TwitterBrainz

what music do people share on Twitter?

February 15, 2015

Open Data Project
Research and Innovation

Quim Llimona
Àngel Farguell

Modelling for Science and Engineering

Universitat Autònoma de Barcelona

Abstract

In this report we present TwitterBrainz, an interactive, real-time visualization of what songs are people posting on Twitter. Our system uses the MusicBrainz and AcousticBrainz APIs to identify the songs and extract acoustic metadata from them, which is displayed along with metadata from the tweet itself using a scatterplot metaphor for easy exploration. Clicking one of the dots/tweets/songs plays it back using the Tomahawk API, which can be used for reinventing our app into a music discovery tool.

Contents

1 Introduction

The aim of this project is to build a web-based interactive visualization using Open Data tools. Since we are both musicians, we thought it had to be about visualizing music, in any of its forms. We knew about the AcousticBrainz API, which provides acoustic information about commercial songs, and found in Twitter a great opportunity for integration.

The original idea was to find tweets from users that referenced the user was listening to a particular song, find metadata about that song, and then plot the metadata along with features extracted from the tweet itself, looking for correlations. For instance, one could expect that in work days people listen to more relaxed music than during the weekend. However, we found out on the way that such correlations were very rare to find – especially because of the real-time nature of the system.

At the same time, after adding a small player that enabled people to listen to the music from a specific tweet with a single click we realized the app had a great potential as a music discovery service. People could discover new music that is being listened to somewhere in the world (thus creating a kind of social bound) and that has some specific characteristics, such as conveying a positive message, or in minor key.

The system works as follows: first, we use the Twitter Realtime and Search APIs to search tweets containing specific keywords. Then, we extract the name of the song and the artist using regular expressions (regex) that match the text of the tweets. From there, we find all matching MusicBrainz recording IDs of the song using the MusicBrainz API. And with this ID, we find a musical features of the song in AcousticBrainz like tonality, rhythm, perceived relaxation, perceived positiveness, etc. We take tonality, tempo, relaxation index and positiveness, and with this information plus the tweet create, body, and posting date we create a real-time, interactive visualization around the scatterplot metaphor using HTML5. The Tomahawk Embedded Player is added to the mix to give the user the ability to play back the songs.

It has been live for a few days already, and it drew some attention on Twitter; we have summarized it on Storify [1].

1.1 Previous similar works

One of the reasons behind choosing AcousticBrainz is it was very recently created, therefore there are not many examples of people using it. One of them is the AB Jukebox [1], which provides a similar scatterplot visualization through which users can select what to play, although based on their local music collection.

On the other hand, the Billboard Music Charts portal provides a Twitter-based ranking [2], in which they track twits mentioning songs and provide statistics of the most popular, updated (as they claim) in real-time.

[1] `https://storify.com/lemonzi/twitterbrainz`

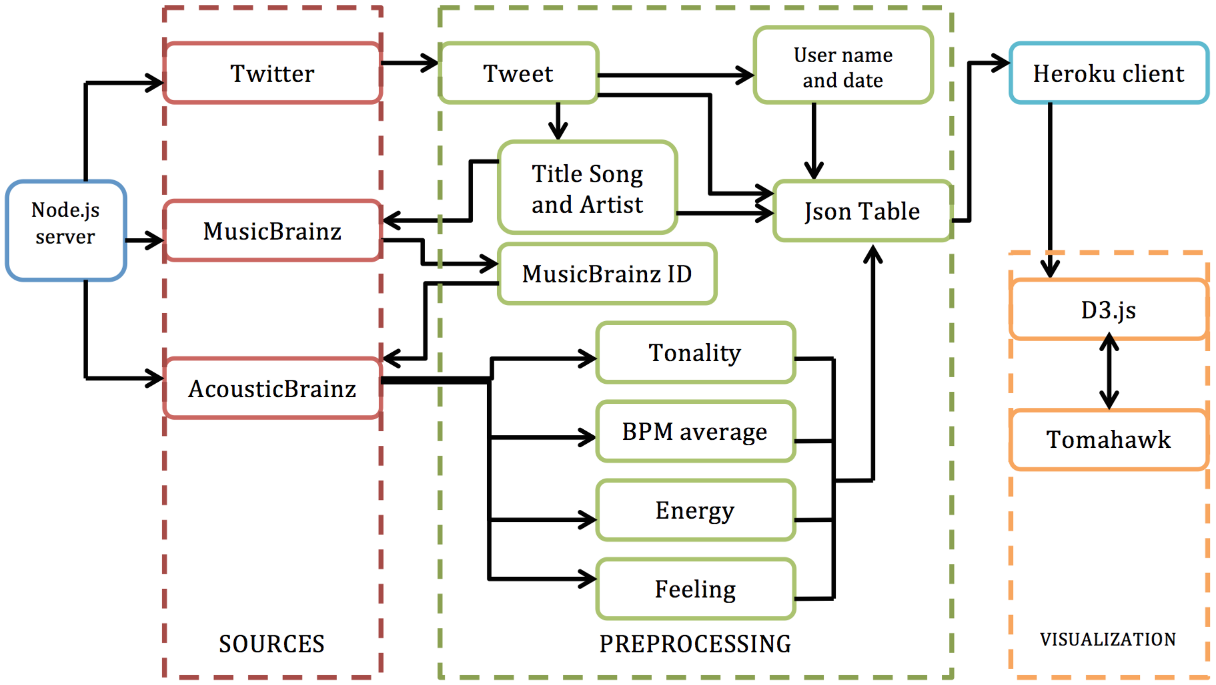


Figure 1: Representation of how our project works.

Finally, our visualization is based on one of the demos listed on the D3.js showroom [3], that fitted perfectly as a starting ground for our project.

2 Development

Our project is written entirely in Javascript; on the server-side, we run a node.js server using the free tier of the Heroku cloud platform; all code is available on GitHub [2], except our API keys. Data flow is divided in 3 principal phases: Sourcing, Pre-processing, and Visualization. All phases have an important task for the development of the project; Figure 1 shows how they are organized.

2.1 Sources

We are using 4 main data sources, all queried from our node.js server:

- **Twitter**
  A big social network where there are a lot of people that a person can follows another person. There are much people that post in Twitter that they are listening a song. Another kind of people use an API of music that posts in tweeter automatically. So, we have a good reference of music that appeals to people

[2]https://github.com/lemonzi/TwitterBrainz

nowadays. We put an option that if the tweets are in real-time or not. In order to take the tweets without real-time we use [3] and the real-time part with [4]. We used the twit javascript library [5] to make it easier.

- **MusicBrainz**
  MusicBrainz is a huge open database of recorded music metadata. They identify each recording with a unique MusicBrainz ID, which many other services recognize. We use it for bridging Twitter and AcousticBrainz, because it provides a Search endpoint that can suggest MusicBrainz IDs from textual data such as song or artist name.

- **AcousticBrainz**
  This database, created in a collaboration between Universitat Pompeu Fabra and MusicBrainz, is an open collection of acoustic features extracted from commercial recordings, such as tempo, key, genre, or mood. These features are crowd-sourced: anyone can extract them locally from a CD or mp3 recording and upload them. This project is very recent, so now the data base isn't very big but it's enough for our project. However, the quantity of songs in AcousticBrainz is growing up very fast. We can see how their data looks like in [6].

- **Tomahawk**
  Although Tomahawk appears at the end of the pipeline because it does not take part into the whole acquisition-processing-visualization process, we are using its API and it's worth mentioning. Tomahawk aims at providing a universal player where one can drag a link from any provider (such as Spotify or Rdio) or search a song by title or artist, and it will "just play" using whatever source is available at the moment, without requiring being subscribed to all available music services. They offer an API through which it is very easy to embed a player that can "just play" a given song plus artist name.

2.2 Pre-processing

The pre-processing part has 4 principal steps:

1. **Extraction:**

   Extract the title of the song and the artist from a tweet. This part needs a regex filter. Let's see it through an example:

   If we have this tweet:

   ```
   I'm listening to You Gotta Believe by Mary J. Blige using @doubleTwist
   https://t.co/xvLClqfCnC
   ```

[3] https://dev.twitter.com/rest/public/search
[4] https://dev.twitter.com/streaming/overview
[5] https://github.com/ttezel/twit
[6] http://acousticbrainz.org/data

and we have this regex filter:

```
/(?:I'm listening to )(.*)(?: by | - | - )(.*)(?: using)/i
```

we obtain finally this two elements:

`You Gotta Believe`

and

`Mary J. Blige`

So, we have the name of the song and the artist that the tweet contains. But, if we have this tweet:

```
Love Me Again [Gemini Remix] by John Newman, from #SoundHound
http://t.co/nZ8hgbUqz7
```

and we use the same regex filter, we can't match the tweet. Therefore, the filter doesn't return anything. So, we need another regex filter.

In this way, we can take the title of the song and the artist from all tweets and if a tweet doesn't match with one filter, then the program try with other. We did a file called *keywords.js* [7] that contain all engines and the regex filters that use each engine. We tested all filters with real tweets using the page [4] in order to improve them.

The tweets are fed in from both the Streaming API, that brings in tweets in real-time as they are published, and the Search API, that searches tweets published recently in the past. The user can select from the client-side whether to receive everything (realtime = no) or just the real-time tweets (realtime = yes).

2. **Matching:**

   With the title of the song and the artist, we find a match on the MusicBrainz song database using their Search API. Then, we look up the matches on AcousticBrainz looking for acoustic features for that recording ID. Notice that AcousticBrainz relies on people extracting features from songs and uploading them (crowd-sourcing), so the ratio of correctly matched tweets will likely grow over time. The MusicBrainz API has a very restrictive rate limit (one query per second), so we implemented a queue system to avoid overloading their server.

3. **Collecting:**

   We collect all information we want in a JSON object. We took the text of the tweet, the user name, the title of the song, the artist, the tonality of the piece, the bpm (beats per minute), the energy and the feeling (this last values are calculated by AcousticBrainz's algorithms and they are values from 0 to 1, that means from less to more energy and from bad to good feeling), and the tweet timestamp (divided into month, weekday, and hour).

[7]`https://github.com/lemonzi/TwitterBrainz/blob/master/keywords.js`

4. **Sending:**

   Then, using the `socket.io` library, we send all information from the node.js server to the client using Websockets. This is done in two modes: when a client connects, a dump of the recently matched tweets is sent to it, at a rate of 5 per second until it is up to date; at the same time, new incoming tweets are sent through another channel.

2.3 Visualization

A first, rough visualization of the data is provided through a console, built with JQuery Terminal [8], which logs all songs that have been correctly extracted by the regex engine.

Then, we did a D3.js visualization based on the visualization of the reference [5]. And we add a Tomahawk player in order to listen the songs only with one click on a circle (the circle of the tweet that we want to hear the song).

Now, We can choose the axis variables and the graphic changes, we have an option between show or hide the console and finally we add a bottom for clearing the display.

3 Product

3.1 Functionality

Our product is a website where we can see the tweets of the people that listened or are listening in this moment a song (they are the colored circles). If the color is blue it means that the song is in one of the major modes and in the contrary, if the color of the circle is orange it means that the song is in one of the minor modes.

We can change the variables in the x and y axis, corresponding to the features of the song or tweets being displayed. In the x axis, there are positiveness level and time measures like hour, weekday and month of the year that the tweet is published. In the y axis, there are energy level, tonality and tempo. The tempo is the average of the bpm (beats per minute) of the song.

There are 3 switches at the bottom of the axis filters: Realtime, Console and Clear. Realtime selects, as already mentioned, whether we receive tweets from both the realtime and search streams or just from realtime.

Console toggles an overlay with a terminal that shows the incoming tweets that are filtered and matched. We can see 3 different messages:

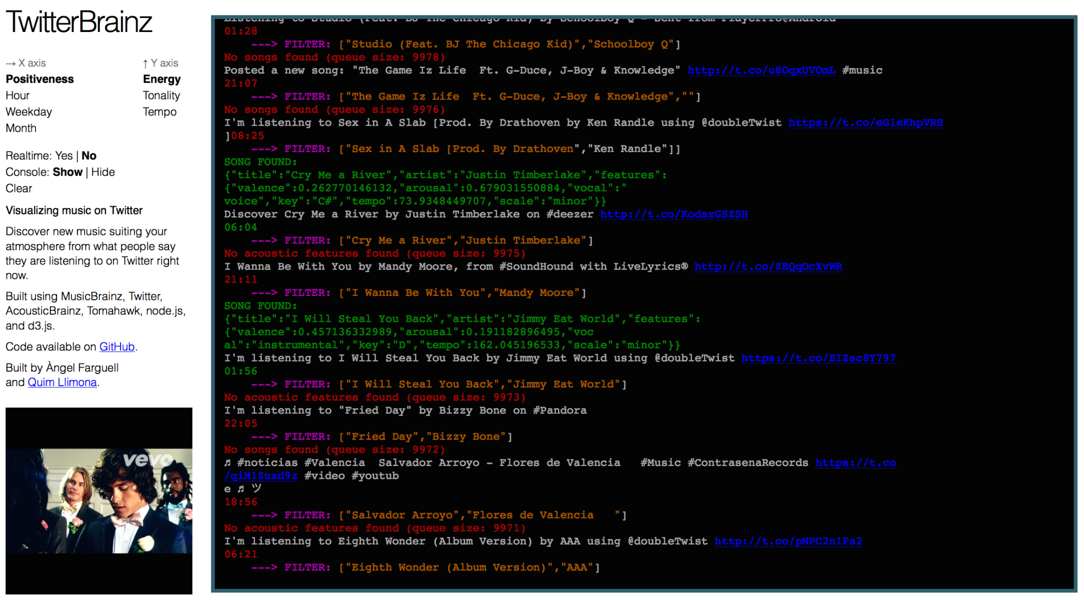
- **No songs found:** This message means that the song and the artist that we have filtered doesn't exist in MusicBrainz database. It has a number in parenthesis,

[8]`http://terminal.jcubic.pl`

which is the queue length of tweets that are waiting to be submitted to MusicBrainz for matching. In some cases, this message appears because the filter takes a very recent song not available yet; in others, the user misspelled something in the tweet or our filters didn't work well.

- **No acoustic features found:** In this case, there was a match on MusicBrainz, but AcousticBrainz didn't have data for any of the recordings that MusicBrainz had associated with the query. This can happen a lot, because AcousticBrainz is very recent and their database is quite limited.
- **SONG FOUND:** It means that the title of the song and the artist that we have filtered from a tweet is found in MusicBrainz and with its MusicBrainz ID, we have found its features.

We always print the tweet, the result of the filtering and the hour of the tweet (then we can see more about the hour of the tweet). In addition, if we have the last message, we print the title of the song, the artist and the features of the song, all from AcousticBrainz. We can see the console in Figure 2.



TwitterBrainz

→ X axis
Positiveness
Hour
Weekday
Month

↑ Y axis
Energy
Tonality
Tempo

Realtime: Yes | **No**
Console: **Show** | Hide
Clear

Visualizing music on Twitter

Discover new music suiting your atmosphere from what people say they are listening to on Twitter right now.

Built using MusicBrainz, Twitter, AcousticBrainz, Tomahawk, node.js, and d3.js.

Code available on GitHub.

Built by Àngel Farguell and Quim Llimona.

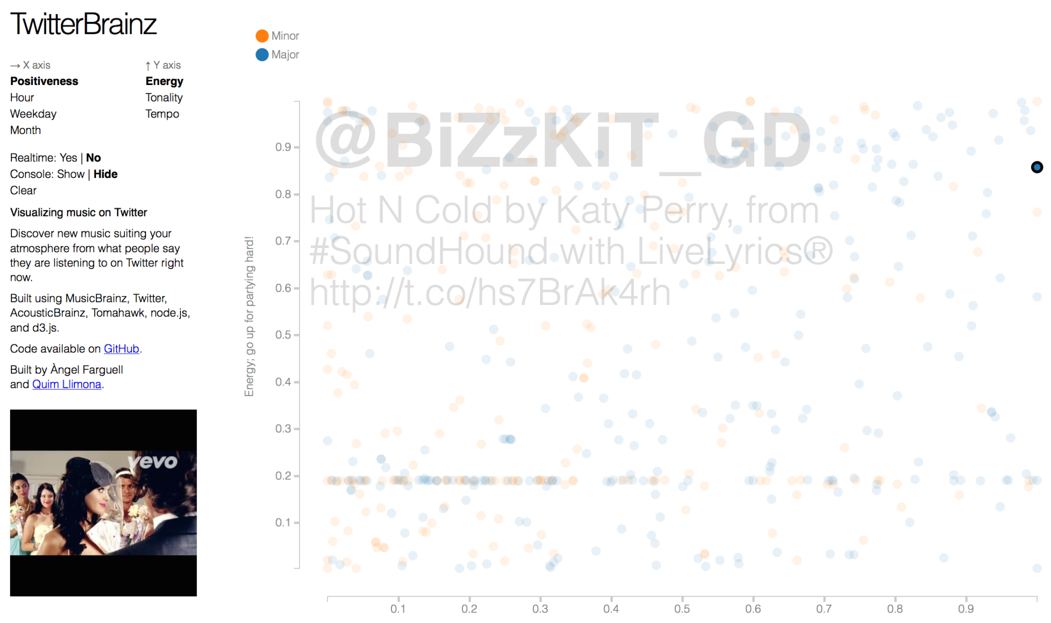
```
01:28
    ---> FILTER: ["Studio (Feat. BJ The Chicago Kid)","Schoolboy Q"]
No songs found (queue size: 9978)
Posted a new song: "The Game Iz Life  Ft. G-Duce, J-Boy & Knowledge" http://t.co/u8OqxUVOmL #music
21:07
    ---> FILTER: ["The Game Iz Life  Ft. G-Duce, J-Boy & Knowledge",""]
No songs found (queue size: 9976)
I'm listening to Sex in A Slab [Prod. By Drathoven by Ken Randle using @doubleTwist https://t.co/eGleKhpVRH
]08:25
    ---> FILTER: ["Sex in A Slab [Prod. By Drathoven","Ken Randle"]]
SONG FOUND:
{"title":"Cry Me a River","artist":"Justin Timberlake","features":
{"valence":0.262770146132,"arousal":0.679031550884,"vocal":"
voice","key":"C#","tempo":73.9348449707,"scale":"minor"}}
Discover Cry Me a River by Justin Timberlake on #deezer http://t.co/KodazGSZSH
06:04
    ---> FILTER: ["Cry Me a River","Justin Timberlake"]
No acoustic features found (queue size: 9975)
I Wanna Be With You by Mandy Moore, from #SoundHound with LiveLyrics® http://t.co/ZEQqOcXvWR
21:11
    ---> FILTER: ["I Wanna Be With You","Mandy Moore"]
SONG FOUND:
{"title":"I Will Steal You Back","artist":"Jimmy Eat World","features":
{"valence":0.457136332989,"arousal":0.191182896495,"voc
al":"instrumental","key":"D","tempo":162.045196533,"scale":"minor"}}
I'm listening to I Will Steal You Back by Jimmy Eat World using @doubleTwist https://t.co/EIZsc8Y797
01:56
    ---> FILTER: ["I Will Steal You Back","Jimmy Eat World"]
No acoustic features found (queue size: 9973)
I'm listening to "Fried Day" by Bizzy Bone on #Pandora
22:05
    ---> FILTER: ["Fried Day","Bizzy Bone"]
No songs found (queue size: 9972)
♬ #noticias #Valencia  Salvador Arroyo - Flores de Valencia   #Music #ContrasenaRecords https://t.co
/qiM1Zuxd9z #video #youtub
e ♬ ツ
18:56
    ---> FILTER: ["Salvador Arroyo","Flores de Valencia   "]
No acoustic features found (queue size: 9971)
I'm listening to Eighth Wonder (Album Version) by AAA using @doubleTwist http://t.co/pNPC2n1Pa2
06:21
    ---> FILTER: ["Eighth Wonder (Album Version)","AAA"]
```

Figure 2: The console in our website.

The Twitter API returns the timestamp in Coordinated Universal Time (UTC), so we take the timezone of the person listening the song in order to calculate the local hour where he is. We can see this calculus in the function *processTweet* on *twitter.js* [9].

Finally, we have a Clear command that clears the plot and tells the server to stop sending data from its history, receiving only data currently processed. This is useful for testing the real-time capabilities of the system, but it can take a while to fill the screen

[9] `https://github.com/lemonzi/TwitterBrainz/blob/master/twitter.js`



Visualizing music on Twitter

Discover new music suiting your atmosphere from what people say they are listening to on Twitter right now.

Built using MusicBrainz, Twitter, AcousticBrainz, Tomahawk, node.js, and d3.js.

Code available on GitHub.

Built by Àngel Farguell and Quim Llimona.

Hot N Cold by Katy Perry, from #SoundHound with LiveLyrics® http://t.co/hs7BrAk4rh

Figure 3: An example to better see the functionality.

with tweets without the help of tweets that were already found by the server before visiting the webpage.

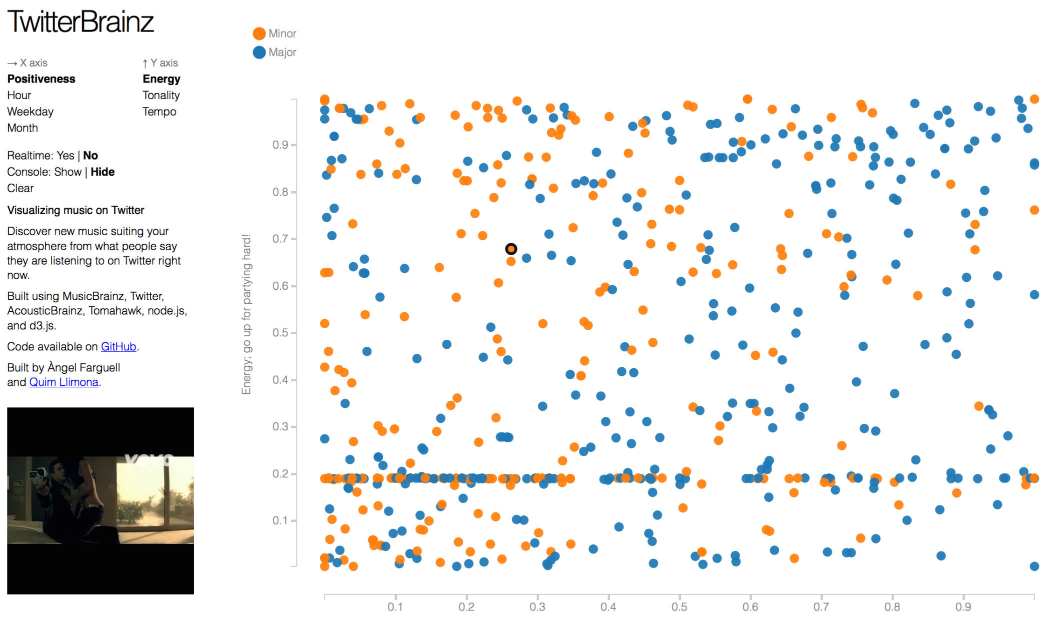
If we put the mouse on a circle, the user name and the tweet appears in grey. And if we click the circle, the song sounds immediately in a square in the left-down corner. We can see it in the figure 3.

Our project is available on `http://twitter-brainz.herokuapp.com` [6].

3.2 Limitations

The limitations are:

- The tweets normally aren't clear. So, it is very difficult do a filter. For this reason in a lot of cases, we don't find the song because the result of the regex filter isn't clear.
- Twitter isn't as open as the other providers. Therefore, if one day they close the API for public access we will have problems. And now, we already have limitations in the number of tweets that we can take per 15 minutes window. MusicBrainz also has limitations, but they could be overcome by creating a local replica of their database, freely available.
- AcousticBrainz is a recent database, so there are many songs it misses because nobody uploaded them yet to their server.


TwitterBrainz

→ X axis
Positiveness
Hour
Weekday
Month

↑ Y axis
Energy
Tonality
Tempo

Realtime: Yes | **No**
Console: Show | **Hide**
Clear

Visualizing music on Twitter

Discover new music suiting your atmosphere from what people say they are listening to on Twitter right now.

Built using MusicBrainz, Twitter, AcousticBrainz, Tomahawk, node.js, and d3.js.

Code available on GitHub.

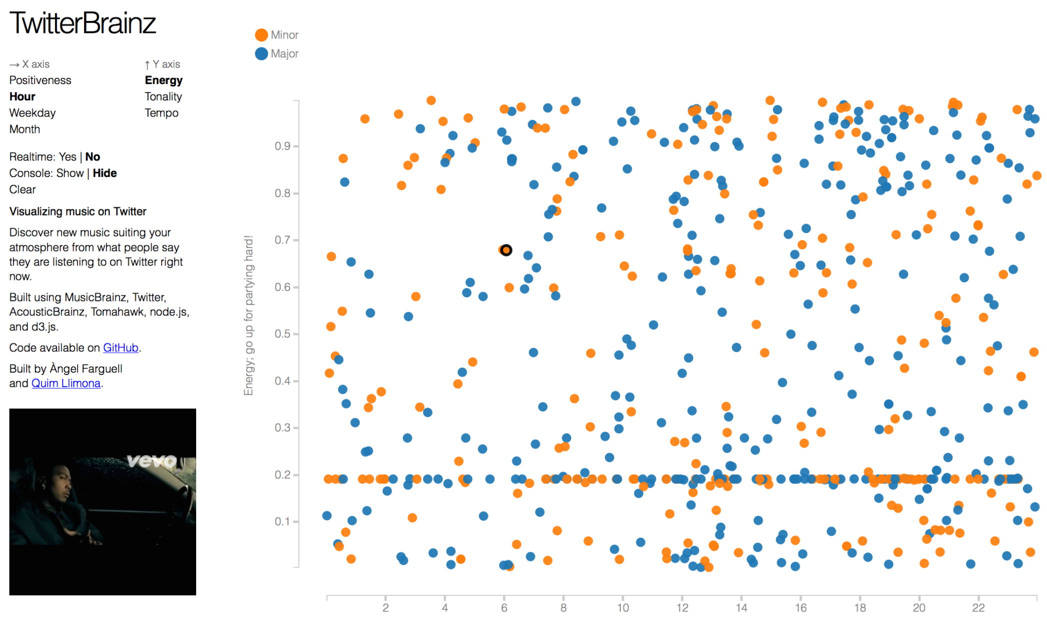
Built by Àngel Farguell and Quim Llimona.

Figure 4: Energy vs Positiveness

3.3 Example uses

We can take some conclusions with the results of the graphics. For instance, if we take attention in the Energy of the song:

- If we put Positiveness and Energy in the axis (Figure 4), we can see that if we imagine a diagonal line $y = x$, the superior part of this function have more minor songs and in the rest of the square area there are more major songs. Moreover, we can see a line in more or less Energy 0.2. So, there are a typically energetic value that is more common in the songs. And this energy is a low energy.
- If we put Hour and Energy in the axis (Figure 5), we can see that the people listen more songs at afternoon and night. This seems logical.



TwitterBrainz

→ X axis	↑ Y axis
Positiveness	**Energy**
Hour	Tonality
Weekday	Tempo
Month	

Realtime: Yes | **No**
Console: Show | **Hide**
Clear

Visualizing music on Twitter

Discover new music suiting your atmosphere from what people say they are listening to on Twitter right now.

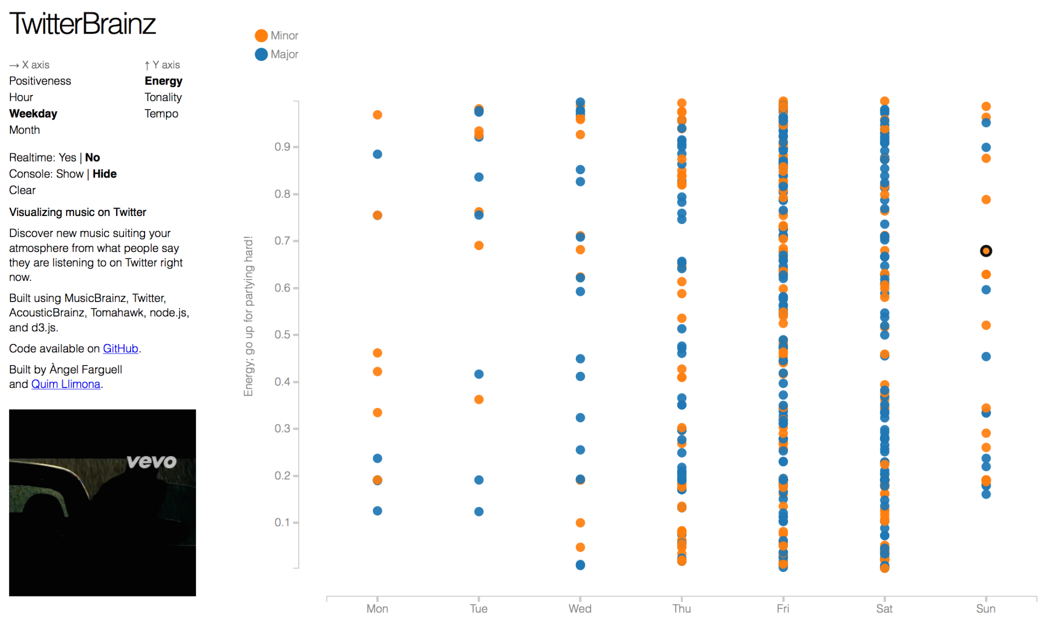
Built using MusicBrainz, Twitter, AcousticBrainz, Tomahawk, node.js, and d3.js.

Code available on GitHub.

Built by Àngel Farguell and Quim Llimona.

Figure 5: Energy vs Hour

- If we put Weekday and Energy in the axis (Figure 6), we can see that the people used to listen songs in Thursday, Friday and Saturday. The other days there are less tweets with song. It could be logical too, but this is probably biased by the fact that we took this snapshot near Saturday.



TwitterBrainz

→ X axis	↑ Y axis
Positiveness	**Energy**
Hour	Tonality
Weekday	Tempo
Month	

Realtime: Yes | **No**
Console: Show | **Hide**
Clear

Visualizing music on Twitter

Discover new music suiting your atmosphere from what people say they are listening to on Twitter right now.

Built using MusicBrainz, Twitter, AcousticBrainz, Tomahawk, node.js, and d3.js.

Code available on GitHub.

Built by Àngel Farguell and Quim Llimona.

Figure 6: Energy vs Weekday

4 Future improvements

During the development of this project, several ideas arose that we could not implement due to mainly time constraints:

- When the song being played through Tomahawk finishes, give an option to automatically play the song with the point closest to the one currently playing in the current feature space. This would work as a sort of radio, further easing music discovery, with the advantage of only playing music matching in a certain fashion to what the user has selected (in a certain key, or certain mood, etc).
- Package the extraction and matching steps as an independent library, so that people can write new apps or visualizations relating Twitter and music. Constructing the regex-based engine was not easy and we believe it is worth sharing.

Since the project is open-source [10], anyone can catch on from where we are leaving and suggest improvements to the app; we are very open to that.

References

[1] Ab jukebox ios app. `https://itunes.apple.com/us/app/ab-jukebox/id960022080?ls=1&mt=8`, 2015. Last accessed: 11-02-2015.

[2] Billboard twitter real-time charts faq. `http://www.billboard.com/billboard-twitter-chart-faq`, 2015. Last accessed: 11-02-2015.

[3] Peter Cook. What makes us happy? `http://charts.animateddata.co.uk/whatmakesushappy/`. Last accessed: 11-02-2015.

[4] Online regex tester and debugger. `http://regex101.com/`. Last accessed: 11-02-2015.

[5] Mike Bostock. Datadriven documents. `http://d3js.org`, 2013. Last accessed: 11-02-2015.

[6] Angel Farguell and Quim Llimona. Twitterbrainz. `http://twitter-brainz.herokuapp.com`, 2015. Last accessed: 11-02-2015.

[10] `https://github.com/lemonzi/TwitterBrainz`