Success Prediction on Music Within a Community

Jason Lo

A10836922

jalo@ucsd.edu

Abstract

There are countless posts on Reddit's default music subreddit made every day, but only few ever are noticed by members of the community, and even fewer make it to the front page, a sign of popularity in the community. We will try to find a correlation between the type of music posted and how well the community receives it. We're able to find loose correlations between genres and popularity, but overall it appears that novelty and familiarity trump any specific taste.

I. Data

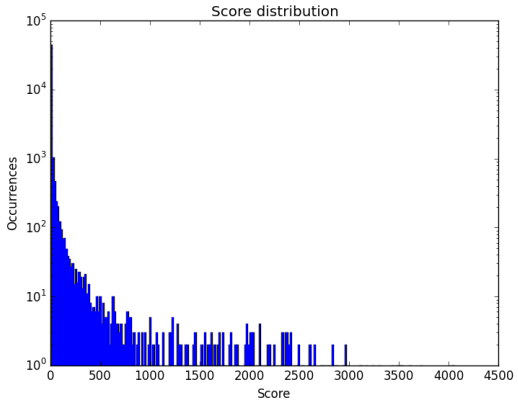
In the online community Reddit there is a popular subcommunity, "/r/music", where members post songs that they enjoy. Others in the community then vote on the posts, and ones that are highly rated are given more exposure, while submissions that are lowly rated quickly are filtered from view. In this study we will analyze what makes a post successful in this particular community by studying prior submissions.

The dataset for this study is roughly 47,000 posts from this community sampled over the last two years. This dataset was then paired with musical genre information from Last.fm music database. The collection of Reddit post data as well as genre information from Last.fm was done specifically for this study. The data entries take the following form:

```
{"score": 245,
 "artist": "st. vincent",
 "date": 1433031894.0,
 "song": "surgeon",
 "tags": ["female vocalists",
          "singer-songwriter",
          "indie pop"]}
```

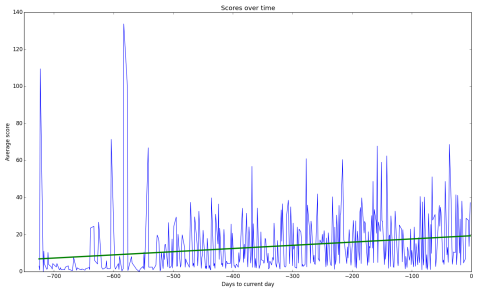
Unfortunately the data collection relied on a standard format for the submission titles. Although most of the posts in the subcommunity followed this standard, some didn't, which leads to some noise in the data, such as artists being misidentified. Also, due to the collaborative nature of Last.fm's data, not all of the entries have complete tag data, with some having no tagged data whatsoever.

Below is the score distribution across all entries in the data set.

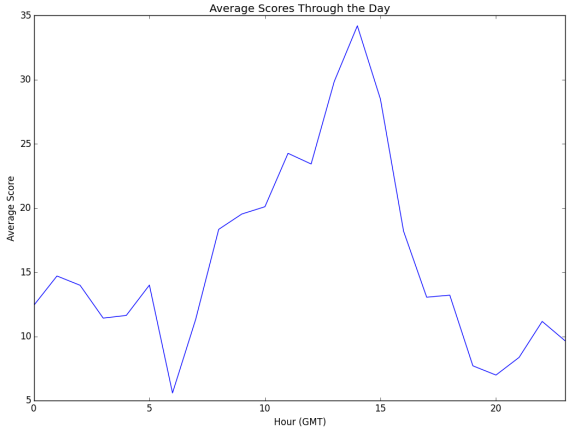


Even with the y-axis being on a log-scale, it is clear that most posts receive extremely low scores. Looking further at the data shows that more than half of the posts in the data set have scores of 0 or 1, with one being the default score for a submission. Eric Gilbert describes this behavior as underprovision, where most of Reddit's users only use the "Popular" filters, leaving very few users in control of the "New"

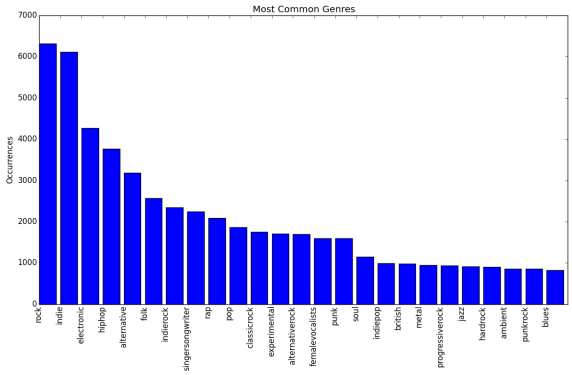
filtered posts and letting many of them go unnoticed (Gilbert). This is going to be a problem in analyzing the data as many of these posts convey little information as to the preferences of the community, but instead only convey the high incoming posts to attentive user ratio.

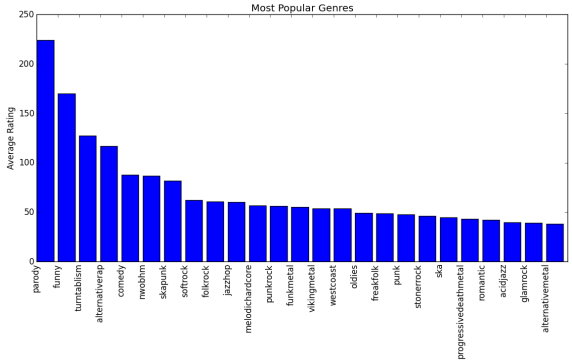


The average score over the last two years hasn't greatly changed, so hopefully we can ignore any community shifts over the time period.



The average score does shift by a greater amount over the course of an average day though, possibly due to the time zones that the majority of the community participate in. We're going to want to consider this in our model.





The community definitely has certain tastes, as some genres completely dominate the total pool submissions, while others dominate the most highly voted on average. Due to being averages though, the popular genres graph is a bit misleading—most of the genres are rarely submitted. In fact, this graph only shows popular genres that have been submitted at least 25 times, as there are quite a few more genres with higher average ratings only due to music of those genres being submitted no more than 3 times over the entire data set.

Also, genres in the most popular pool are generally more specific than those in the most common pool, with some of them being specific variants of the more general forms (e.g. "punk" going to "melodic hardcore", or "hiphop" going to "turntablism"). This also means that the more general genres in the common pool are going to be covering more artists, and therefore be less helpful than the more

specific genres.

II. Model

We want to find a way to predict the success of the post, some function of the score, given the musical information of the post. The bottom-heaviness of the data doesn't seem to lend itself well to regression, as less than a third of the submissions ever get beyond 2 points. Posts that one might consider to be successful (i.e. scores above 100) are very few, almost to the point of being statistical anomalies.

To deal with this, I've decided to lower the scope from "popular" to "noticed", where being "noticed" is arbitrarily being defined as having 5 or more points. Even a post with only 5 points is in the top 15% of the community.

With this, this become a classification task, predicting whether or not a post will reach that point of being "noticed". To do this, we have at hand:

- The artist of the song
- The name of the song
- The top three genre tags as provided by Last.fm
- The date and time the song was posted

We're going to be using logistic regression as our classification algorithm of choice. Logistic regression typically performs well when the two sides being classified are as far away from the decision boundary as possible, so we're hoping that can work with the huge pool of data entries that sit at 0 and 1 on the score scale.

III. Research

The data specifically for this study was collected through the Reddit API[1] for posts from the "/r/music" subcommunity and the Last.fm API[2] for artist genre information. Reddit data in general has been studied in the past in figuring out the "success" of a post, albeit using more general parameters such as titles and community makeup (Lakkaraju).

Gilbert's analysis in the typical use patterns on Reddit (Gilbert) as well as the properties of heavy-tailed distributions (Beirlant) are what lead me to transform this into a classification problem, as well as apply a modified model that will be discussed below.

IV. Results

I. Preprocessing

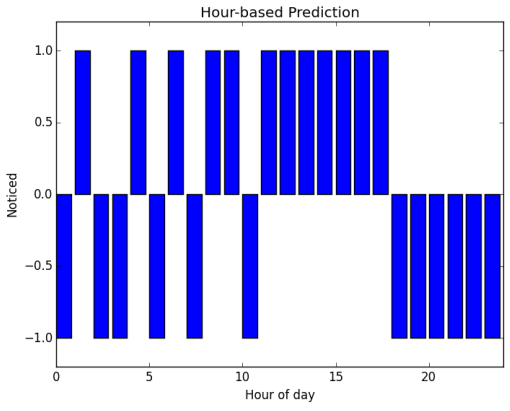
In an effort to further decrease the inherent bias of our data set towards non-popularity, we will be operating over a subset of our data that is constructed such that there is an equal number of "noticed" and "non-noticed" posts in our data.

We'll also be splitting our data, letting 4/5ths of it lie in the training set and the remaining 1/5th be for validation.

II. Time Baseline

To start things off, let's first create a one-hot vector representing the hour a post was submitted, and run logistic regression on that to build a predictor for "being noticed".

With just this, we get a classification error of 0.4596 on the training set and 0.4742 on the validation set. As the data is roughly evenly split, we're currently doing just a little better than just predicting yes or no all of the time.



[1] https://www.reddit.com/dev/api
[2] http://www.last.fm/api

Here are the hourly predictions with this baseline classifier. They roughly seem to correspond to the real-valued hourly prediction graph above, especially with the pocket of positives in the middle-right portion of the day.

III. Artist Data

Using artist names (or song title for that matter) is potentially risky because unlike the crowd-sourced and crowd-verified genre tags from Last.fm, the artist name is purely provided by the single author of the post and is therefore is vulnerable to spelling mistakes or even outright mislabeling. Also, as the "/r/music" community is mostly YouTube-based, one-off artists that produce popular YouTube videos might be posted to the subreddit. These sorts of posts aren't useful if we're going off of artist names as they will often never show up more than once.

To mitigate user input noise, we'll first be roughly "sanitizing" all artist names, essentially removing all spaces, casing, and punctuation, therefore making sure "St. Vincent", "st.vincent", and "st vincent" are all the same artist.

We'll then construct features similar to the genre pools mentioned when analyzing the data. We'll have one pool that'll contain the most-posted artists, and another that contains the most noticed artists *that have appeared at least twice*. To utilize these in our feature vector, we'll first just add two features that correspond to participation in these sets.

Through testing on our validation set, I've found that roughly 400 artists in our most popular pool and roughly 150 in our most highly-rated pool gives the best results, those being 0.3394 classification error on the training set, and 0.3566 on the validation set.

The coefficients for our classifier that correspond to the two artist features are overwhelmingly positive to the point where any entry that participates in either of the two above pools is classified as being noticed.

IV. Tag Participation

We will now incorporate the song tags along with the hours and artist name features. The song tags are safer when it comes to misspellings, but they can suffer from representation problems (some tags encompass too many songs and therefore convey little information when differentiating songs) or even just identification problems, as only about 70% of the entries have a full set of three tags. Last.fm also has idiosyncrasies when it comes to tags, such as marking "hip hop" and "hip-hop" as separate categories, although we can and will deal with things like that with the sanitation process outlined above.

We will construct our tag features similarly to how we handled the artist features above. We'll create a set of most-used tags and another of most-noticed tags, both limiting to amounts designated through testing (250 and 100 respectively). Then we'll add three features for each pool, each corresponding to whether the nth tag in the item is participating in the most-used or most-noticed group. As some entries have less than three tags, we'll just fill the *right* side with with zeros as appropriate.

The reason we append to the right end is that Last.fm assigns tags by group consensus, with the firstmost tag being the most agreed upon as describing the artist correctly (if albeit too generally as we've demonstrated). Therefore if we have less than three tags, in order to keep the meanings of the weights consistent, we pad the right hand side with nonexistent tags that would be worth less anyway.

With this, we get classification errors of 0.3198 on the training set and 0.3357. An improvement on just artist name, but not by much. This is probably due to tags being associated with *artists* rather than the independent songs. This means that the information that they relay is extremely highly correlated with the artist. Tags only really help when the artist isn't in the the top artists pool, which may explain the slgiht

References

[Gilbert, 2009] Gilbert, Eric (2009). "Widespread Underprovision on Reddit".

[Lakkaraju, 2013] Lakkaraju, Himabindu and McAuley, Julian and Leskovec, Jure (2009). "Whatś in a name? Understanding the Interplay between Titles, Content, and Communities in Social Media".

[Eirlant, 2001] Eirlant, J.B., Atthys, G.M and Ierckk, G.D (2001). "HEAVY-TAILED DISTRIBUTIONS AND RATING".