CIS 6930 Project 2 Report

Tae Seung Kang and Yifei Sun

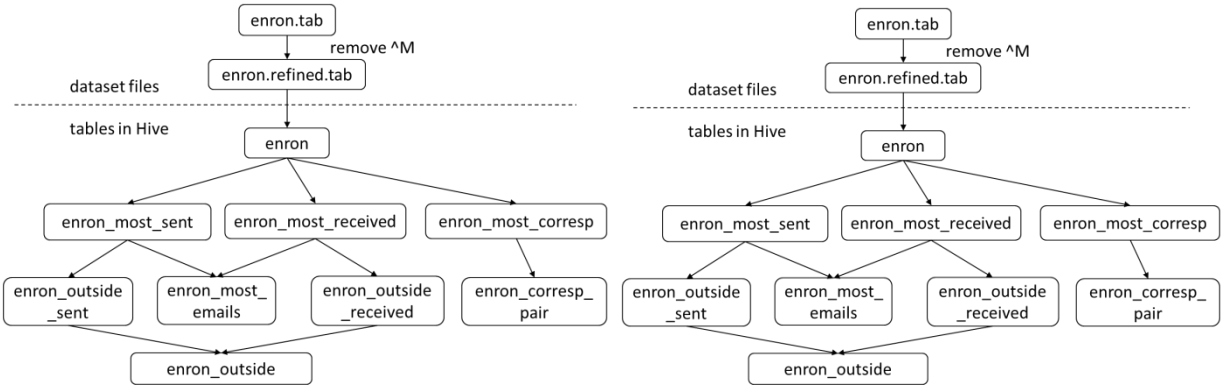
Workload division

- Yifei: visualization of the result data using Tableau and movie_per_year under /src/yifei folder.
- Tae Seung: designing data processing pipeline with table dependencies, implementing HiveQL queries, dataset processing, setting up the local Hive environments, running/testing/debugging queries with Hive on EMR, writing the report, making the presentation ppt file

A. Enron Dataset

1. What is your data processing pipeline? (Graphs and words description)

The following is the illustration of data processing pipelining with table dependencies.



The original dataset 'enron.tab' is refined to remove the character '^M' and then loaded into a table in Hive. An arrow in the figures represents the dependency of the lower table to the upper table. For example, the three tables 'enron_most_sent', 'enron_most_received', and 'enron_most_corresp' are derived from enron table. The enron table is created by /src/enron/create.enron.sql.

2. What kind of analytics do you apply on the dataset? What are the Hive queries?

We analyze the data based on the following questions. For Hive queries, refer to the attachments (/src/enron folder). The HiveQLs for base tables are in /src/enron/ken.sql and /src/enron/steve.sql.

- Who are the top-k central persons in the enron company?
  Degree centrality: (1) receives and sends most emails; /src/enron/most_received.sql, /src/enron/most_sent.sql, /src/enron/most_emails.sql (for both received and sent)
  (2) Correspond to most number of people; /src/enron/most_correspondents.sql

(3) cc column: After the recipient list in 'cc' column is included, the rank has changed when comparing to the list without cc.
e.g., steven.kean@enron.com takes the 3rd from 4th in enron_most_received.txt
This may indicate the importance of the person.

– Which person-pairs have the top-k correspondences?: /src/enron/most_corresp_pair.sql

– Who are key correspondences outside of enron?: /src/enron/outside.sql

– Who/What Ken Lay and Steve Kean writing to/about?: To answer the question, we analyzed the frequent emails they sent to and the frequent words of the subjects and the contexts of their emails. /src/enron/ken_email.sql, /src/enron/ken_subjects.sql, /src/enron/ken_words.sql

/src/enron/steven_email.sql, /src/enron/steven_subjects.sql, /src/enron/steven_words.sql

[Yifei] Script used to find out how many emails each sender sent (only show the top 1000 sender who send most), where the enron.tab file is located under the directory of /user/hduser/test_table/ in HDFS:

```
create external table enron (email_id string, time string, sender string,
receiver array<string>, cc array<string>, subject string, context string) row
format delimited fields terminated by '\t' collection items terminated by ','
stored as textfile location '/user/hduser/test_table/';
```

Then retrieve the result in a local folder:

```
insert overwrite local directory '/home/hduser/hive_result' select sender, count(*) as total_num from
enron group by sender order by total_num desc limit 1000;
```

3. Which visualization do you use on the dataset using Tableau?

[Yifei] For visualization by Tableau, I used horizontal bars, pie chart, packed bubbles to make the plots.

4. What are the programming lessons? And what are the good resources you found?

- Whenever we found a bug, we needed to rerun the relevant queries.

- Aggregate functions such as collect_set, collect_list are very useful to deal with recipient and cc column which consist of an email list separated by comma.

- Good resource for data manipulation when dealing with rows and columns.
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

- Lateral view is very useful to convert a column into multiple rows.
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+LateralView

- In the first place, it was too difficult to split a string into multiple rows with Hive. Ever since we realized that Hive provides a plenty of functions for manipulation of data (e.g., split, explode, trim) everything went well.

- It was so embarrassing that multiline comments /*..*/ are not supported in Hive scripts. Before knowing this, we kept getting an error message.

```
FAILED: ParseException line 3:0 cannot recognize input near '/' '*' 'drop'
```

- To remove duplicates of a pair such as (a, b) and (b, a), set an order rule like a < b.

[Yifei] The following websites are quite helpful:
http://archanaschangale.wordpress.com/2013/08/24/hive-installation-on-ubuntu/
https://cwiki.apache.org/confluence/display/Hive/GettingStarted
https://cwiki.apache.org/confluence/display/Hive/Tutorial

5. What is the runtime experience for queries and visualizations?

- It was surprising that inserting the enron dataset which is almost 1GB took only a few minutes.

- The runtime to execute select queries for the enron dataset is short. It usually takes 1-2 minutes for a single query.

[Yifei] Quite convenient basic tool with limited abilities.

6. What difficulties you faced and what you learned from this project?

- The data given to us didn't fit into Hive format as the enron data file contains ^M character which is considered as tab. Hence, we had to remove the character before loading to Hive table.

- Designing HiveQL queries was the challenging part. Especially, parsing columns was most difficult. This is mainly because Hive does not support abundant SQL syntax like traditional RDBMS (Oracle, MySQL, or MSSQL).

[Yifei] I have tried running hive in two ways:

(1) Create an interactive hive job flow on AWS, and then run the hive in the terminal of Ubuntu system. One difficult I found is that there is a requirement on the right of the private key (the pem file) for it to be successfully used by the ssh command. I used chmod 600 command to modify the right of the private key, then everything works fine although it runs slower and costly than running on the local machine.

(2) Run hive on a Ubuntu system.
Difficulty 1: Hadoop need to be running in order for hive to run correctly, otherwise error about metadata will appear.

Difficulty 2: After download the hive file and configure the environmental variables, I encountered a difficulty that the DDL command of hive could not be executed and an error about metadata appeared. Actually the metastore_db was not created at all. After a serial of search, I finally found that it was because the current directory from where I run hive did not have writing permission. After moving to a suitable directory, I finally get the hive running correctly.

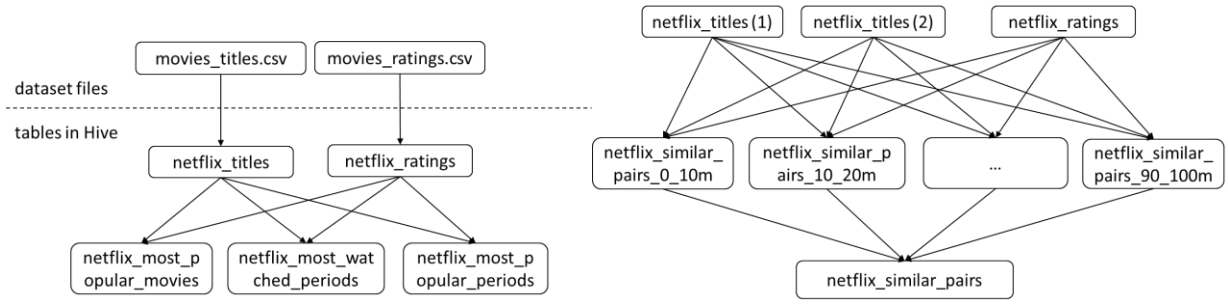
Difficulty 3: Get error “ParseException line cannot recognize input near...”. The issue isn't actually a syntax error, the Hive ParseException is just caused by a reserved keyword in Hive. Use different names and problem solved.

Difficulty 4: When I tried to export the query results from hive to a local folder by an insert command, I did not realize that I would overwrite that local folder. After the command was executed, I sadly found that I had overwritten an important folder, the home folder of the user running the Hadoop and Hive! I tried to use extundelete to recover that folder but it doesn’t work out. So I have to recreate that user account and repeat the painful configuration steps for Hadoop and Hive again as a newbie to Linux. Lesson learned: always double check before using command involving deletion operation, no easy restore options!

B. Netflix Dataset

1. What is your data processing pipeline? (Graphs and words description)

The following is the illustration of data processing pipelining with table dependencies.



The datasets ‘movies_titles.csv’ and ‘movies_ratings.csv’ are loaded into tables in Hive. An arrow in the figures represents the dependency of the lower table to the upper tables. For example, the three tables ‘netflix_most_popular_movies’, ‘netflix_most_popular_periods’, and ‘netflix_most_popular_periods’ are derived from ‘netflix_titles’ and ‘netflix_ratings’ table that are created by /src/netflix/create.titles.sql and /src/netflix/create.ratings.sql with join query, respectively. For the most similar movie pairs (netflix_similar_pairs table), we divided netflix_ratings (100M records) into 10 smaller tables of 10M records (netflix_similar_pairs_0_10m, netflix_similar_pairs_10_20m …), ran the join query on the small tables with the original table (netflix_ratings) and two netflix_titles tables, and merged the results into a single table netflix_similar_pairs (/src/netflix/most_similar_pairs_merge.sql). This had to be done because join on two big tables (100M rows * 100M rows) consumes tremendous amount of intermediate disk space incurring errors. Running the same query on EMR takes too much time.

2. What kind of analytics do you apply on the dataset? What are the Hive queries?

We analyzed the dataset based on the following questions. For Hive queries, refer to the attachments (/src/netflix folder).

– Movies from which period do people watch/love/hate the most? (50's, 60's 70's, 80's, 90's, 2000's?): /src/netflix/most_watched_periods.sql, /src/netflix/most_popular_periods.sql

– Which is the top-100 popular movies?: /src/netflix/most_popular_movies.sql

– Which movies are similar? Rating prediction?: We applied confidence of association rules.

Hypothesis: two movies X, Y are similar if most users like (hate) X like (hate) Y also and vice versa.
Top-100 similar movie pairs: /src/netflix/most_similar_pairs.sql
Calculating the similarity score of movie pair (X, Y): avg(|X.rating – Y.rating)|)
The similarity is defined as the probability that users who like (hate) movie X like (hate) movie Y following the hypothesis 1 and 2 in the slide. First, we find the entire movie pairs with the first movie which has greater than or equal to 100 votes. Then, among these pairs, we find all the movie pairs with the second movie whose rating is greater than 3 and sort by the similarity score in ascending order (/src/netflix/most_similar_pairs.sql). We set 100 votes as the minimum requirement for a movie pair to be considered as similar since too few votes easily lead to high similarity score close to 0. Hence, we did not take movie pairs with less than 100 votes into account. One of the interesting observations is that a few specific series are dominant in the results, e.g., Friends and The Twilight Zone.

[Yifei] Script used to find out how many movies are available for each year:

```
create table movie_title (id int, year int, title string)
row format delimited fields terminated by ',' stored as textfile;
load data local inpath '/home/hduser/hive_test/movie_titles.csv'
overwrite into table movie_title;

insert overwrite local directory '/home/hduser/hive_result'
select year, count(*) as total_num from movie_title
group by year order by total_num desc;
```

3. Which visualization do you use on the dataset using Tableau?

[Yifei] For visualization by Tableau, I used horizontal bars, pie chart, packed bubbles to make the plots.

4. What are the programming lessons? And what are the good resources you found?

For large tables, we had to optimize the HiveQL queries to reduce the running time. For example, when joining two large tables, we needed to reduce the rows of the first table and then iterate over the second table. Also, we rearranged the query execution order by putting a WHERE clause inside of FROM clause from outside of the FROM clause.

5. What is the runtime experience for queries and visualizations?

- Inserting the netflix movie titles, movie ratings, and most queries data took no longer than 2 minutes.
- Nested query took 3 min which is longer than non-nested query.
- join query on triple tables to get the list of similar movie pairs took the longest time.

- join on two movie_ratings tables (50M ratings and 100M ratings = 50M*100M ratings) for netflix similar pairs with greater than or equal to 100 votes (/src/netflix/most_similar_pairs.sql): failed; checksum error in 232m0.231s (4h)
- join on 5M*100M ratings for netflix similar pairs: 412m49.087s (6h 53m)
estimated time for all ratings (100M * 100M): 6h 53m * 20 = 120h + 1060m = 120h 18h = 138h = 5days 18h (only if there are no errors during execution)

- join on movie_ratings (100M) with ratings from 40M-50M lines (10M): 451m25.725s (7h 32m)
estimated time for all ratings (100M * 100M): 7h 32m * 10 = 70h + 320m = 70h + 5h 20m = 75h + 20m = 3 days 3h 20m (assuming there are no errors while running the queries)
- join on movie_ratings (100M) with ratings from 70M-80M lines (10M): 115m33.626s
- join on movie_ratings (100M) with ratings from 80M-90M lines (10M): 2h 25m
- The running time depends on the number of ratings for similar movie pairs. That is, the smaller number of ratings for each similar movie pair results in shorter running time.
- The running time for join of 10M*100M ratings varies from less than 2 hours to more than 10 hours.
- In total, it took 5 days to run the join queries of the complete set of 10M*100M ratings after many trial and errors.

6. What difficulties you faced and what you learned from this project?

- join on large dataset like movie_ratings (100M x 100M records) is challenging because of runtime and intermediate disk space. We needed to come up with a smart way to deal with the issue. When we ran the query on the original datasets, we got the checksum error or were unable to rename the output file due to lack of disk space.

To reduce the intermediate disk space, we first divided the file into ten 10M records and inserted into tables. We then ran join queries on the dataset movie_ratings (10M x 100M records) to obtain the top 100 similar movie pairs. Finally, we merged the resulting tables into a single table (UNION ALL) (/src/netflix/most_similar_pairs_merge.sql).

- Disk space issue: When the dataset is huge, the intermediate disk space did matter when running on the local Hadoop cluster. As the algorithm proceeds, it easily takes up several Gigabytes of disk space. The problem is that the underlying MapReduce framework does not deal with balancing of disk spaces across the HDFS nodes. That is, the MR framework consumes disk space of a node which already is running out of space (DFS remaining less than 1GB) rather than using ones with large remaining space (DFS remaining greater than 100GB). We have seen the case where running a join query on large netflix tables was stuck and we noticed that one of HDFS nodes has only around 300MB disk space. This was solved by running a balancer script (/bin/start-balancer.sh) and deleting other files on the node to increase the disk space. The balancer worked quite well to achieve the even distribution of disk space usage across the nodes in the cluster by moving blocks. While EMR Hive does not experience this disk space unbalance problem, the cost for S3 is a bit expensive.

- Failure of nodes: While running the queries, we encountered node crashes which included the master node. After recovering the cluster, we get the following error.

```
FAILED: RuntimeException org.apache.hadoop.ipc.RemoteException:
org.apache.hadoop.hdfs.server.namenode.SafeModeException: Cannot create directory /tmp/hive-
root/hive_2014-03-03_16-26-04_108_7701905717104262. Name node is in safe mode.
The ratio of reported blocks 0.9507 has not reached the threshold 0.9990. Safe mode will be
turned off automatically.
```

=> This was solved by hadoop dfsadmin –safemode leave