Zewail City of Science and Technology

CIE 427

Big Data Analytics

Mini Project 1 - Analyzing Reddit Comments

Fatma Moanes NourEl-Din	201700346
Mohamed Elshaffei	201700030

Special Thanks to Eng. Ahmed Sameh/Elsayed and Eng. Muhammad AlAref

Contents

1 Preface

This document is not intended to be our business document. All the visualizations and the insights can be find in the presentation slides. Instead, this document is merely a documentation for our work in this mini project, our approaches, assumptions made, and justification for all the assumptions and for the way we designed our jobs.

2 Data Inspection

The dataset is quite large, so inspecting it to find trends and problems will take a huge time if each time we need to know something we run a job through the whole dataset. To workaround this problem we created a mini-dataset of 10 thousand comments sampled from the original dataset using **uniform random sampling** to ensure that our sample is as representative as possible.

This random sample was obtained using a Python script which reads from the compressed dataset at random byte offsets and smartly detect where does a comment start and where does it end to output a proper comment randomly sampled from the dataset. Note, that the file is not decompressed in this process, only the bytes that form one comment are decompressed in each iteration.

The random sample obtained was small enough to be read by a pandas dataframe to be analyzed. We analyzed all columns to get an understanding of what unique values are present in each column and what the distribution of unique values are for each column. This analysis allowed us to detect potential problems in the dataset and we used MapReduce jobs to confirm that those problems were in all of the dataset.

3 Data Analysis

By analyzing the data, we had some observations:

1. **”Controversiality”** is zero for all comments.
2. All comments have zero **”downvotes”**.
3. **”ups”** has negative values.
4. **”gilded”** has values of 0,1 only.
5. **”body”** has several [deleted] entries, links, emojis, and some entries begin with ”XXXXXXXX 123 /u/”.
6. **”Name”** always begins with t1.
7. **”parent id”** begins with t1 or t3.
8. **”id”** is part of **”Name”**
9. **t3** means it is a post, while **t1** means it is a comment/reply.

4 MapReduce Jobs

4.1 Top N Subreddits

Mapper: The mapper in this job has a very simple task, it parses the incoming line from standard input as a JSON object and simply **prints** the subreddit associated with the current comment.

Reducer: In the reducer, we need to maintain a **max heap** of the top N subreddits. However, we can not assume that there will be a single reducer that will process all the data, because if we have 54 million comments then this means that we have 54 million lines to be processed by the reducer. In this case, using multiple reducers may be a better idea, but the problem will be that each reducer will output the top N subreddits from its own perspective. Therefore, after all reducers output their top N subreddits we use the **Linux sort** command to simply concatenate and sort those files by the count of subreddits then output only top N. Here, we can safely assume that a single node will run this sort command, because the amount of data is has to handle is very small (only top 10 subreddits, for example, emitted by each reducer).

4.2 Top Topics Per Top Subreddits

Mapper: The mapper needs to filter comments by top subreddits, so it only processes a comment if it belongs to one of the top N subreddits. Firstly, we must assume that this job will run after the job that outputs the top N subreddits. the mapper will start by reading the file containing the top N subreddits to store them in a set. It is safe to assume that the mapper will read this file directly from the disk, because the file will be very small, so there is no violation to any Big Data concept.

The mapper will only process comments belonging to a top subreddit, and the **comment processing** involves:

1. Removing links
2. Removing non-alphabetic characters
3. Removing stopwords
4. Removing any word that is not a noun (the way we do this is not very smart without POS due to computation consideration, but it is computationally more efficient)

Finally for each word in the cleaned comment, the mapper will **output** the subreddit name and the word speared by a comma as a key.

Reducer: the reducer receives the sorted keys, and each reducer maintains a **max heap** of the top N topics for the current subreddit. Here, we can't assume that one reducer will be used, because the incoming data is huge and it might be a better idea to process them in multiple reducers. Therefore, each reducer will only output the top N topics for each subreddit from its own perspective. Consequently, to merge the output of all reducers into one final output we created another job that does a similar job to the first job, but where we can assume that only one reducer will be used simply because the first job will greatly filter the data and the second job will work with a much smaller amount of data.

4.3 Other Jobs

All of the other jobs work in pretty much the same way as the previously described jobs. They all share the same design choices and the same assumptions. The only difference is what does the mapper output from the comment JSON object.

4.4 How to run a job?

Each job is placed in a folder where a script specific to this job is available to run it and pass the necessary parameters to the mapper and reducer. If a task requires two jobs to produce the final output, then the script will run the first job and then feed its output to the second job.

4.5 Using the compressed dataset

To use the compressed the dataset we edited the file HADOOP_HOME/etc/hadoop/core-site.xml to add the bzip2 codec. The jobs were running with the compressed dataset used as input without any problems. However, later we discovered that we did not need to modify anything for Hadoop to read from a bzip2 file, and we did give a try by removing the bzip2 codec from the core-site.xml and once again the jobs were able to run with the compressed dataset as input without any problems.

5 Challenges and Optimization

The most difficult challenge we had to deal with, which is also concerned with optimization, is that running a job on all of the dataset especially jobs that require text processing and cleaning takes so many hours to an extent that makes us unable to test new jobs and modify existing jobs. Below is a list of steps we made to try to minize the execution time as much as possible:

1. Using Python sets instead of python lists to identify if a word is a stop-word/noun. This lead to a great reduction in time as searching in a python set is an $\mathbf{O(1)}$ operation
2. Moving as much pieces of code outside the main loop as possible to ensure that little time is wasted as much as possible
3. Forcing a single mapper to execute the whole job by setting the minimum split size to a number larger than the total dataset size. This option is only used as we test the jobs locally on our computers, because all file splits are processed on a single device, so a huge amount of time is wasted in initializing a mapper for each split. When running on a cluster this will not be a problem, because each node will process a much smaller number of splits so this option will be omitted.

6 References

We used code snippets from the below to achieve certain functionalities in our jobs.
https://stackoverflow.com/questions/11957845/unix-sort-descending-order
https://stackoverflow.com/questions/20583211/top-n-values-by-hadoop-map-reduce-code