Data gathering process of *Forecast emerging artists success on Last.fm service: a data-driven study*

Alexandra Lavinia Bradan

May 2020

The data figured in my work was obtained by following the described process. First of all, I manually retrieved a list of 100 music charts from Last.fm platform[1], mainly involving new artists (competitors in talent shows like X-factor, The Voice, Amici, American Idol or musical festival like Sanremo Giovani or Eurovision) and new musical genres (trap, electronic, rap, hip-hop, alternative), geographically contextualized (Europe, America, Africa, Asia, Australia). The chart lists can be found in the *artists_charts_links.csv* file, in the form of urls. Though an HTTP connection provided by the Python Library Urllib[2], I iterated over each url, retrieving the HTML code, parsing it using the Python Library BeautifulSoup4[3] and collecting the artists figuring in the charts. The retrieved artists can be found in the *artists.csv* file. Next, I iterated over this file, redoing the previous scrapping process for every artist figuring in it, this time to retrieve each artist's top listeners, by using the following parametric url:

https://www.last.fm/music/*artist_name*/listeners

The retrieved users are store in the *node_map1.csv* file and represent the social network partially analyzed (we will see shortly that some users will be filtered). Indeed, I collect for every user and artist their associated info,

[1]https://www.last.fm/
[2]https://docs.python.org/3/library/urllib.html
[3]https://pypi.org/project/beautifulsoup4

subscribing a developer account on Last.fm service[4] to use the supplied API. The following methods were mainly used:

1. artist.getInfo: returns an artist metadata (listeners, play-count, similar artists, tags);
2. artist.getTopTags: returns the top tags for an artist, ordered by popularity;
3. user.getInfo: returns information about a user profile (country, age, gender, date of registration, play-count, playlists);
4. user.getFriends: returns a user's list of friends[5].

Having the data associated with users and artists made possible to filter out users without friendship ties and artists which could be classified in a main music genre. Artists' classification in a main music genre was done considering a pool of 13 main music genres: 1. alternative; 2. blues; 3. classical;4. country; 5. dance; 6. electronic; 7. hip-hop/rap; 8. jazz; 9. latin; 10. pop; 11. r&b/soul;12. reggae; 13. rock.

Last.fm indeed allows assigning personal tags to each artist to describe him and his music, so the result is a messed up list of tags for every artist. In this scenario, I had first of all to outgo a mapping process and map every music tag encountered in the restrict music genre set with which I choose to work. I iterate over each artist's tag list, filtering the words referring to a musical genre, thanks to a manually constructed dictionary[6], counting 1,453 words mapped in the 13 main music genres and adequate to classify the relevant artists. The file containing my self-built dictionary is *merged_music_tags_map.json*.

Since a conspicuous part of the retrieved artists were listened only by a handful of users, I narrowed the final artist set further more, too. Final

[4]https://www.last.fm/api/

[5]Since 2017, Last.fm has introduced asymmetric friendship ties, so the list returned by the API corresponds to user's followings. A follower API isn't still provided, but can be easily implemented following the scrapping process previously described.

[6]The dictionary was build consulting the work done in *https* : *//www.researchgate.net/publication/*307878345 and *https* : *//link.springer.com/article/*10.1007/*s*11042 − 018 − 6511 − 6, as well as the provided list found at *https* : *//www.musicgenreslist.com/* and *https* : *//en.wikipedia.org/wiki/List_of_music_styles#Avant − gard*.

users and their friendship bound can be found in the *filtered_edge_list.csv* file, while final artists and their main music tag[7] can be found in the *filtered_artists_with_main_tag.csv* file. The first file is encoded in the following format:

user_encoding\t other_user_encoding\n

where *user_encoding* is a uni-vocal integer associated with an user [8] and *other_user_encoding* is the uni-vocal integer associated with the user followed by the first one. The second file is encoded in the following format:

artist_encoding\t main_music_genre_encoding\n

where *artist_encoding* is a uni-vocal integer associated with an artist[9] and *main_music_genre_encoding* a uni-vocal integer associated with his main music genre[10]. Artists impossible to classify in a main music genre are stored in the *filtered_artists_without_main_tag.csv* file.

All users and artists info can be found in the *seed_user_info.csv* and *filtered_target_artists_info.tar.xz* files. Users' info is encoded with the format:

adopter\t gender\t age\t country\t playcount\t registered_on\t friends\n

Artists' info are encoded in a dictionary form, one file per artist and whose key attributes are self explanatory. Users' listening can be found, instead, in the file *week_user_artist_count.tzr.xz*, encoded as follows:

week_encoding\t user_encoding\t artist_encoding\t playcount\n

where the first column is the timestamp[11] of the action; the second column is the user id; the third column is the artist id and the fourth column contains the number of listening made by the user of that particular artist in that particular period. To retain only the most active users and most adopted artists, the file *adoption_log.csv* contains a subset of the adoptions present in the *week_user_artist_count.tar.xz* file.

[7]Computed as the greatest musical occurrence in each artist's tag list.

[8]All users' encoding can be found in the *node_map1.csv* file.

[9]All artists' encoding can be found in the *artists_map1.csv* file.

[10]All music genres' encoding can be found in the *music_genres_map1.csv* file.

[11]Week id, present in the *weeks_map1.csv* file.