Keegan Millard

Dr. Shrideep Pallickara

CS455

19 April 2019

HW3-WC

Q1. Assume that you have complete control to reorganize and distribute the dataset. What is the fewest number of MapReduce jobs that you would need to implement the complete set of tasks in HW3-PC? Note that your program should emit outputs for each task into a separate file.

[300-400 words]

As the dataset currently is, it would be trivial to answer every question with one task. The method for doing so would be to emit output keys for each question from both a metadata mapper and analysis mapper and partition the intermediate key space such that keys for a question are reduced together.

The organization of the dataset could be changed to improve performance however. When questions use analysis data about songs and link that with an artist, it is not possible to use a combiner function. For example to answer "Q4: Which artist has the highest total time spent fading in their songs?" an intermediate key containing the fade time and song id must be emitted for every song in the analysis file, and the mapper must emit keys for each artist that contains their associated song ids such that the fade time for a song can be linked to its artist in the reducer.

Even where it seems like a combiner would be helpful the separation of the data for a song poses a problem. For example "Q3:What is the song with the highest hotttnesss (popularity) score?", here it is possible for the analysis file mapper to scan its input and only emit the song with the highest hotttnesss. But the analysis file only has the song's id so a metadata mapper must emit a key <u>for every</u> song id and song name so that in the reduce phase the song name for the single hotttessst songid can be found.

These reasons are why combining the metadata and analysis files into a single file with the metadata and analysis fields for a song being on the same line would drastically improve the usability of the dataset. Even further improvement could be made if songs for a single artist are listed together, and if a custom input format is defined so that mappers are guaranteed to have all of the songs for an artist

Q2. You are designing a new streaming service that can scale to millions of users and stream songs that are personalized to a user. How would you extend assignments HW1-PC, HW2-PC, and HW3-PC to accomplish this goal?

[400-500 words]

I would construct a network with between several hundred and several thousand endpoint nodes that users connect to. A name server would resolve DNS requests to endpoints in a manner to ensure a load balance. Each endpoint node would be connected to a song server node, an internal logging node and a node to handle queries about users. Thread pools would be utilized at every node in the network to handle requests.

An endpoint node would handle a users request to stream a song by sending a request for the song to be streamed to the users ip to its song server. This server could redirect the request to a server containing the song and a subset of the library in memory. The endpoint would also send a message containing the user and the song to its logging server which would update an internal distributed database on the users action. Apache Cassandra could be the choice for the distributed database, as it has Hadoop integration.

It would be inefficient to run MapReduce jobs for every user’s individual recommendations, so I would opt for running periodic analysis jobs that output for every song - a set of similar songs or associations and similarities between songs. I am uncertain on the details of how analysis should be done, but I envision the output of an analysis job being something that could be partitioned and stored in the memory of various internal servers.

When an endpoint gets a request for a users song recommendations, a message would be sent to the endpoint's query node. This node would query the internal distributed database and get the previous recommendation if it was created recently, otherwise it would get the users song choices and create a new recommendation. This would be done by going through the list of songs a user has listened to and getting similar songs from the servers containing the results of the periodic MapReduce analysis.

Q3. What if a family has a shared account on your streaming service? How would your streaming service recommend songs so that each family member is satisfied? Please design your own measure for satisfaction.

[300-400 words]

If a family's shared account worked such that each family member had their own "profile", I would treat each profile as a separate user behind the scenes such that no logic would need to be changed for recommendations. If this was not he case, then the ip address or hardware identifier could be logged when a user listens to songs, and this data could be used to recommend different songs based on the device they are listening on.

If one profile was used between the whole family and it was not possible to tailor recommendations to devices, changing the recommendation algorithm of the service would be necessary. Biasing recommendations for recency would allow recommendations to quickly react based on a users dislikes or skips of the songs the being recommended. This would mean that the songs a user finishes listening to kept in a small running set of "liked" songs that would be used to create a recommendation for the next song.

Alternatively, some scheme could be created where a machine learning algorithm is run on users streaming sessions to group them into clusters. Songs that belong to distinct clusters could be presented to the user upon opening the service with a question: "What do you feel like listening to today?" The answer to the question could be used to start streaming by choosing songs that are similar to the ones streamed in sessions where that song was played.

The metric for user satisfaction in the service could be the total time spent listening and maybe minimizing the number of skips or dislike actions a user takes. This metric could be used to evaluate the efficacy of one recommendation method over another through the use of A-B testing.

Q4. You are starting a music production company and you are working with local artists in town. How would you perform micro-adjustments to a song so that it is a commercial success in different countries? How would you avoid concept drifts - what is in vogue today may not be in the near future?

[300-400 words]

To perform this task I would take from my experience in HW-3PC. I would want to create a dataset containing the commercial success of songs along with certain characteristics about the songs, or perhaps its full audio file. The dataset could be stored in a HDFS cluster and then operated on with MapReduce programs.

One way of tackling the problem would be to go through all of the hot songs in a region, and using unsupervised machine learning to find what features of songs (tempo, key, rhythm, instruments used) are characteristic of the region. A program could list these features and then a human could examine the characteristics of the song in question and see if it can be modified in a way to exhibit a popular trait of songs in the region. The hit song could even be constructed from scratch by combining the hit elements of multiple regions.

Another way of changing a song to be more successful could be to use an adversarial network where one half modifies the song and the other evaluates the song based on other hit songs in the region to predict its success. The algorithm could be run on every region in the world to modify the song to reach a global maximum prediction of success.

A simpler way of modifying songs to be successful in a region could be to identify the rhythm and beat of a popular song in a region via machine learning or other means, and then supplant the original beat or rhythm with the new one. This would keep some of the songs appeal in the original region it was intended for while introducing a familiar element for listeners in a different region. Popular instruments could be identified and then used in the background to add subliminal appeal for people in the target region.

Q5. The kind of music a person likes may change/evolve over time. Describe a potential scheme to design "interest trajectories" that allows you to recommend songs that a person may like in the future.

[300-400 words]

It is important to break down what a person's interest *is* in order to track and predict its trajectory over time. Imagine distributing all songs in space, with their attributes as dimensions in that space. It is plausible that a person would listen to songs that are similar to one another in some ways. If a set of the songs a person has listened to is selected, and the positions of songs that are listened to more frequently are weighted more heavily, we could define a persons interest as the point in the song attribute space where the distance to the songs in their listened set is minimized. I would call this a person's "interest center."

If the interest center of many people is identified and tracked over time, trends could be identified. The inclusion of a song in one's interest set, could be compared with the future interest center of that person. Imagine a person listens to vocaloid music, that users interest center can be tracked over time and each individual song's contribution to the change in location of the center can be in location over time can be measured and stored as its interest derivative.

By large scale analysis of all a services users and their song choices over time, the average change in interest over time can be determined for every song. Then, a users songs choices can be examined and the interest derivative summed to estimate the users future interest center. Then songs that are near the estimate future interest center would be logical to recommend to the user as songs they would like in the future. Perhaps even farther out estimations can be made if the derivative of interest for songs near the future interest center can be used to predict an ever further out interest center and songs near it.