for Spark master hdfs and worker nodes, using two two-core instances, a four-core, and an eight-core worker nodes. These nodes were interconnected by updating manually set up instances and manually configuring the etc/host file. Horizontal and vertical scalability were achieved by adding or removing worker nodes separately. The modular architecture ensured data consistency across worker nodes and fault tolerance.

However, the distributed system had drawbacks such as sanity checks of each instance and difficulty monitoring resources. Maintaining the same version of packages for Hadoop and Spark was challenging, and the architecture lacked self-healing properties. To avoid version conflict problems, setting up manually was easy, but automating this task could be beneficial.

To overcome these limitations, the study suggests incorporating Docker into the architecture and handling multiple Docker containers using Docker Swarm or Kubernetes [7]. Kubernetes offers self-healing capabilities, monitoring resource utilization, and maintaining the same version across all containers. Networking is also easier with Docker containers compared to manually setting up cluster networks in separate VMs, which is difficult to maintain if large number of instances are there.

4.3 Conclusion

We have successfully deployed a distributed architecture comprising six instances, of which one instance is configured with Hadoop-HDFS in pseudo-distributed mode, and the remaining five instances are configured with Apache Spark. The dataset has been meticulously hosted in Hadoop-HDFS. We have implemented both vertical and horizontal scaling in this setup, and conducted a comprehensive comparative analysis of weak scaling and strong scaling. Our empirical findings substantiate the fact that strong scaling is more efficient than weak scaling, as the runtime exhibits a significant decrease with an increase in the number of nodes. In contrast, weak scaling shows an increase in runtime with the increase in datasize and nodes. These observations unequivocally establish the superiority of strong scaling.

5 Appendix

Github Repository Link

References

[1] Syed S, et al.. Webis-tldr-17 Corpus. Zenodo; 2017. Available online at `https://doi.org/10.5281/zenodo.1043504`.

[2] Völske M, Potthast M, Syed S, Stein B. TL;DR: Mining Reddit to Learn Automatic Summarization. In: Carenini G, Cheung J, Liu F, Wang L, editors. Workshop on New Frontiers in Summarization at EMNLP 2017. Association for Computational Linguistics; 2017. p. 59-63. Available from: `https://aclanthology.org/W17-4508/`.

[3] ¨ MV, Potthast M, Syed S, Stein B. TL;DR: Mining Reddit to Learn Automatic Summarization. ResearchGate; 2017. Available online at `https://www.researchgate.net/publication/322582925_TLDR_Mining_Reddit_to_Learn_Automatic_Summarization`.

[4] Benbrahim H, Hachimi H, Amine A. Comparison between Hadoop and Spark. 2019 03.

[5] Weets JF, Kakhani MK, Kumar A. Limitations and challenges of HDFS and MapReduce. In: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT); 2015. p. 545-9.

[6] Overview of Scaling: Vertical And Horizontal Scaling;. Accessed: 2024-03-01. https://www.geeksforgeeks.org/overview-of-scaling-vertical-and-horizontal-scaling.

[7] Shah J, Dubaria D. Building Modern Clouds: Using Docker, Kubernetes Google Cloud Platform. In: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC); 2019. p. 0184-9.