

203 Project Ideas – Last.fm dataset

Goals:

- Use Hadoop and related technologies at least for a part of the project.
- Ambitious goal would be to do the map visualizations.
- Present the visualizations in an extremely attractive format in the front end.

Download the datasets:

- <http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html>

Schema:

userid-timestamp-artid-artname-traid-traname.tsv

userid \t timestamp \t musicbrainz-artist-id \t artist-name \t musicbrainz-track-id \t track-name

userid-profile.tsv:

userid \t gender ('m'/'f'/empty) \t age (int/empty) \t country (str/empty) \t signup (date/empty)

- <http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-360K.html>

Schema:

File usersha1-artmbid-artname-plays.tsv:

user-mboxsha1 \t musicbrainz-artist-id \t artist-name \t plays

File usersha1-profile.tsv:

user-mboxsha1 \t gender (m/f/empty) \t age (int/empty) \t country (str/empty) \t signup (date/empty)

Common Problem 1

- Feasibility on single node/multi-node Hadoop cluster on own machine(s) OR in cloud servers using Amazon Elastic MapReduce (Paid, you can easily configure number of nodes by simply specifying the number of nodes as a command line argument)
- Trying to solve each of the idea/problem in the map-reduce way.
- ***Some pointers:***
- Mrjob: <http://pythonhosted.org/mrjob/>
- For Hadoop Installation on local machine:
<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/>
<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/>
<http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>
- NOTE that, for most out our ideas, we need to do a task for EACH user, or EACH track..etc. So try to think how it can be done in the Map-reduce way. (scatter the data, calculate the sub-results and then gather together to get combined data at the end??)
- We might find that using databases or NoSQL (mongoDB – I have some familiarity with this) may be more suitable for some tasks.

Common Problem 2

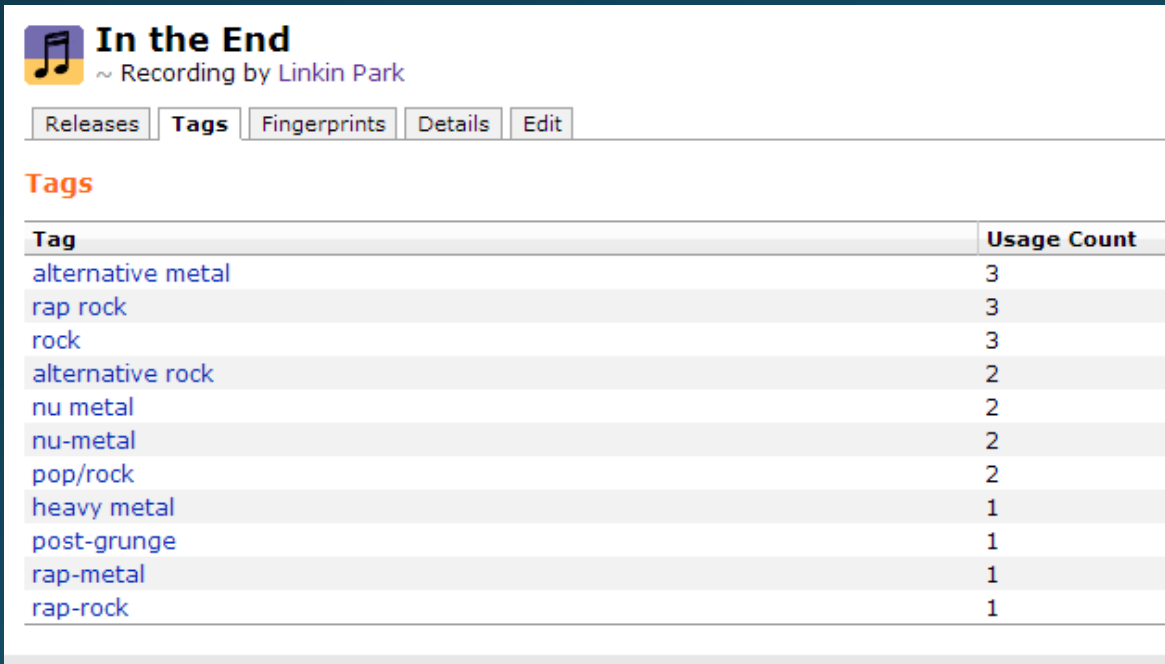
- For some ideas, we require to get 'tag' info into our data. This combining of data may be complicated to achieve consistently across all artists or tracks.
- ***Some pointers:***
- Huge MusicBrainz database available for download, which has tags for any artist you want:
http://musicbrainz.org/doc/MusicBrainz_Database/Download
example:
Artist: <http://musicbrainz.org/artist/f59c5520-5f46-4d2c-b2c4-822eabf53419/tags>
Track: <http://musicbrainz.org/recording/9d70086c-5d7a-4e7f-b1ed-c53c4b11310f/tags>

Note that, our data set also uses the same MUSICBRAINZ ID, which makes it easy to find tags for those artists/tracks that have the tag attribute present.

- This data contains tags for 505,216 tracks scrobbled on last.fm
<http://labrosa.ee.columbia.edu/millionsong/lastfm>
This million song data should have more "semantic" information about the track, as the MusicBrainz data is mostly only "genre" related tags. (DETAILS ON NEXT SLIDE)

Common Problem 2

- Music Brainz has only “genre” related tags:

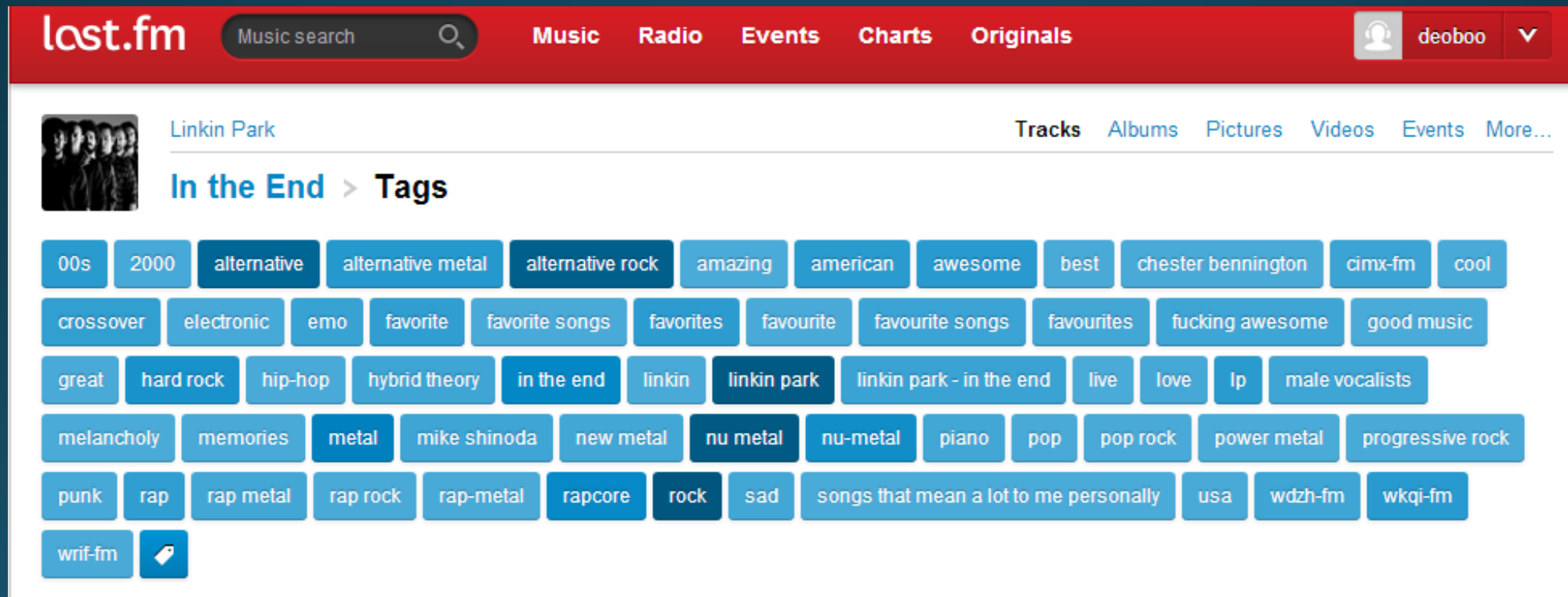


The screenshot shows the MusicBrainz interface for the song "In the End" by Linkin Park. The "Tags" tab is selected, displaying a table of genre tags and their usage counts. The table has two columns: "Tag" and "Usage Count".

Tag	Usage Count
alternative metal	3
rap rock	3
rock	3
alternative rock	2
nu metal	2
nu-metal	2
pop/rock	2
heavy metal	1
post-grunge	1
rap-metal	1
rap-rock	1

Common Problem 2

- Million song data set has limited number of tracks (still a large number), but more “semantic” tags i.e. more meaningful tags (they are actually sourced from last.fm in the million song dataset)



The screenshot shows the last.fm website interface. At the top is a red navigation bar with the last.fm logo, a search bar, and links for Music, Radio, Events, Charts, and Originals. A user profile 'deoboo' is visible in the top right. Below the navigation bar, the page is for the song 'In the End' by Linkin Park. It features a grid of blue tags representing semantic information. The tags are organized into rows and columns, with some tags like 'linkin park' and 'in the end' highlighted in a darker blue. The tags include:

- 00s, 2000, alternative, alternative metal, alternative rock, amazing, american, awesome, best, chester bennington, cimx-fm, cool
- crossover, electronic, emo, favorite, favorite songs, favorites, favourite, favourite songs, favourites, fucking awesome, good music
- great, hard rock, hip-hop, hybrid theory, in the end, linkin, linkin park, linkin park - in the end, live, love, lp, male vocalists
- melancholy, memories, metal, mike shinoda, new metal, nu metal, nu-metal, piano, pop, pop rock, power metal, progressive rock
- punk, rap, rap metal, rap rock, rap-metal, rapcore, rock, sad, songs that mean a lot to me personally, usa, wdzh-fm, wkqi-fm
- wrif-fm

Ideas

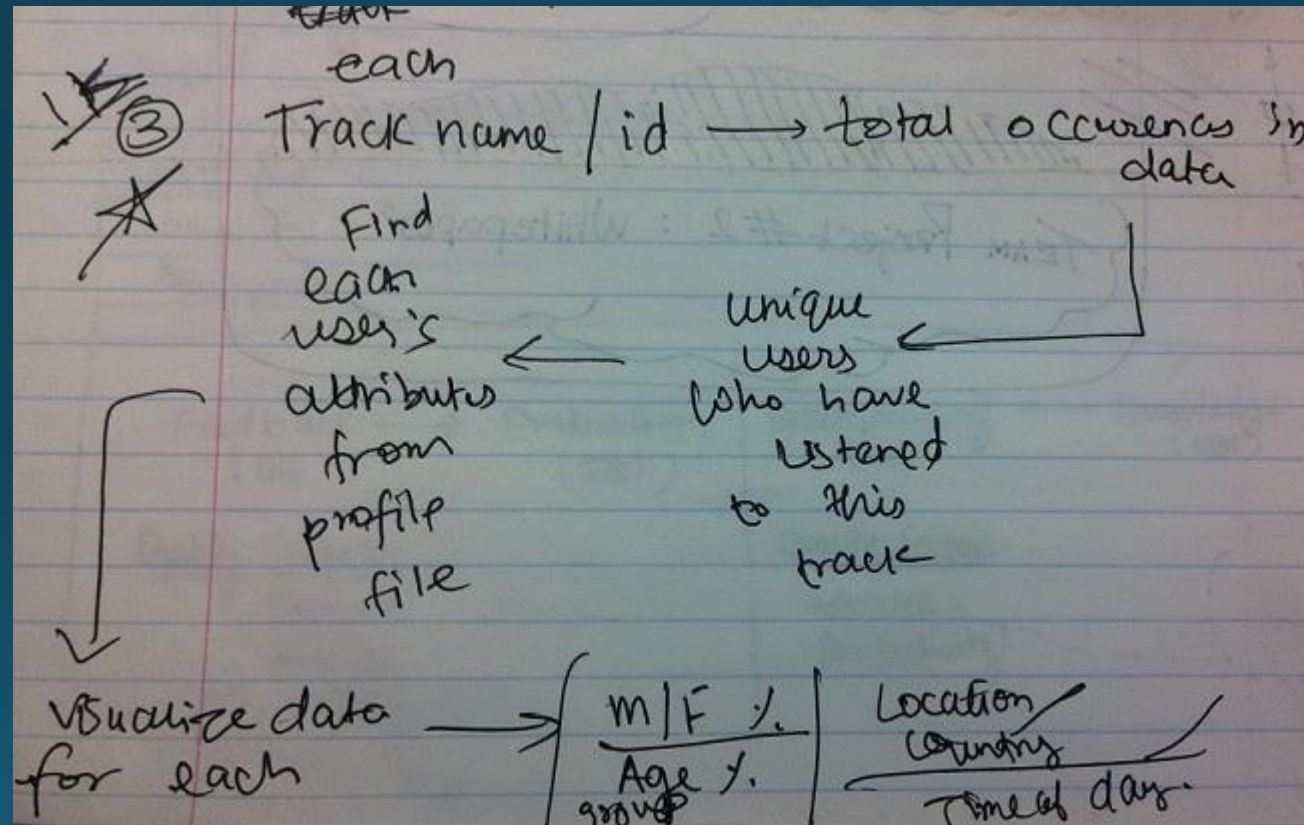
#1. Music compatibility between 2 users (360K data set)

- Find the “music compatibility” between 2 users. This would be similar to what last.fm would calculate when you are logged in and visit some other user’s profile.
- Ex: Your musical compatibility with gauravzforever is **LOW** (this level will vary as per the comparison between the top artists of the two users.....VERY LOW, HIGH, MEDIUM, VERY HIGH depending on the comparison factor we calculate)
- For this it was proposed to use Weighted Mean (http://en.wikipedia.org/wiki/Weighted_arithmetic_mean), although this can change if this does not suffice or has some pitfalls.
- To start with, we could take a small subset of the data (say of 100 users) and test out if the weighted mean method works on that.

#2. For a given track, analyze and then visualize on graph: frequency of plays as per time of the day (time range within a day), OR, as per day of the week. (1K data set)

- For this we require the track and the timestamps associated with each track.
- Pitfall: Users might scrobble more on weekends, and that's why you'll find that most tracks have more frequency on weekends.

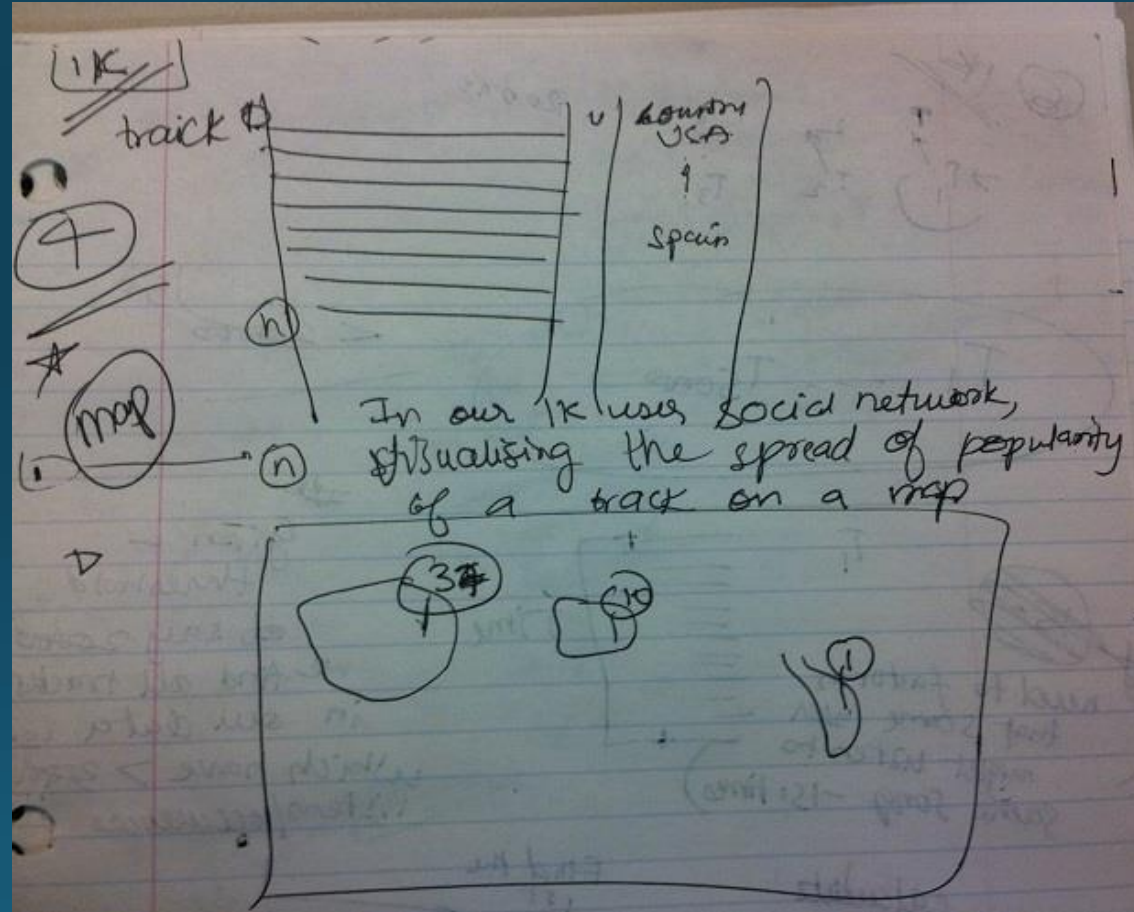
#3. For each track, find the user attributes (M/F %, Age group %, Country %, Time of day %) and visualize on pie chart or any other kind of interesting visualization. (1K data set)



#4. In our 1K user social network, visualizing the spread of popularity of a track on a map (1K data set)

- For each track , find the total occurrences in data (this will give us timestamps).
- Then, find the corresponding users listening to the track and match them in the profile file. This will give us their location.
- Add the location data to each associated timestamp.
- So for each track we will have a series of (timestamp, location) pairs. Use this time series to visualize the spread of popularity on the map.

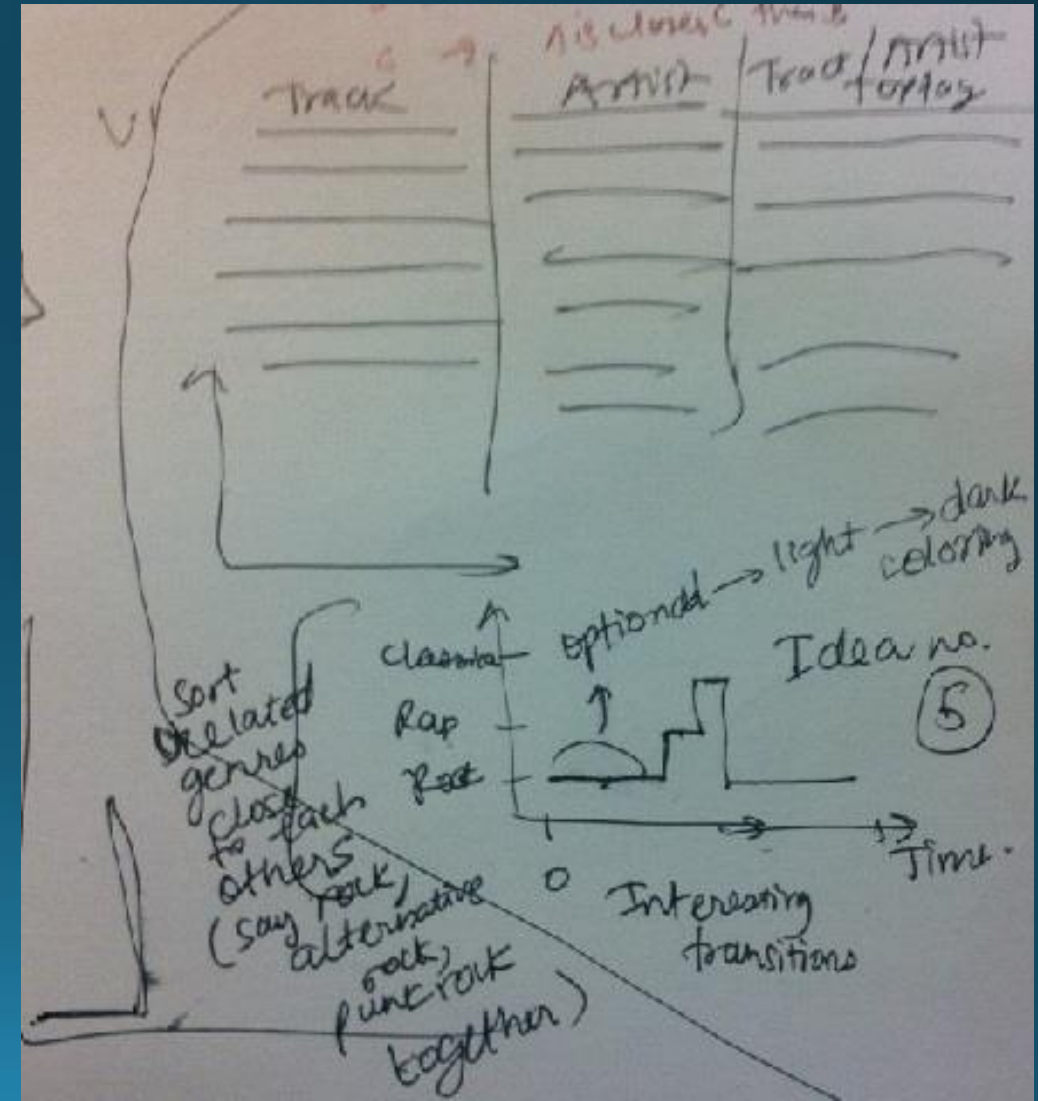
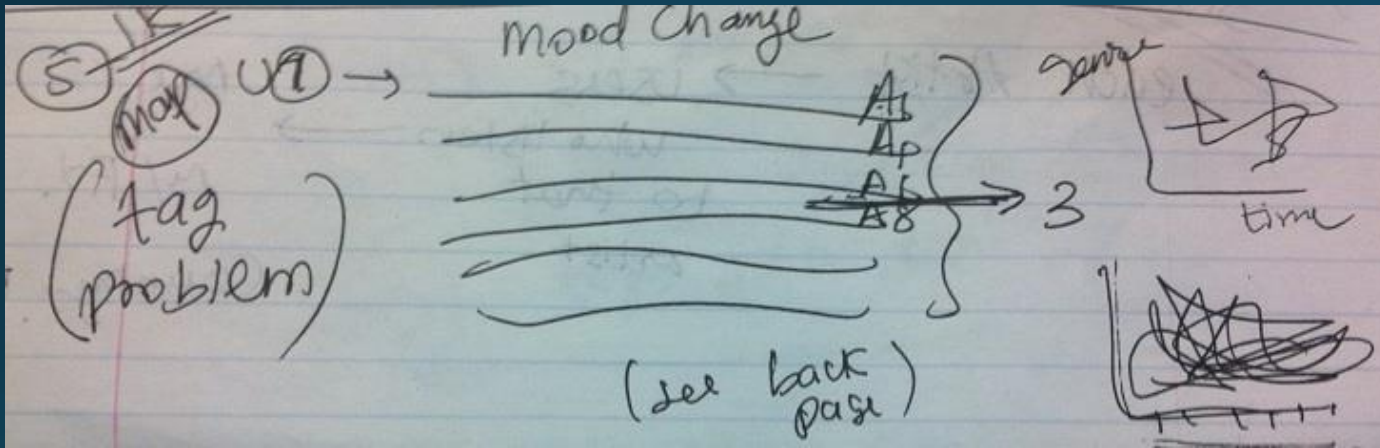
#4. In our 1K user social network, visualizing the spread of popularity of a track on a map (1K data set)



#5. Parse through a user's listening history and show the mood (genre/tag word frequency) changes (1K Data set)

- Get the listening history for each user.
- Challenge is to match the track or artist with tag/genre.
- For each user, parse through the listening history and find the associated genre or tag for each track/artist.
- With genre or tag word frequency on the Y-axis and time on the X-Axis, make a visualizations showing to show how a user's listening mood varies as a function of time. The pattern for each user will vary.

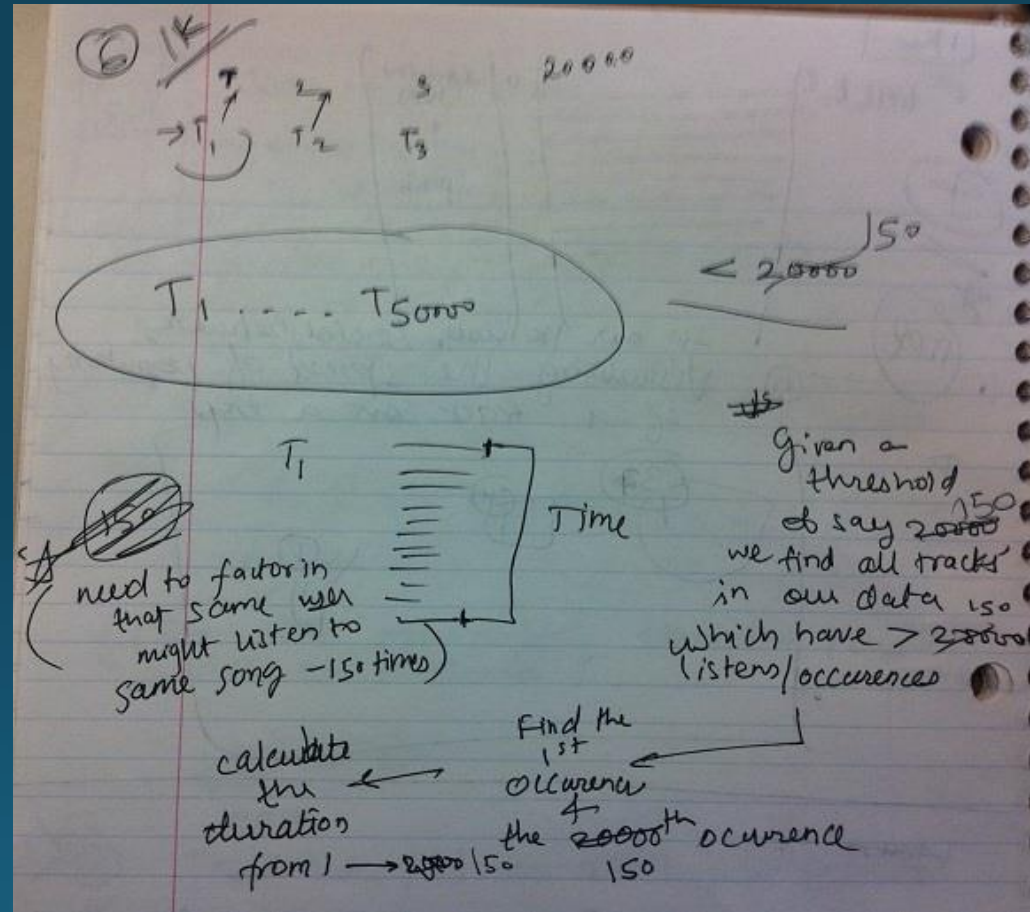
#5. Parse through a user's listening history and show the mood (genre/tag word frequency) changes (1K data set)



#6. Find the tracks which reached popularity very fast / went viral (1K data set)

- Set a threshold to measure this, say 150. This means we get the tracks which reached 150 listens the fastest.
- To get this figure, we could possibly use the average number of listens of all tracks in the system and then remove all tracks which have less than average listens. For the remaining tracks, we can then find the tracks which reached popularity very fast/ went viral.
- To do this, we find the earliest timestamp of the track and then its 150th timestamp. We take their difference. Do this for each track and then sort them.
- Pitfall: Need to factor in that SAME user might listen to same song 150 times, as this is a relatively small data set.

#6. Find the tracks which reached popularity very fast / went viral (1K data set)



#7. Similar to Idea#3, but for Artists: find the Age%, M/F %, Country % and visualize it. (360K data set)

- For each artist, get the list of users who listen to that artist.
- Match each user with their attributes in the profile file (age, sex, country..etc)
- Visualize the user attribute distribution for each artist (Age%, M/F % or country %) on pie chart or any other suitable visualization.

#8. Suggest a playlist/best tracks for a particular day/festival or as per season (1K user data)

- From the data, get a list of tracks that users have listened to on a particular day (say, valentine's day) or in a date range (on the advent of winter)
- Get the most frequent tracks from this list.
- Use that to suggest tracks to users for that particular day or season.

#9. Show the tendency of users to listen to certain types of music on a particular day/festival/season as per the tag word frequency

- Try to get tags for each track.
- Then show how the word frequency of tags changes on a particular day (say, valentines day has high occurrence of love related tags) or season (high occurrence of gloomy tags in winter).
- Visualize this in a dynamic tag cloud, or any other suitable visualization.
- This might yield very interesting results, like users in our network listen to a negative genre songs on Valentine's day.
- ALTERNATIVELY, use the song title to associate with positivity or negativity.