Interdisciplinary project in data science
Last.fm music listening data analysis

Author: Marek Furka
Main supervisors: Marcia R. Ferreira, Niklas Reisz
Co-supervisor: Prof. Allan Hanbury

May 22, 2021

1 Introduction

Many datasets these days can be analyzed in a form of a network, where some entities (nodes) are connected by some relationships (edges). There are some phenomena that can be often observed in such networks such as preferential attachment (new entities are more likely to be attached to the already popular entities than the less known ones). Another common property is forgetting (entities gradually become less popular over time), and "evergreen" phenomenon (some entities are not forgotten over time in contrast with the forgetting property). These phenomena can be observed in many domains - for example, in collaboration network of scientific articles, there is a preferential attachment pattern where authors are more likely to collaborate with more popular authors [1].

We analyzed these properties in the context of the musical domain - specifically on listening data from online music recommender service Last.fm. The motivation for this is to answer the following research questions: how is people's choice of music dependent on the age of the music and the popularity of the music? Are the patterns different for men/women, people from different geographical areas and for different music genres? We believe it is plausible that age and popularity have an influence on the choice similarly as in the case of scientific papers [1].

Last.fm tracks individual music listenings of its users from different sources such as online radios, user's computer, portable device and nowadays also streaming services like Spotify. We have combined several datasets available online as well as collected some information from an API service We have then preprocessed and analyzed this data to find out whether there are some serious problems that would make the data not usable. Next step was creation of several network graphs to get a better understanding of what the data is like and also to get a some first insights for the preferential attachment. For the preferential attachment and forgetting, we created custom approaches to calculate the intensities of these phenomena. These custom methods were also then used to compare the strengths of forgetting and preferential attachment for different subsets of the data based on its attributes (e.g. music genres). Commonly used approaches were also applied to evaluate preferential attachment, forgetting and the occurrence of evergreens.

2 Materials and methods

2.1 Dataset description and preparation

2.1.1 General description of the main dataset

The main part of the data is known as Last.fm 1K users dataset [2]. It contains information about all the song listening's of a subset of nearly 1000 users during the years 2005-2009. The attributes consist of user id, timestamp, artist id/name, track id/name. As a part of the same dataset, there is some additional information about users available in a separate file - user's gender, age, country and date of registration to last.fm. Note that the other bigger dataset made available by the same author on the same website - Last.fm 360K is not suitable for this project because it does not contain individual song listenings over time but instead just aggregated values indicating how much someone listened to an artist.

This data was collected thru the last.fm API [3]. For the API's method which was used to collect this data (user.getRecentTracks) it is needed to specify the user for each call of the method. This means that if we wanted to collect data for more users ourselves we would need to have a list of existing user ids. It is not possible to get a list of users by any other method of the API, neither it is possible to get the listenings by any other combination of API's methods (without using an external data source). An alternative would be to fetch some user names from an online source such as Reddit.com and then fetch their friends using user.getFriends method since we expect most of the users to be a part of a single connected component within the friends network structure. However, this would be very time consuming and thus out of the scope of this project.

Despite the dataset containing only 1000 users, it contains over 19 million individual records which we considered sufficient for the task.

2.1.2 Song release date information - API collection

For this project, it was also needed to have the release dates of the songs available. This information is unfortunately not part of the 1K users dataset so it was needed to get it from a different source. At first, we have used release year data for songs provided as a part of Million Song Dataset [4] [5]. Later on, we decided it would be better to have more precise information about the release times of the songs so we opted for other sources.

Originally, we tried to get the exact release dates for the songs with Last.fm API's album.getInfo() method. For this, it was first needed to get the album for each of the songs with track.getInfo() Last.fm's API method. With this method we managed to get an album id for 81.7% of songs. The data collection took over 81 hours in total, there is no issue with running multiple code instances in parallel from the same ip address in case of the Last.fm API. Unfortunately after trying calling album.getInfo() for the gathered ids we have realized that the method does not return the release date despite the documentation saying so (there are also posts on a discussion forum by people experiencing the same issue which assured us it is not a temporary issue or an incorrect method call).

Another option, which was successful was the Musicbrainz API [6]. MusicBrainz is a community-maintained open source encyclopedia of music information and it uses the same album ids as the ones we have previously collected so it was easy go get the release dates with the API's release method. Despite the API not requiring an API key it is not possible to query the API in parallel as it does not respond if there are too frequent calls from the same ip address. It is needed to stop the code execution for approximately 1 second in between the individual API calls to overcome this issue. As this would take over 50 hours on a single machine (single ip address) the code was executed in parallel on servers with different ip addresses. This way we got a release date for 50.5% of the albums. This left us with 10 737 770 data samples. 5.9% of those have release date with only yearly precision - we removed these and only kept albums for which at least the release month is known. For 15.7% of the songs, there was their first listening event earlier than the indicated release date, we remove these records as well since these release dates were most likely incorrect.

2.1.3 Properties of the resulting dataset

There are no missing values except for artist and track id attributes, because of this we use a combination of artist name together with song name as a unique song identifier. Despite the documentation saying the data ends on May 5th 2009, there are few thousands of records beyond this time point. Additionally, the first few days of May 2009 have suspiciously low number of samples compared to the previous days so we remove all the data samples from May 2009 onwards. Finally, we get 989 unique users, 284 719 unique songs and 6 357 883 data samples in the data.

2.1.4 Join with other datasets

Join with 'userid-profile.tsv' file that was provided together with the main file is perfect - the user ids are unique and we don't lose any data. There are some missing values - 706 out of 992 users have the age attribute missing. On the other hand, gender and country attributes are usable with 108 respectively 85 missing values. The gender ratio is more less balanced. In case of the countries, big majority of users are from the USA so we decide to compare only the USA against Europe in the further analysis as there are not enough users from other individual countries.

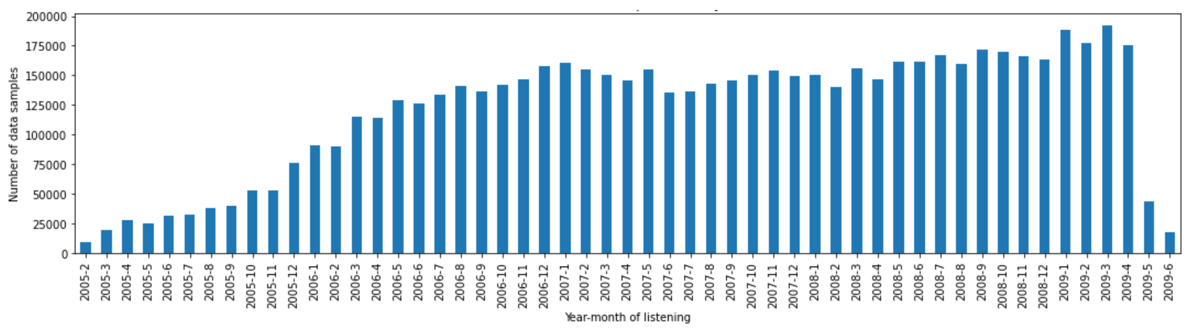

Figure 1: Number of data samples monthly

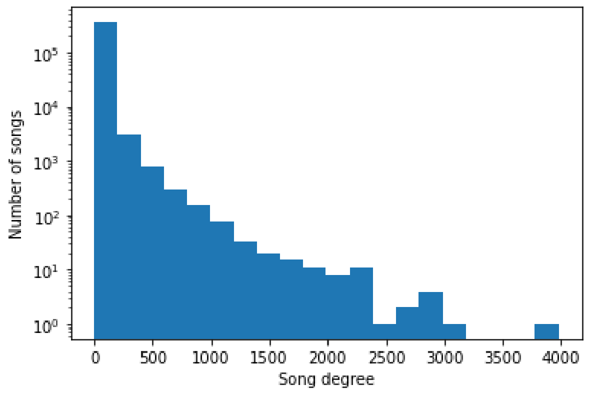

Figure 2: Histogram of song degree distribution of songs in user-song multigraph (log scaled y axis)

To get information about the genre of the songs Million Song Dataset tags dataset was used [4] [5]. Each song can have more tags assigned, for each song-tag pair there is also a value which indicates how much the tag matches the song according to Last.fm's data. To keep it simple we choose the tag with the highest value for each song. As there are very many unique song tags we keep only the songs with the top 5 most common tags so that the further analysis for these subsets has a reasonable number of samples for each subset. The top 5 tags are: rock, pop, alternative, indie and electronic (in the decreasing order from the most common one). The listenings data prepared in this way has 2 426 576 samples.

The preprocessed data is then saved to csv files - first one contains the listenings with user information (since users are a perfect join) and the second contains the listenings with tags.

2.2 Initial graph structure analysis and network metrics

To get a better understanding of the data and also to get some first insights for the preferential attachment we create several graph structures from the data. First, we create a multigraph - a graph where multiple edges between the same 2 nodes are allowed. Each edge connects a user to a song and represents an individual listening. The average degree of song vertices in this graph is 22.15 (degree of a vertex is the number of its edges). In figure 2 we can see a histogram of its degree distribution. It is clearly not evenly distributed - there are few very famous songs while there are very many unpopular songs. This is an indication of the preferential attachment pattern.

Next graph we have created was also graph with user-song edges but in this case only with distinct edges. The average degree of song vertices in this graph is 4.97 which is very low because of many almost unknown songs. The degree distribution is very similar to the one observed in the multigraph previously.

We have also plotted the degree distribution of user nodes in this graph. In figure 3 we can see the degree decreases gradually but quickly - it is unlikely someone listens to dozens of thousands of unique songs. The average user degree here is 2441.70.

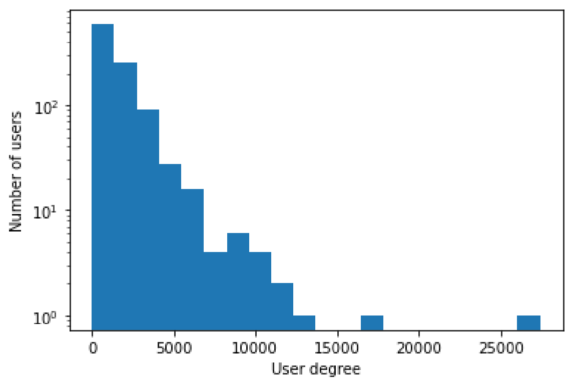

Figure 3: Histogram of degree distribution of users in user-song graph (log scaled y axis)

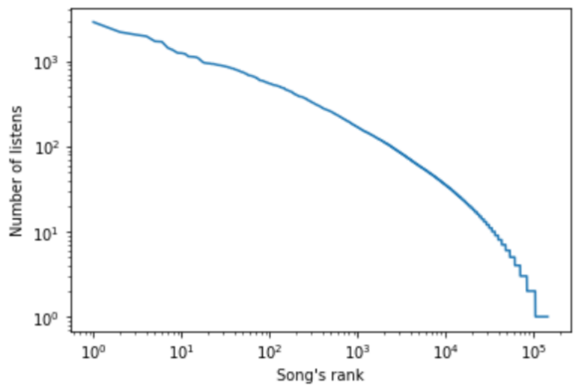

Figure 4: Song ranks against number of listens (log scaled axis)

The density of this graph is only 0.5% (density is the ratio of existing edges to the total number of possible edges). This is logical because a typical user only listens to a small fraction of existing songs.

The whole graph is a single giant connected component which was expected as it is unlikely there would be a set of users listening only to some subset of songs while not a single other person would listen to "their" songs. In real world network structures it is also often the case that vast majority of the nodes is contained within the biggest connected component of the graph [7] [8].

We have also constructed a graph with user-artist edges. The degree distributions are very similar to the ones observed in user-song graph. The average artist vertex degree is 10.82 while the average user vertex degree is 477.51 which is a bit surprising. This means that in the average a user listens to that many unique artists which is probably caused by the relatively long period of time over which a user can actually accumulate that many unique artist listenings. The density of the graph is a bit higher than for the songs - 1.1%.

2.3 Preferential attachment

For the analysis of preferential attachment we only use the songs released after the beginning of the dataset. First, in figure 4 we have plotted song ranks (where the most listened song has a rank of 0, second has 1 etc.) vs. the number of listens. We can see the number of listens goes down extremely quickly and there is the so called long tail of the unpopular songs (a few very popular items and very many unpopular items).

We have developed a custom function to calculate the intensity of the preferential attachment in the dataset (if there is any). The idea is to loop over the listening events ordered by the timestamps in ascending order. At each step, if the song was already listened to at least once we calculate 2 kinds of probabilities. Random probability as 1 divided by the total number of songs that have appeared so far - this represents the chance of choosing the particular song if the user's decision is not influenced by preferential attachment. The second probability is calculated as the number of listens of the particular song up to that point in time divided by the total number of listens of all songs up to that point in time. At the end, we calculate the sums of all these individual probabilities for both kinds and divide the sum of the popularity weighted probability by the random probability. The measure is also defined in equation 1, where $listenCnt_{ij}$ represents number of listenings of a song j at at intrinsic time i, $listenCntSum_i$ represents the total number of listenings of all songs at at intrinsic time i, $songsAppeared_i$ represents the total number of songs that appeared up to at intrinsic time i, n being the total number of listenings, m being the total number of songs and α_{ij} has a value of 1 if song j was listened to at intrinsic time i, otherwise it has a value 0.

$$preferentialAttachmentIntensity = \frac{\sum_i^n \sum_j^m (listenCnt_{ij}/listenCntSum_i) * \alpha_{ij}}{\sum_i^n \sum_j^m 1/songsAppeared_i} \tag{1}$$

If there is no preferential attachment this should give us a value near 1 which we have in fact tested on a randomly generated song listenings data. We have also created two other random artificial datasets. One where the probability of a song listening was proportional to the number of listens of the song so far, for this dataset the resulting value was 1.97. The third test dataset was generated in the same manner as the previous one, but the difference is we doubled the preferential attachment intensity by counting each listening as 2 listenings. For this dataset, we got a result of 3.01 which shows how the value increases with increasing preferential attachment. The advantages of the approach are that it makes use of the continuous time attribute and it naturally deals with the issue that the number of users in the dataset is rising over time (which means that a song with the same number of listens in 2005 is not as popular as a song with same number of listens in 2009).

With the own proposed method, we have quantified and compared the preferential attachment intensity within the data subsets. We wanted to also estimate the functions that describe preferential attachment and forgetting in this dataset. We use a method originally introduced by Newman [1] to estimate the shape of the attachment kernel. For this, it was first needed to put the data in a compatible format - an Sql lite db file which contains 2 tables. Nodes table containing all the songs with name (unique song name as its name combined with the artist's name), realtime (song's release date), id_nb (unique song id), intrinstic time (indicates the order of the release dates), timebin (the bin represents each of the months over the 4 years of data).

The other table is called stream and contains information about the listenings with attributes realtime (the listening timestamp), node (the id of the song), node_origin (unused in this domain), intrinstic time (indicates the order of the listenings) and timebin (represents each of the months over the 4 years of data). The code then produces the 3-dimensional plot in figure 7 with time in months (t) and number of listens of song (k) on the 2 of the horizontal dimensions and probability of a song listening on the vertical axis.

2.4 Forgetting and evergreens

For forgetting we have first taken look at the top 5 most listened songs from 2006 by plotting their number of listens per month over time. In figure 5 we can see each of the songs has some peak of popularity after which the number of listens starts decreasing which indicates forgetting can be observed in our data.

To compute the intensity of the forgetting we have developed a similar function as for the preferential attachment. Again we loop over the listening events ordered by the timestamp in an ascending order. At each step, we calculate the song's age and the total age of all songs that were listened to so far. Similarly, as before we calculate two kinds of probabilities - random which is again 1 divided by the total number of songs that have appeared up to that point in time. The other one is proportional to the song's age and is calculated as the song's age divided by the total age of all songs. Similarly as before - if the age of the song did not influence the user's song selection, then the sum of these two kinds of probabilities should be equal at the end. Again we have generated some random listening data to test the function which resulted in nearly the same sums of these 2 probabilities. The function returns the sum of the random probabilities divided by the age weighted probability (vice versa as in case of preferential attachment) so that higher value of the function means stronger forgetting. The measure is also defined in equation 2, where $songAge_{ij}$ represents the age of a song j at intrinsic time i, $totalSongAge_i$ represents the total sum of ages of all songs at intrinsic time i, $songsAppeared_i$ represents the total number of songs that appeared up to the intrinsic time i, n being the total number

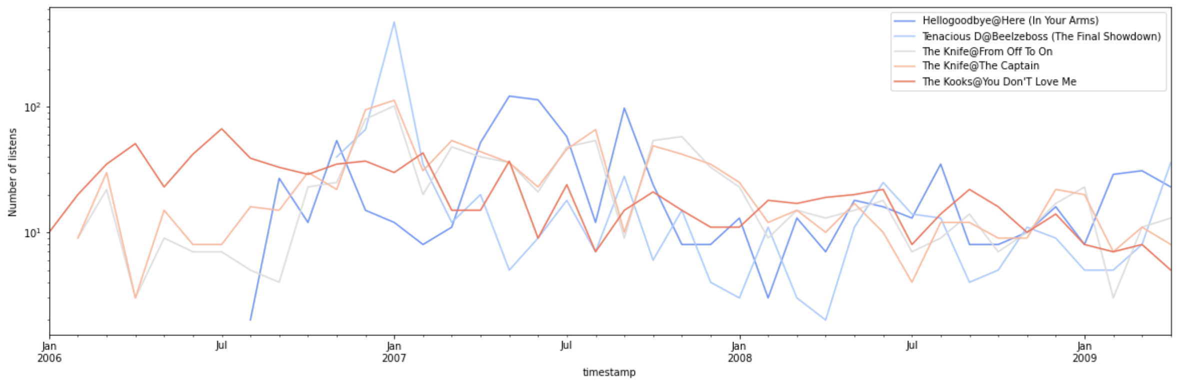

Figure 5: Top 5 most listened songs in 2006 - number of listens over time

of listenings, m being the total number of songs and α_{ij} has a value of 1 if song j was listened to at intrinsic time i, otherwise it has a value 0.

$$forgettingIntensity = \frac{\sum_i^n \sum_j^m (1/songsAppeared_i) * \alpha_{ij}}{\sum_i^n \sum_j^m songAge_{ij}/totalSongAge_i} \tag{2}$$

Initially, the function was extremely slow as we were calculating the total age of all songs by subtracting the song's release date from the current time and summing these values. We have solved this by calculating the total age of the songs by adding at each loop step the time elapsed from the previous timestamp multiplied by the number of songs that have appeared so far (i.e. how much all the songs got older). With this improvement, the function is easily usable on millions of data samples.

For the analysis of the evergreens, we have applied the method to calculate the values for each song (we consider the probabilities only for each song separately). For this kind of analysis, we have only considered the songs that had at least 200 listens in total so we minimize the chances of some songs being marked as evergreens due to randomness.

3 Results

3.1 Dataset preparation

The results of the project are: the data preparation which in the data exploration phase shown some minor issues in the main data file [2] such as some of the records being outside of the time range specified in the data description which was solved by removal of those problematic records. A major part of the dataset preparation consisted of data collection from API services. This step was needed because other datasets available, which we also experimented with at first, did not offer enough granularity for the song release date attribute. We have discovered an issue with the last.fm API [3] which is that it does not return a release date for an album despite its documentation saying so. Instead, we have used Musibrainz API [6] to get the release dates. As the API services are slow (or in the case of Musicbrainz not reply in case of too frequent request), the API data collection was done in parallel.

We have also joined the main dataset with users data which has information about all the users in the main dataset and its attributes gender and country did not contain many missing values so they were suitable for further usage in the analysis. Another dataset which we joined to the main one was the tag dataset available on the Million song dataset's website [4][5]. As each song can have more tags (song genres) we have used only the tag with the highest value for each of the songs (the most matching one). There were very many distinct tags in the data so for further analysis we decided to only use the 5 most common ones to have a reasonable number of samples for these subsets.

3.2 Network metrics

In the next step, we have created multiple graph structures from the data and calculated some properties for these networks. This analysis of degree distribution histogram has shown that there are

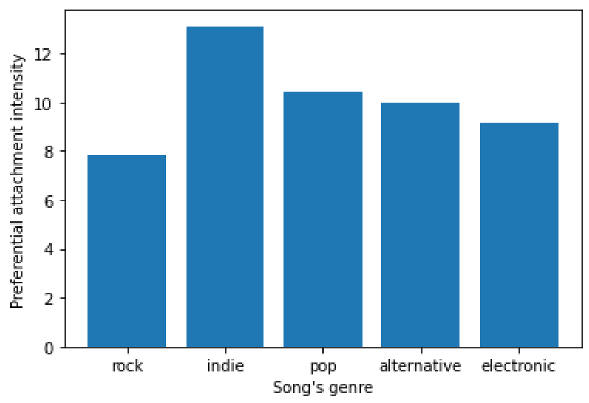

Figure 6: Song genres - preferential attachment

relatively few very popular songs and very many unpopular songs which was the first sign of preferential attachment. Other interesting findings were the low densities of the graphs - 0.5% for the graph with distinct user-song edges and 1.1% for distinct user-artist edges. These numbers are not too surprising as one would expect that a person only listens to a small fraction of existing songs. More details can be found in the corresponding subsection of ”Materials and methods”.

3.3 Preferential attachment

The own proposed method has shown there is a certain amount of preferential attachment in the data. For the general dataset, we get a value of 4.78 which tells us the user's song choice is somehow influenced by the song's popularity at that moment.

We have also used the method to calculate the values for different subsets of the data. For males only we get value of 4.18 while for females we get 6.51. When splitting the data based on geographical regions of the users we don't see much of a difference - we get 5.13 for USA and 4.57 for Europe. In terms of song genres, as we can see in figure 6, indie genre seems a bit stronger in terms of preferential attachment. Surprisingly it is not the pop genre as one would expect. It is also important to note that the dataset containing the tags is smaller in size thus the results are less reliable.

For estimating the functions that describe preferential attachment and forgetting we have used the method originally proposed by Newman [1] which resulted in the 3-dimensional plot in figure 7. For any k value we can see steep decline with increasing time values which indicates the forgetting phenomenon. The forgetting is particularly strong in the first few months after song is released (the steep ”hill” in the plot). The ”valley” in the left part of the figure is caused by lack of data as there are not any songs in our data which would be listened to that much at the time when they are almost new (low t and high k). For any value of t we can observe an increase for increasing values of k which indicates the preferential attachment phenomenon.

We can see the relationship of the probability and k in detail in figure 8, where we see the probabilities for different k values when t is fixed at 3 together with the orange points representing polynomial regression fit. We can observe the influence of an increase of k is a bit lower when k is higher.

3.4 Forgetting and evergreens

By using the proposed method we get, for the general dataset, a value of 1.12 so the older songs have a lower chance of being picked by a user which indicates the presence of the forgetting phenomenon. Similarly, as in the case of preferential attachment, we calculate the values for subsets of the data. For males only we get the value of 1.15 while for females we get only 1.07 so that women are probably a bit more likely to listen to older songs. For users from different countries, there does not seem to be a very noticeable difference - we get a value of 1.13 for the USA and 1.11 for Europe.

In figure 9 we can see the forgetting intensity for the top 5 most frequent genres. We can clearly see forgetting seems to be a bit stronger for pop music, which is intuitive since pop songs are generally popular for a shorter period of time and not much listened to as they get older. Surprisingly there is no forgetting for indie, rock and alternative genres according to the data. It might be the case

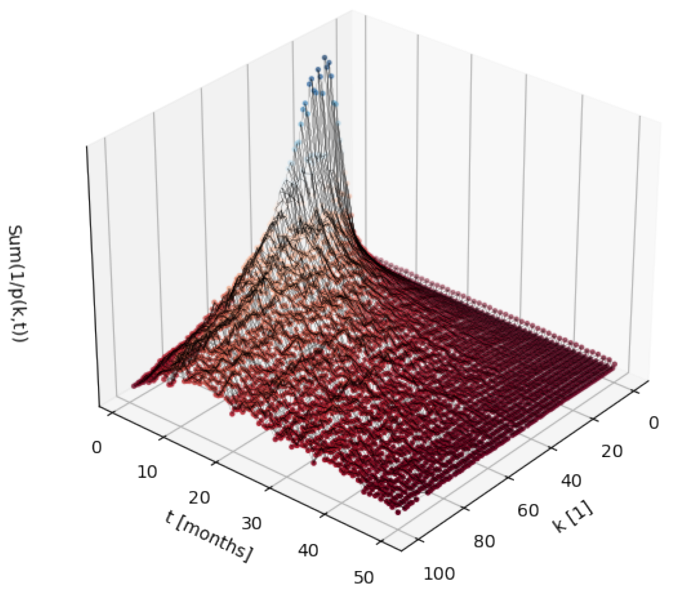

Figure 7: Probability of song listening for different ages and popularities

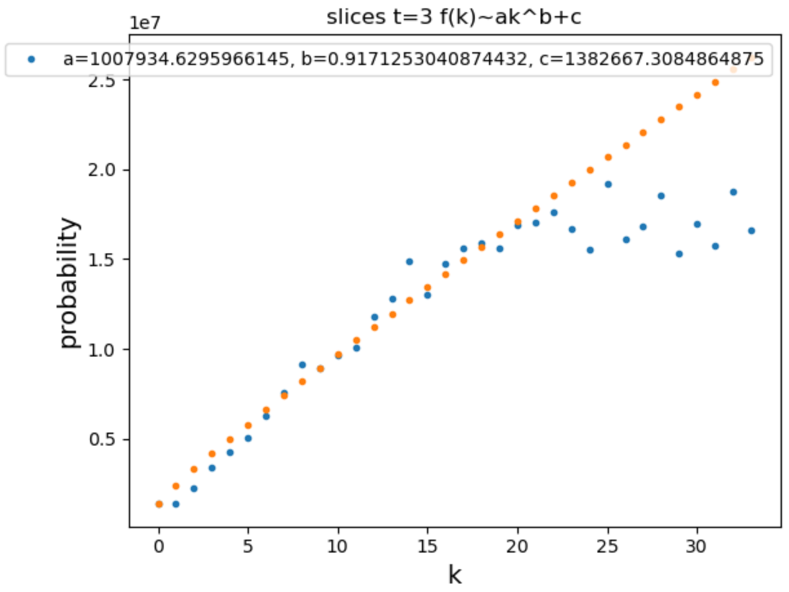

Figure 8: Detailed view of figure 7 - probabilities for different k values when t is fixed at 3

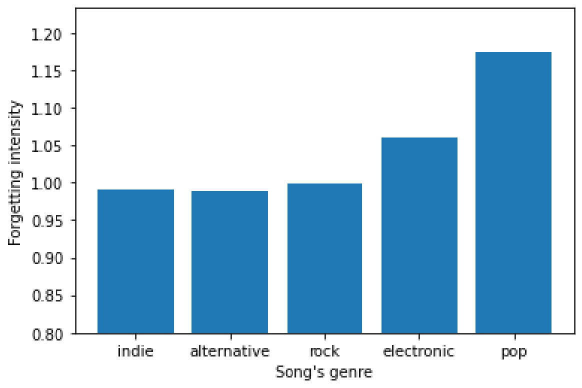

Figure 9: Song genres - forgetting

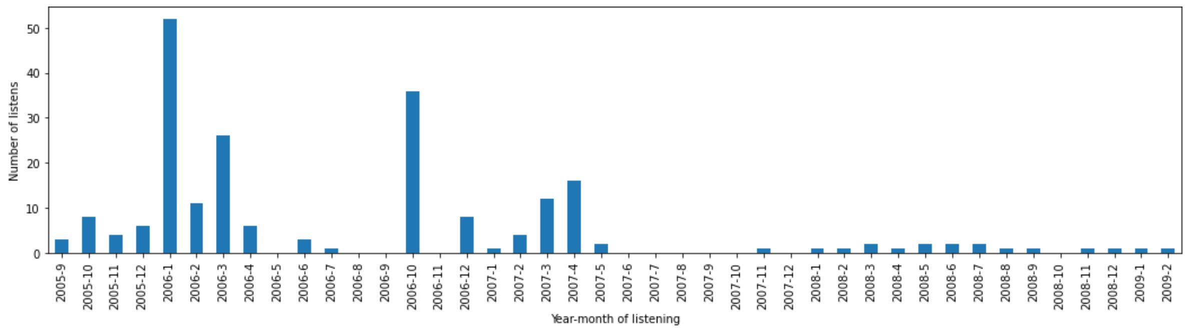

Figure 10: The monthly listen counts of the "most forgotten" song - Isho by

the preferential attachment overpowers the forgetting - e.g. out of 20 most listened rock songs in the dataset 18 were released in the year 2005. As there is preferential attachment for these songs, people listen to these popular songs and even if they are listened to a bit less over the time they are probably still preferred over new but unpopular songs in general which causes the resulting forgetting value to be slightly lower than 1 (values lower than 1 represent the opposite of preferential attachment when older songs are preferred).

In figure 10 we can see the monthly listens of the most forgotten song according to our metric. We can see its listens went to almost 0 after some time. In figure 11 we can see the monthly listens of the least forgotten song according to our metric (the biggest so called evergreen). We can see that the song is even after few years still getting a bit of some attention. The top 5 evergreens are listed in table 1.

In figure 12 we can observe the number of listens from the point of view of the last month - i.e. how old were the songs which were listened to at that time. We can see that the most of the songs

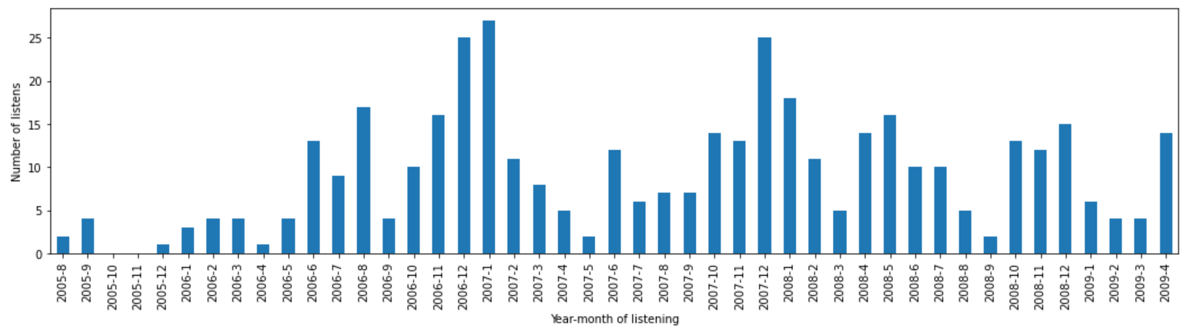

Figure 11: The monthly listen counts of the "least forgotten" song - The Flowers by Regina Spektor

Artist	Song name
Regina Spektor	The Flowers
M.I.A.	Amazon
Klaxons	Four Horsemen Of 2012
Death From Above 1979	Dead Womb
Between The Buried And Me	Selkies: The Endless Obsession

Table 1: The top 5 least forgotten songs (the biggest evergreens)

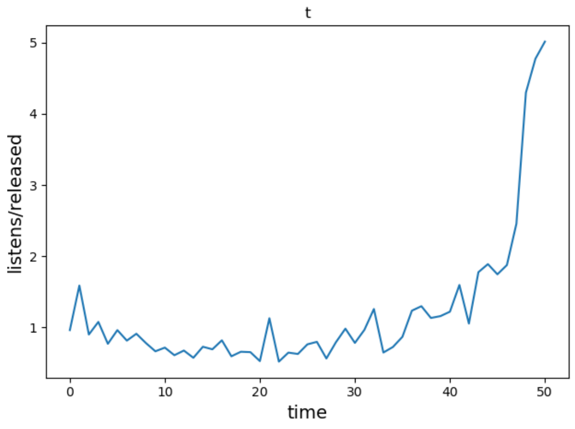

Figure 12: Listens vs. time relative to the last month

that were listened to were very recent. The number of listens drops quickly for the older songs - the ones approximately 5 months and older are not that likely to be listened to.

As already briefly described in the preferential attachment subsection, in figure 7 we can also observe forgetting. There is a clear declining pattern with increasing t for basically any value of k. In figure 13 we can see the relationship of time and the probability in detail for particular k which is 20 in this case. We see that the probability decreases very quickly in the first year after the song is released.

For the analysis of evergreens, we have used the custom forgetting method which evaluates the forgetting intensity for each song individually. The songs with the smallest forgetting value are then the so called evergreens. We see the evolution of monthly listen counts over time for the biggest evergreen as well as for the most forgotten song in figure 11 respectively 10. In table 1 we can see the song names and artist names of the top 5 evergreens identified this way.

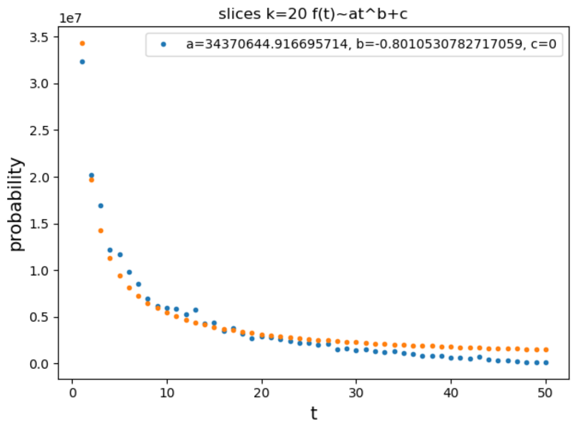

Figure 13: Detailed view of figure 7 - probabilities for different t values when k is fixed at 20

The code with all the results mentioned (also some additional results) is available in a public Github repository. The repository structure is explained in its readme file [9].

4 Discussion

Despite getting some interesting conclusions from the analysis it is debatable whether the data sample size was sufficient as it has decreased after all the preprocessing steps and joins with other data sources. In some cases, this had directly noticeable implications such as the problems with the lack of data for the 3-dimensional plot in figure 7 described in detail in the corresponding chapter. Another example is the evergreen analysis where we can see in figures 11 and 10 that we only have a few dozens of listens monthly with some irregular patterns caused by the fact that there are only 992 users in the data.

An alternative and possibly better approach for getting the song genres (tags) would be to use the Last.fm API instead of the Million song dataset. This could slightly increase the resulting dataset size. The reason we have opted for the already prepared dataset was to avoid another time consuming API collection procedure.

Another different data collection procedure would be to fetch some user names from an online source such as Reddit.com and then fetch their friends using Last.fm's API user.getFriends() method since we expect most of the users to be a part of a single connected component within the friends network structure. Then with the user ids known we could get their listening information. This could possibly result in a much bigger dataset, however it would be very time consuming and thus out of the scope of this project.

We have answered the initially proposed research questions - people's choice of music is clearly influenced by the music's popularity and age. The more popular songs are more likely to be listened to and the songs released within approximately the last 5 months are a lot more likely to be listened than the older ones. There seems to be no difference to these phenomena for men/women and different geographical areas. However, based on our research the phenomena have different intensities for different music genres. It is no surprise that people are more likely to listen to songs that are famous, perhaps the songs that they hear in the radio or maybe the ones their friends recommended them. The forgetting is also expected, people possibly get bored by listening to the same song after some time.

Overall we consider this project as a solid foundation and source of information for future work in the area of music listening analysis. Based on this work, one can model the dynamics of this network.

References

[1] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2), Jul 2001. ISSN 1095-3787. doi: 10.1103/physreve.64.025102. URL `http://dx.doi.org/10.1103/PhysRevE.64.025102`.

[2] Òscar Celma. Last.fm dataset - 1k users. `http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html`, 2010.

[3] Last.fm. Last.fm music discovery api. `https://www.last.fm/api`.

[4] Million song dataset. `http://millionsongdataset.com/`.

[5] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[6] MusicBrainz. Musicbrainz api. `https://musicbrainz.org/doc/MusicBrainz_API`.

[7] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *CoRR*, abs/1111.4503, 2011. URL `http://arxiv.org/abs/1111.4503`.

[8] Seth A. Myers, Aneesh Sharma, Pankaj Gupta, and Jimmy Lin. Information network or social network? the structure of the twitter follow graph. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, page 493–498, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327459. doi: 10.1145/2567948.2576939. URL `https://doi.org/10.1145/2567948.2576939`.

[9] Marek Furka. Interdisciplinary project last fm. https://github.com/maref1/interdisciplinary_project_last_fm, 2021.