Inside The World's Playlist

Wouter Weerkamp Manos Tsagkias Maarten de Rijke
ISLA, University of Amsterdam
w.weerkamp, e.tsagkias, derijke@uva.nl

ABSTRACT

We describe Streamwatchr, a real-time system for analyzing the music listening behavior of people around the world. Streamwatchr collects music-related tweets, extracts artists and songs, and visualizes the results in three ways: (i) currently trending songs and artists, (ii) newly discovered songs, and (iii) popularity statistics per country and world-wide for both songs and artists.

Keywords

Music, stream processing

1. INTRODUCTION

Social media is changing the way we consume music. Online music services such as iTunes, Spotify, last.fm, and YouTube, enable us to access music from everywhere, anytime, and share our playlists with the world in the form of tweets, or status updates. People tweeting the tracks they are currently listening to generate more than half a million tweets per day. This offers us insights into people's music listening behavior at world scale. Historically, this type of research has been mostly based on surveys, or music charts [1], either of which is limited in scope or use, as it is often privately held. The most important drawback, though, we believe, is the data gathering process itself, which decouples what people listen to (or buy) from the context within which they do this this. Social media can complement this data, as it is mostly about people's activities [3, 7], with music being an important one [5].

There are two major challenges in mining social media for studying music listening behavior, except the sheer volume of incoming data: (i) how to identify music related content, and (ii) how to deal with the semi-structured, unedited nature of user generated content. For the first challenge, Hauger and Schedl [2] used three popular music hashtags on Twitter for identifying tweets potentially related to music: #iTunes, #nowplaying, and its shorthand, #np. We use the same set of hashtags, plus #spotify. Tweets tagged with #iTunes and #spotify are automatically generated by the respective software music players, while those tagged with #nowplaying are not associated with a particular source and may contain additional information to the track, e.g., lyrics, or experiences.[1],[2]

For the second challenge, we follow [2, 5] and use a set of regular expressions to generate a candidate set of artists and songs, which we curate using the Musicbrainz[3] database (an open music knowledge base). Our difference with previous work is that we use Youtube search for increasing the recall of our method, and develop tailored webpage extractors for the #iTunes and #spotify tagged tweets; we provide a comparison of these methods in the next section. Another dimension to this challenge is to identify the geolocation of a tweet. Schedl [4] uses the Yahoo! Placemaker API for this purpose, however, we find that the rate limits imposed by the service are not adequate for real-time use. We use Geonames,[4] an open geo database, for mapping a tweet's coordinates (or extract the twitterer's coordinates from their profile description, or location) to a geolocation.

In this demonstrator we present Streamwatchr: a real-time system for analyzing music listening behavior at world scale. Streamwatchr aims at (i) mapping unstructured, user generated content to structured data in real-time, and ii) providing up-to-date visualizations of what the world is listening to, what songs and artists are trending, and what will be the next big music hit. Streamwatchr can be accessed at: `http://streamwatchr.com`.

2. DATA AND BACKEND

Streamwatchr consists of a multi-stage approach for identifying songs and artists from tweets. The process is informed by observations on a training set of manually annotated tweets tagged with music-related hashtags. The assessment exercise revealed that the difficulty of correctly extracting the artist and song from a tweet ranges from very low to very high; see Table 1 for examples of both easy- and hard-to-extract tweets. Also, many tweets originate from radio stations which post their airplay. Streamwatchr ignores radio users by filtering usernames that match "radio," "fm," or "play" in them. Below, we describe our multi-stage approach.

Check hashtag. First, we check which hashtag is used to determine the next step. #spotify and #itunes tweets are transferred to the **Page extraction** stage, while #np and #nowplaying are moved to the **Baseline extraction** stage.

Page extraction. We follow the URL contained within the tweet and fetch the page. From this page we extract the artist and song, according to site specific regexes.

Baseline extraction. If a tweet matches certain basic regexes, we use these to extract candidate artist and song. These candidates are

[1]"Skip skip skip , emm haa . #np Hell above"
[2]"They say love is blind oh baby you so blind #np"
[3]`http://musicbrainz.org`
[4]`http://www.geonames.org`

Table 1: Examples of easy and hard to extract tweets.

Easy
#nowplaying Richard Marx - Angelia
#np funeral for a friend/love lies bleeding - dream theatre
#nowplaying Hey Soul Sister-Train
Hard (with answers)
They say love is blind oh baby you so blind. #np (G-Dragon - That xx)
When you feel my heat Look into my eyes It's where my demons hide It's where my demons hide Don't get too close It's dark inside #np (Imagine Dragons - Demons)
you got mud on your face you big disgrace somebody better put you back into your place #np #queen #wewillrockyou (Queen - We will rock you)

Table 2: Test results in Precision, Recall, and F-measure of three extraction methods and their combination (All) for songs (top) and artists (bottom). Recall is measured over the number of tweets offered to the method (e.g., Youtube only processes tweets that Baseline could not resolve).

Method	Tweets	P	R	F
Songs				
Baseline	29/200	**0.9655**	0.1400	0.2445
Youtube	96/171	0.7188	0.4035	0.5169
Fuzzy	41/75	0.3902	0.2133	0.2758
All	166/200	0.6807	**0.5650**	**0.6175**
Artists				
Baseline	29/200	**0.9655**	0.1400	0.2445
Youtube	96/171	0.7396	0.4152	0.5318
Fuzzy	41/75	0.5610	0.3067	0.3966
All	166/200	0.7349	**0.6100**	**0.6667**

then issued to MusicBrainz and if a match is found on the combination of artist and song, we store this result. If not, or if the regexes do not match anything, we continue to the next stage.

Youtube extraction. Based on the observation that people often refer to songs using lyrics, misspelled names, or other ways of "creative" writing, we need a large, music-related source with user-generated content. Youtube fits this description, as it is one of the main platforms for music (video) dissemination, in which each video comes with user-generated metadata (e.g., lyrics, comments, tags). We use the full tweet as query to Youtube, filter the results by category (Music), and retrieve the top 10 results. We then use regexes to extract candidate artists and songs from all results and rank these by weighted frequency (first result is more important than result 10). We move down the ranked list and try to match combinations of candidates (as song and artist) in MusicBrainz. In case a match is found, we store the result, otherwise we move to the final step.

Fuzzy extraction. In case we cannot find an exact match in MusicBrainz, we look for a matching artist in our candidate list from the **Youtube extraction** stage. Having found an artist, we use the other top candidate as song, even though we might not be able to match it in MusicBrainz.

The multi-stage approach ensures an efficient approach by starting with fast methods (**Page** and **Baseline**) and only moving to more expensive methods if the fast methods fail.

To develop and test our extraction pipeline we need a set of annotated tweets. To this end we manually annotate two sets of tweets, which contain the hashtags #np and #nowplaying. We ignore the other hashtags as they depend solely on the **Page extraction** step, which we is too basic to fail. From the two sets of tweets one is used as training and development set (consisting of 250 tweets) and the other set is used as test set (200 tweets). For all tweets we identify the song title and artists, including their MusicBrainz identifier. Tweets for which either the song, the artist, or both are missing are annotated with "UNK" for the particular field.

To test the various extraction methods we apply our complete pipeline to the tweets in the test set and record the extracted information and the method that is responsible for this information. So, if a tweet can be "solved" by the **Baseline extraction**, we record its solution and the fact that this method found this. Table 2 shows the results from the extraction methods. Note that these results, if a tweets is annotated with "UNK" and a method assigned a real band or song it is listed as an error.

The results show what we expected to find: the **Baseline** method performs extremely well on precision, missing only one of the 29 extracted song and artist pairs. At the same time, this method is only capable of processing 14.5% of all tweets (29 out of 200). Most previous work on Twitter and music (e.g., [2, 4, 5]) only report on using regular expressions on tweets to extract information. Our analysis suggests this leads to a significant loss in information.

For those tweets for which regular expressions fail, we apply our **Youtube**-based method. We find an increase of recall compared to the baseline, combined with a drop in precision. This method is capable of dealing with almost half of the total number of tweets, and more than half of the tweets offered to this method. The precision of this method remains fairly high, something which does not hold for the **Fuzzy** method. This method shows a substantial drop in precision, especially for song titles. However, because of the relatively high recall its F-measure is still higher than that of the **Baseline** method.

Finally, if we combine the methods into our complete pipeline (**All**), we obtain the highest recall and F-measure scores for both song title and artist name extraction.

3. INTERACTIONS AND VISUALIZATIONS

The extraction of song titles and artists is only a necessary step in our demonstrator to support four types of analyses. In this section we discuss four functions our demonstrator offers to gain insights in the listening behavior of people: charts, currently popular, discovery, geo analysis.

3.1 Charts

One of the obvious functions our demonstrator offers are charts. We collect statistics for both songs and individual artists on an hourly basis, which allows us to plot the number of plays on a fairly detailed level. Aggregating the data, we can show plots for any time period at any detail level (e.g., per month or per year). By plotting the data users can quickly identify trends in popularity or (un)expected peaks in listening behavior.

3.2 Currently popular

While the charts represent a somewhat old fashioned view of popularity, the *currently popular* function of our demonstrator tries to exploit the stream character of Twitter. As the tweets, and therefore the music, flows into Streamwatchr as a stream, we want to present the user with a real-time list of the most popular artists. A traditional popularity ranking would monitor the stream for a set period of time, counting plays of each song, and after this period report on the final ranking. Streamwatchr, however, uses a metric that rewards songs that are played often in a short period of time and punishes songs that were not played for a while.

More formally, Equation 1 represents our temporal popularity score, tp, for item x (song or artist) at time t:

$$tp_t(x) = \begin{cases} 1 & \text{if } tp_{t-1}(x) = 0, \\ tp_{t-1}(x) \cdot e^{-\beta \frac{\delta_{t,t-1}}{\theta}} + 1 & \text{otherwise.} \end{cases} \quad (1)$$

Here, β is a parameter indicating the "damping" factor (we set $\beta = 0.693$), $\delta_{t,t-1}$ represents the difference in seconds between this occurrence and the previous occurrence of x, and θ is the time unit, which we set to 60. From the equation we can see that if an item occurs for the first time, it is assigned score 1 and as soon as it is played again, we damp the current score. Items that occur frequently in a short period of time damp less and receive a higher score. The proposed method requires all items to be updated when a new occurrence enters the system. Given the amounts of data we are processing this is not feasible. We therefore use the equation to update the items that actually occur and use a second script, which runs every one minute, to update all items before publishing the popularity ranking.

The top of the resulting list of songs or artists represents what is currently popular on Twitter and this can be used as input to, for example, apps that allow users to play the currently trending music or to systems that inform radio stations and clubs about the popular music of this moment.

3.3 Music discovery

One of the hardest things for music lovers is to keep track of new music. Since we are monitoring music listening behavior continuously, we can quickly detect new music that is on the rise. By presenting newly discovered songs in Streamwatchr we offer a service to people looking for easy access to new music.

We employ a heuristic method to discover new music. First, a song needs to have at least 50 plays in one hour to be added to our list of candidate discoveries. We remove songs that have previously been discovered and finally check to see if the song was already played more than a week ago. The intuitions behind these decisions are the following: (i) For a song to be discovered it needs to have a certain level of attractiveness, represented by a substantial number of plays within an hour. (ii) Songs that were already played more than a week ago are not considered to be "new" and can therefore not be discovered. A typical pattern we find for songs that should be discovered by Streamwatchr shows a couple of spread out plays in the 2–3 days before discovery, followed by a sudden move upwards in the number of plays. We aim at presenting new songs to users of Streamwatchr right at the beginning of this move upwards.

3.4 Geo analysis

The final analysis that is facilitated by the demonstrator are location-based charts. When possible we extract the geo information from tweets using either the coordinates in the tweet or the user-provided location string in a user's profile. Although the percentage of geo-tagged tweets is very low ($\sim$1%), we believe that showing local charts can prove insightful for many users.

Using a map of the world we allow users to select the country for which they want to explore the charts. As mentioned before, we store plays for artists and songs on an hourly basis, which allows for detailed analysis. For future extensions we plan to implement comparison possibilities between countries, e.g., show popularity for an artist or song for two or more countries at the same time in a plot, or use different colors on the world map to indicate the date when a song became popular in particular countries. This allows for analyzing which countries play an important role in defining the world music scene.

4. FUTURE EXTENSIONS

Although Streamwatchr in its current form allows for various ways of analyzing and accessing music data, we envision three important and useful future extensions. First, to improve the level of insights we can gain from the data, we want to offer comparisons between either artists or between countries. In case of the former, we would like to offer the possibility to add multiple artists (or songs) to plots as to compare the popularity of two or more items over the same period of time. Similarly, we would also like to be able to add multiple countries to the plot of one song or artist, or to visualize the data on the world map using different shades of color (e.g., How long has a song been popular?).

A second extension is automatic peak detection and explanation. From observing plots we can see that certain artists have peaks in which many people listen to their songs. This raises an interesting question: why are all these people listening to this artist? To answer this question we want to implement a method that automatically detects peaks [6] and, more importantly, tries to automatically explain why this peak appears.

Finally, we want to look into contextualizing music. We are using Twitter and people do not only post *#np* tweets, but also other, more informative, messages. To what extent we can use these other tweets to contextualize reports on listening to music? What do people do just before or after listening to a song?

5. CONCLUSIONS

We have described Streamwatchr, which aims to give new insights into people's music listening behavior as reported on Twitter. Streamwatchr extracts song title and artist information from music-related tweets and presents it in a variety of ways: (i) traditional charts and (ii) location-based counterparts, (iii) real-time popularity rankings, and (iv) discovery of new music.

6. REFERENCES

[1] D. J. Hargreaves, D. Miell, and Raymond. *What Are Musical Identities, and Why Are They Important?* Oxford University Press, USA, 2002.

[2] D. Hauger and M. Schedl. Exploring Geospatial Music Listening Patterns in Microblog Data. In *AMR '12*, Copenhagen, Denmark, October 2012.

[3] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *WebKDD/SNA-KDD '07*, pages 56–65. ACM, 2007.

[4] M. Schedl. Leveraging Microblogs for Spatiotemporal Music Information Retrieval. In *ECIR '13*, March 2013.

[5] M. Schedl and D. Hauger. Mining Microblogs to Infer Music Artist Similarity and Cultural Listening Patterns. In *AdMIRe '12*, April 2012.

[6] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD '04*, pages 131–142. ACM, 2004.

[7] W. Weerkamp and M. de Rijke. Activity prediction: A twitter-based exploration. In *TAIA '12*, 2012.