ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

CS-422: Big Data

Professor Christoph Koch

Final Report:
Assisted Music Promotion Tool

Team members:

ABBADI Hajar
BENABDELJALIL Amine
BENBIHI Hind
CASTELLANI Mikaël
GIROUX Arthur
MATTER Valentin
NAOUS Dana
OUAAZKI Abdessalam

Supervised by:

Amir Shaikhha

20-05-2014

Introduction:

A critical step when you create something nowadays is the promotion. The Internet has become the best way to promote your product through social media or marketing campaign but it's very hard to know when, where and what to do for the promotion. Indeed, If you start your promotion too soon, people will forget your product if you don't keep the hype up. On the other end if you start your promotion too late or at the wrong time, your product might go unnoticed

The project goal is to help artists launch their music album in the best way possible. They will enter some information about their album like the launching date, the genre, the number of their Facebook followers etc, and the project will establish a timeline and tell them what to do. For example: Let's say that you want to launch a music album in 6 months. The project will tell you to a first album announcement on a given date, to tweet about the album cover at another date, etc.

Project organization

For each task we identified either at the beginning or all along the project, we split the team in sub-teams, to avoid leaving anyone alone. We tried to mix the sub-teams every time, and were allocating people based on both the competencies they had and the ones they were willing to get. Our schedule was given in a Gantt chart, defining at the same time the responsibilities of each student. All along the project, we always tried to have the tightest deadlines possible, to give ourselves some buffers to solve unexpected issues, or manage unexpected results. For example, our last biggest milestone was set from the beginning one month before the end of the project. On the other hand, a weekly meeting with our TA was also scheduled to make sure we would identify upcoming issues in advance.

The tasks were divided between the team members as follows:

Documentation on gathering data	
Twitter	Hajar + Mikael
Spotify	Arthur + Valentin
iTunes	Abdessalam + Amine
Facebook	Dana + Hind
Paper reports	
Detailed plan	Dana + Abdessalam + Amine
Data Gathering	
Tweets by account	Hajar + Mikael
Freebase data extraction	Dana + Hind
Facebook	Arthur + Valentin
One million song	Abdessalam + Amine
Finalize data gathering	Mikael + Arthur
Data storing	
Server + DB	Mikael + Valentin + arthur
NLP	Amine + Hajar + Dana + Arthur
Implementation	
Recommandation system	Abdessalam + Valentin + Hind + Mikael
Research	
Implementation	
Web frontend	Valentin
Enter basic data	
Visualize timeline	
Result refinement	
NLP	Hajar + Arthur + Dana
Recommendation system	Valentin
Clustering	Amine + Valentin
Report	
Final presentation	Michaël + Abdessalam + Hind
Report	Everyone

Architecture

The final system is composed of several parts that are described below. Since we wanted to keep everything in one place, every program we used/built ran on *icdatasrv4*. We also configured Git to be able to easily get the latest updates of our programs on the server, directly from our Github repository.

Database:

For storing our data, we chose to use MongoDB mainly because it's scalable, flexible and also because it made us learn something new. As scalability is important in a project related to big data, we didn't have to use multiple MongoDB servers but if, at some point, the database would have been too slow, it could easily have been scaled. Since MongoDB is document based, we were able to add fields as we needed without having to endlessly change a database schema. MongoDB would have been a bad choice if you needed joins but that wasn't a requirement for what we built.

Front-end

The front-end is entirely built in HTML, CSS and Javascript. When the client receives the web page, his recommendation request is sent by his browser (using AJAX) to a small Java web server (that runs on port 8000 of icdatasrv4) that parses the request, launches an instance of the recommender and send the result back in JSON to the client which displays the timeline. The choice of using a small Java web server for handling the recommendation requests came from the fact that we needed to find a way to make our recommender communicate with the outside world and since our recommender was in Java, it was the simplest way.

Data Gathering

To be able to reach our goal through analysis of the data, we needed to identify at first what data were necessary and where we could get them from.

1) Identification of the different sources

As our tool aimed at providing insight on social medias, we knew we had to parse them. We didn't need to parse all publication in the world though, but just communication made by artists in order to promote their albums. We decided then to only get the publications (and comments from the public) made through their accounts, on Twitter and Facebook, as they are the most used ones and as we had chosen those two as an output of our system.

In order to get data published by each artist on social medias, we needed to have their account information. We started to look where we could get such information, and found the Freebase from Google, a community fully accessible database, containing more than 100Gb of data on all

kind of subjects. We didn't need everything and therefore used their API to query the data we needed and fill our own database.

As we couldn't assess the positive or negative impact of a marketing campaign (the basic question : "How can we separate the success of the album brought by marketing compared to the quality of the album in itself ?" cannot be answered), one of the first assumptions we made was that the popularity of an artists is strictly tied to its popularity on social medias. Therefore, a popular artist should be part of our training set. But as Freebase didn't have any information on the popularity of artists, we used the website EchoNest, a renowned company in the music domain, to provide us trend on the artists.

2) Data Parsing

With those four data sources, Echo Nest to provide us trendy artists, Freebase for the data on artists and albums, Twitter and Facebook for the social medias, we began to design our data parsers and then integrated them in a chained process. All our parsers were programmed in Java and were storing the intermediate results in our MongoDB database. As we knew Mongodb isn't designed to handle concurrent transactions, we ran the four jobs sequentially and not simultaneously. Because of rate limits, for example the most restrictive 150 queries per 15min of Twitter API, we requested several developer keys (8 in total) and designed our bots to change the keys when the rate limit was reached.

3) Final data and limitations

Within a few days of parsing, we reached the amount of 6k artists having a facebook or a twitter account, 20k albums, 7M tweets and 300k Facebook publications.

During our data gathering, we already encountered several limitations inferred by the current context:

- The usage of Facebook and Twitter for marketing purposes is quite recent. We therefore only considered album promotions from year 2010, which reduces the size of our dataset.
- For privacy reasons, Twitter only gives access to 3500 tweets for each user. For some very active artists, this allows us to get only one or two years of tweets, which decreases again the exploitable size of our dataset.
- Some artists (around 10%) didn't have a Facebook and a Twitter account.

4) Conclusion

Data gathering took around 30% of our time on the project. We must not forget that this operation is really important and has to be executed carefully as it provides the data we will build our model on. Thanks to this work, we learnt that after each session of parsing, one must always challenge the data to try to find inconsistencies to then refine the parsers. For example this is how we decided to remove Spotify from our sources, or how we refined our parsers several times before reaching a good quality of data.

Natural Language Processing (NLP)

In this project, NLP is used in two phases for processing the social networks data available. It is first used in the event extraction step, and following that a sentiment analysis of user comments is done to assign scores (or weights) for the given events necessary for the recommendation process.

Events Extraction

From the Tweets and Facebook posts we collected, we needed to determine the ones that were relevant to our project, i.e. the posts relevant to promotional events. This list of events is as follows: single release, CD release, press campaign, presale campaign, first tweet, first Facebook post, countdown, announcement, album cover, interview, video clip and teaser. Furthermore, we needed to keep only the posts about events of the artist, and get rid of events of artists different than the page owner.

We first determined a list of events that we wanted to have in our promotional campaign. This list was obtained from research about music album promotional campaigns. After determining the list of keywords relevant to those events, we used a stemmer to get the stems of the list of events and we used those stems to help us determine the relevance of the Tweets and Facebook posts.

We used Part-of-Speech Tagging to help us extract events that are relevant to the artist. Part-of-Speech tagging marks up words in a text that correspond to a particular part of speech. We used the Maxent Tagger from the Stanford NLP library to tag the words and identify the cases where possessives are used. The tags were used to determine whether the posts or tweets about a certain event was really about the artist, or was a post or tweet about a different artist. Therefore, we specified rules about the possessive case to check if the artist was referring to himself/herself or not. The rules we set up for the possessive cases are as follows:

- In case of the occurrence of a possessive s ('s) , we check if the word right before is a noun by checking the POS tag. If it is a noun, we check if it is the last word of the artist's name or the artist's album name, or if occurs in our list of stems. If so, we check the noun right after and if it occurs in our stem list or if it corresponds to the first word of the album name, we keep the message. (example: Coldplay's album release).
- In the case of possessive pronouns, if the pronoun is "my" or "our", we check if the next noun is our stem list or if it is the first word of the album name. If yes, we keep the message. (example: My latest album is finally available). If the possessive pronoun is different than "my" and "our", we retrieve the previous noun and we check if it contains a stem or the last word of the album name. If yes, we keep the message (example: Coldplay released their album).
- If the message doesn't contain possessives, we keep it.

The cases that are different from the ones listed are discarded.

Sentiment Analysis:

The approach used for the sentiment analysis of comments uses sentence level analysis and a sentiment dictionary, to assign scores of positivity and negativity of words extracted from each sentence. Stanford POS tagger[1] is used to tag sentences in order to extract descriptive words that can specify the score of the comment.

The Stanford NLP library [2] is also used to determine the sentiment scores, 0 is negativity, 2 is neutral and 4 is objectivity for a sentence to define its sentiment thus allowing the classification

of comments as positive or negative oriented. Moreover, the extensive use of emoticons in social networks requires considering these for the scoring process as well. A list of common emoticons with predefined scores is used for this purpose.

The scoring of each comment is used in a Map-Reduce algorithm to define the weights of events for recommendation. For each Facebook event we emit for each of the comments the tuple (event_id, comments).
In the mapper we map each tuple to the sentiment analysis score associated to this comment.
In the reducer we aggregate all the sentiments for each comment into one score between 0 and 1 where 0 = negativity, 0.5 = neutral and 1 = positivity.

Recommendation System:

Our system aims at predicting a timeline of the promotion of an album to an artist. To do so, we used a content-based recommendation system adjusted to our specifications and needs.
Users are artists about whom we gathered information from the different data sources as described in the previous sections, whereas items are the set of events that constitute the promotion campaign.

Each artist has a profile that consists of: region, genre, number of albums released, number of Facebook followers and number of Twitter followers. Then each item is described with a set of features that indicates the number of days needed for each event to occur with respect to the release date. The events are : single release, CD release, press campaign, presale campaign, first tweet, first Facebook post, countdown, announcement, album cover, interview, video clip and teaser.

An artist using our system to predict a promotion timeline is required to fill in the elements that make a user profile as described previously. Based on the profile, our system computes the similarity with stored artists' profiles to get a set of item features of the similar artists. Next, the similar item features are processed to give the user a final promotion timeline that will most likely result in a successful promotion campaign given our assumptions.

The recommendation matrix is hence made of users' profiles - artists profiles- and items' features - albums events- as shown in the figure below.

Part 1: Artist profile

Region	Genre	# of facebook followers	# of twitter followers	# of albums

Part 2: Album features

single release	CD release	press-camp-aign	presa-le camp-aign	1st Twitter post	1st FB post	count-down	announc-ement	album cover	interview	video clip	teaser

Figure 1: Recommendation matrix structure

1. Similarity computation between users profiles:

Entries in the user profile are boolean to simplify our similarity computation.

The similarity is computed using the 5 attributes that constitute an artist profile. A clustering is performed on 3 attributes: number of albums, number of facebook followers and number of twitter followers, so as to determine ranges of those attributes values to be used in the recommendation matrix. Then, a cosine similarity is calculated for each attribute between the artist that is using our system and one artist from our database. The final value of the similarity between the two artists is obtained by performing a weighted sum using the weights assigned to each attribute.

This similarity computation is calculated iteratively between the artist- the user of the system- and all the artists from our database. This step gives intermediate results containing a set of artists with a similarity value greater than the threshold we determined.

Clustering

As we discussed earlier, in order to perform the cosine similarity in the recommender system, we needed to group the artists together and make sure the new artist belongs to one of the groups. For this matter, we decided to create clusters of artists depending on the 3 attributes "facebook likes", "twitter followers" and "number of albums".

The algorithm we used, which uses the three attributes as points coordinates for clustering, is a modified version of K-Means called "K-Means++" (proposed by David Arthur and Sergei Vassilvitskii in 2007) and available for use in an Apache Library. Like the normal K-Means, we have to specify the number of clusters.

After we generate clusters, we identify them with their centers. We later assign to each artist -and hence, album- in our database the center of the matrix it belongs to.

Thereafter, in the recommender, we assign the user to one of the clusters by choosing the closest cluster to it. In other words, we compute the distances between the user's provided attributes and the center of the clusters and choose the cluster that minimizes that distance (min(d(X,centeri))).

2. Aggregation of items features:

The items corresponding to the resulting set of artists from similarity computation contain different values for the features. The purpose of this step is to aggregate the results so that they converge toward a single value that would be the output of the recommender system. This is performed by using a result of processing social media data described in previously.

References:

[1] Stanford POS Tagger [online] Available: http://nlp.stanford.edu/software/tagger.shtml

[2] Stanford NLP Sentiment analysis: http://nlp.stanford.edu/sentiment/