CAARS: A Context-Aware Artist Recommender System for Twitter Users

Abstract

In this work, we introduce a context-aware hybrid artist recommender system that uses Twitter users' tweet-time patterns as context and users' bias about gender and types of musicians to recommend artists. Our model offers a novel approach to improve a personalized music recommender system as it extracts implicit information from the users' past tweet-behavior and combines that with related content. The proposed model performs significantly better than collaborative and hybrid recommender systems and encourages further exploration.

1 Introduction

How does a recommender system know you well enough that it helps you select the exact items in minimal time? How will it keep the list interesting and not repetitive? Researchers are investigating these basic questions by looking into this potential research area, Analysis of Recommender Systems, from different domains to offer a reliable model for achieving user satisfaction. Every system is significantly different than others; the features and aspects to be considered in various domains may be dissimilar. For example, the *Goodreads* recommender system will not be looking into the same features as the *Amazon*'s book recommender system though their common goal is to recommend books of interests to their users. Again, a movie recommender system is not a well suited for song recommendation. With the abundance of datasets and domains, we propose a recommender system for musical performers. Our goal is to personalize the system by looking into the users' preferences on some content and expand the horizon by considering dominant contexts; hence we call it the *Context-Aware Artist Recommender System (CAARS)*.

In building *CAARS*, we use Million Musical Tweets (Hauger et al. 2013) and MusicBrainz (Metabrainz 2015) datasets. We consider the available information about the listeners and their tweet-histories to find some association with the musicians and the types of music. We then use context information (users' tweet-time pattern) and personalize the contents (their biases about the gender of the artist and the type of the artist) to refine the recommendation list. This composite system helps identify users who are similar in terms of the same tweet-time patterns and produces interesting results. To the best of our knowledge, we are the first group to offer such a model using these datasets.

Our model combines the strengths of both of the traditional recommender system approaches: collaborative filtering (CF) and content-based (CB) filtering. In CF, people collaborate to help each other perform filtering by recording their reactions to items that they use (Goldberg et al. 1992). CF recommender systems record users' preferences as ratings (numerical values). The more ratings the system can draw out from the users, the more effective the recommendations are (Elahi, Ricci, and Rubens 2016); which directs us towards the cold-start problem, the major drawback of a basic CF approach. In CB recommender systems, the descriptions of the attributes of items are used to make recommendations. CB methods are effective at providing recommendations for new items (Aggarwal 2016), mitigating the cold-start problem. CB methods have different trade-offs: they are not effective at providing recommendations for new users and sometimes good descriptions of the item-features are not available to the system. On the other hand, the incorporation of contextual information about the user in the recommendation process has recently attracted major interest (Verbert et al. 2012). Considering all these, we tune context with a combination of CB and CF approaches in our artist recommender system.

The paper is organized as follows: In section 2, we discuss the existing models and their limitations and introduce the basic differences between implicit and explicit feedback. In section 3 we describe the datasets and the features used in our model, in section 4 we present our hypothesis, in section 5 we describe our model in detail and analyze the performance in section 6. Finally, in section 7, we discuss the scope of this work and how this can be expanded in future.

2 Related Work

We have built on existing research in four areas:

- Music Recommender Systems
- Hybrid Systems

- Context Aware Recommender Systems
- Implicit User feedback

2.1 Music Recommender Systems (MRS)

With the aid of technology, personal music collection have grown dramatically in the last decade. But the idea of automatically recommending music genuinely did not begin until 2001 (Celma 2010). As an evidence of people's interest in this domain, we notice a consistent raise in the number of research articles related to music recommendation in the International Society for Music Information Retrieval (ISMIR)[1] every year; this is a clear indication of the usefulness and impact of music recommender systems. But sales record of the American music industry[2] shows that music consumption is biased towards a few popular artists. The scenario might be similar for the rest of the world as well. We need a remedy to this situation to better support the whole music industry.

Systems designed for movie or book recommendations cannot readily be applied to music recommendation: the lyrics of a song are usually shorter (and more incisive) than a movie description or a novel, the genres are quite numerous, the same song sung by two different artists might not be equally appealing to a listener, and so on. Sometimes a user listens to the same song several times, which likely indicates a strong preference for it. But this kind of behavior is rare in case of movies or books. Also, context (weather, location, user's current mood or activity) plays a dominant role on users' listening patterns. Users song preferences are more often tracked implicitly than by asking for explicit ratings. Hence, MRSs usually favor hybrid and context-aware approaches.

2.2 Hybrid Systems (HY)

CF approaches usually rely on users' numerical ratings, and more importantly, do not work for from-scratch startups/businesses without a good store of data about users and their preferences. These techniques are inherently domain-agnostic, and can be easily applied to explicit music rating data (Shardanand and Maes 1995). On the other hand, CB systems use the feature descriptions of items previously consumed by a user to suggest items with similar content features. This process is time-consuming, expensive, and also suffers from the user-end cold-start issue. The CB systems need well-defined content information to work reasonably well. Besides the cold-start problem, CB and CF, both suffer from the diversity issue. As the traditional systems only consider similar users or similar items, users have fewer chances to be exposed to new items of different types that may become new favorites.

Hybrid RS systems have been designed to overcome the limitations stated above and explore more possibilities. Followings are the most popular hybrid recommender system approaches (Aggarwal 2016):

- feature combination/augmentation
- parallel design
- sequential design
- weighted and/or monolithic, etc.

In our design, we use weighted feature combination, cascade the output of CF to CB ,and finally use context and cascade the output of HY to CAARS to produce the list of recommendations.

2.3 Context-Aware Recommender Systems

Context is often defined as an aggregate of various categories that describe the setting in which a recommendation is deployed (Verbert et al. 2012). Time, location, social interaction, the user's mood, etc. can be considered as context. Context can be a property of the user, a property of the item, a property of the combination of user and item, or totally independent from user and item. Context can be single-dimensional or multi-dimensional. One context can have no or composite relationship with other contexts. Context can be susceptible to noisy environments.

Several research in behavioral studies show that decision making is contingent upon the relevant context consumers in (Adomavicius et al. 2011). For example, a person might want to listen to a warm fuzzy song on a Saturday night with his family, but he might want to listen to an energy-boosting song on a Monday morning while commuting to work. Several research groups have shown that context-awareness builds a more robust recommender system.

Collecting appropriate context directly is difficult due to issues of privacy and stability. Context is not always readily available; it can be learned explicitly by gathering information through surveys or other means (Aggarwal 2016). Sometimes people are not comfortable sharing all the contextual data with others, which makes the surveys less diverse and less credible. In addition, data about context are sometimes variable, even inconsistent.

2.4 Implicit User Feedback

The core task of any RS is to profile users and items and find how to relate them. For that, RS depends on different types of input which we can broadly categorize as *explicit-feedback* and *implicit-feedback*. Explicit-feedback includes users' explicit opinions (e.g. ratings, usually provided in a numeric range like 1 to 5, -10 to 10, thumbs up or thumbs-down, etc.) regarding their interest in products. They are relatively easy to use/interpret but time-consuming and expensive to collect (Hu, Koren, and Volinsky 2008).

Implicit-feedback techniques seek to avoid this bottleneck by inferring something similar to the ratings that a user would assign from observations that are available to the system (Oard and Kim 1998). Implicit information is inherently ambiguous but a good source of representing complicated/interrelated information. For example, the number of times a user plays a song, their actions during and/or

[1] http://www.ismir.net/

[2] https://www.businesswire.com/news/home/20080104005604/en/Nielsen-Music-2007-Year-Music-Industry-Report

after listening to that song such as“ save to favorite”, replay, or share with friends, or skipping it in the middle, etc. imply very useful feedback. By gathering these behavioral patterns, we can compile a sizable database. Although, just like any other mechanism, users' implicit behavior might not always be obvious and requires a lot of pre- and post-processing.

3 Dataset

In all of the models, we use two large datasets: the Music Million Tweets Dataset (MMTD) which provides time, location and songs from the tweets of users from 2011 to 2013, and MusicBrainz which contains information of songs and artists' features such as artist type, artist area, and artist gender.

Dataset	Basic Statistics
# of unique artists	24673
# of unique users	214741
# of unique tweets	1074713
# of unique tracks	133228

Table 1: Basic Statistics of MMTD dataset

We use tweet count as the initial implicit feedback.

3.1 Music Million Tweets Dataset

The dataset (Hauger et al. 2013) contains listening histories inferred from Twitter, each of which identified by an unique Tweet ID and linked to a user ID. Each tweet is annotated with temporal (date, time, weekday, timezone), spatial (longitude, latitude, continent, country, county, state, city), and demographic information of the country. Most importantly, each tweet contains a song with the corresponding artist(s). The dataset also includes song and artist references to other music-related platforms (musicbrainz, 7digital, amazon). There are a few other freely-available collections of audio features and metadata for music tracks like Lastfm (Bertin-Mahieux et al. 2011).

3.2 MusicBrainz

MusicBrainz[3] is a music metadata project that includes information about artists, release groups, releases dates, recordings, works, and labels, as well as the many other relationships between them. For this work, we consider two features:{*artist's type, artist's gender*}, as the content of artists.

The type attribute is used to state whether an artist is a person, a group, or something else. Musicbrainz dataset identifies six possible artist types: {*Person, Group, Orchestra, Choir, Character, Other*}. The gender attribute identifies a person as either *male, female* or *neither*. Groups do not have genders.

4 Our Hypothesis

Our hypothesis is that **the temporal pattern of tweets can provide helpful information to recommender systems**.

[3]https://musicbrainz.org/doc/MusicBrainz_Database

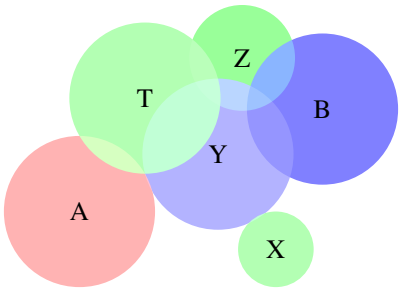

Figure 1: User-User Similarity based on Context and Content

For example, if two users like to tweet about songs usually at late night, it is quite possible that their choice of songs are similar to some extent. We only need to determine how to adjust this context to other features while building a recommender system. Our hypothesis is explained through figure 1.

Let we have six users: $\{T, A, B, X, Y, Z\}$ in our dataset. All of them may or may not live in the same time-zone; but they are similar in terms of some content (expressed as overlapped circles) and/or context (expressed with color codes). The distance between the circles in the figure represent the geographical distance between the users. The radius of the circles represent the amount of implicit information we have about each user. From the figure, we see that user Z and X are more similar to user T in terms of context than the other users. So, a context-aware recommender system should prioritize Z and X's preference lists over the rest of the users while suggesting some options for T.

If we can determine related and dominant contexts, we can use them along with content and other features to enhance the performance of recommender systems. In the next section, we explain how we implement this hypothesis to build our model.

5 Our Methodology

In the next couple of sections we detail the steps of our approach.

5.1 Preprocessing Data

One of MusicBrainz' aims is to be the universal lingua franca for music by providing a reliable and unambiguous form of music identification; this music identification is performed through the use of MusicBrainz Identifiers (MBIDs). An MBID is a 36 character Universally Unique Identifier (UUID) that is permanently assigned to each entity in the database. We used this key between the MMTD and MusicBrainz dataset to extract artist's *type, gender, area* as possible contents for our systems.

5.2 Our Model

Figure 2 shows the basic components of our model. We detail the steps as follows:

CF System: The collaborative filtering stage builds a neighborhood matrix by counting the total tweets of

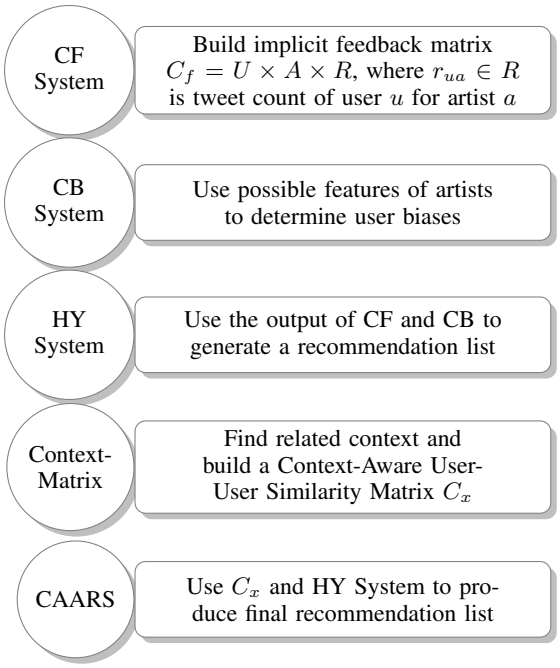

Figure 2: Basic Architecture

each user for each artist. As the tweet count implicitly provides some feedback, we use the Alternating Least Square (Felsenstein 1997) method to factor our matrix into small neighborhoods and find similar users based on their tweet counts. In order to calculate the final recommendation score of each artist for each user, we use the method discussed by Hu, Koren, and Volinsky(2008).

We use the following notations in describing our model: Let U be the set of all users, A be the set of all artists, and R be our implicit feedback matrix with dimensions $|U| \times |A|$. Each element $r_{ua} \in R$ represents the total number of tweets posted by user u for artist a. By using the alternating least-square method, we calculate the recommendation (collaborative filtering) score, $cf\text{-}score(u, a)$, of each test user for each artist.

CB System: We consider two features (artist's gender and artist's type) as content, each with its own implicit vectors. We compute the bias, $f\text{-}bias(u, f_i)$, of a user u toward a feature f_i with the formula:

$$f\text{-}bias(u, f_i) = \frac{twf(u, f_i)}{tw(u)} \tag{1}$$

where, $twf(u, f_i)$ is the total tweets made by u for a specific feature category f_i and $tw(u)$ is the total number of tweets made by u.

Suppose a user makes 10 tweets, 4 for female artist, 3 for male artist, 2 for groups, and 1 for others. Our system will calculate the user's bias toward gender feature as table 2. Similar computation can be done for the type feature.

Since we use only two features, we call them $gender\text{-}bias$ and $type\text{-}bias$.

category	$gender\text{-}bias$
male	0.4
female	0.3
group	0.2
other	0.1

Table 2: Calculating Gender-bias

HY System: Our hybrid model combines $cf\text{-}score(u, a)$ with $gender\text{-}bias$, and $type\text{-}bias$ using the following formula:

$$\begin{aligned} hy\text{-}score(u, a) = cf\text{-}score(u, a) \times \\ \{1 + gender\text{-}bias(u, gender(a)) \\ + type\text{-}bias(u, type(a))\} \end{aligned} \tag{2}$$

If the $gender\text{-}bias$ and $type\text{-}bias$ of a user is zero or insignificant, the HY system generally predicts the same list as the CF system.

Modeling Context Metrix: We divide the twenty-four hours of a day into four major time-slots:

- Morning (5:00 AM - 12:00 PM)
- Afternoon (12:00 PM - 5:00 PM)
- Evening (5:00 PM - 9:00 PM)
- Night (9:00 PM - 5:00 AM)

We count total tweets of each user in each time-slot based on his/her/their **local time-zone**. Then we apply the *Chi-Square Test of Independence* between time-slots (normalized, as we do not divide the durations evenly) and *total tweet count*. The $p\text{-}value$ is less than 0.001 with a confidence of 99%, which indicates that time-slots and tweet-counts are *dependent* on each other.

We consider *weekday vs. weekend* as a possible second context. But *weekday vs. weekends* does not show dependency with *total tweet count*. Hence, we use **time-slot** as the only context in our current model.

Context-aware Similarity Matrix: We build a context-aware, $|U| \times 4$ dimensional, user-user similarity matrix C_x. In C_x, each row represents a distinct user and each column represents the normalized tweet count of the user based on our chosen context: *time-slot=(morning, afternoon, evening, night)*.

$$C_x = \begin{pmatrix} 0.28 & 0.43 & 0.0 & 0.29 \\ 0.50 & 0.50 & 0.0 & 0.00 \\ 0.67 & 0.33 & 0.0 & 0.00 \\ 0.00 & 0.0 & 0.00 & 1.00 \\ \cdots & & & \\ \cdots & & & \end{pmatrix} \tag{3}$$

As we have a multi-dimensional vector to define each user, we find neighbors of each of them from C_x using KD-Tree (Bentley 1975). KD-Tree is a space partitioning data structure and commonly used for finding nearest neighbors in a K-dimensional space. We use the implementation of KD-Tree from SciPy library. The KD-Tree

returns a distance score, $dist(u_i, u_j)$, between each pair of users u_i and u_j. The range of the $dist$ scores is $[0, 1]$. We change it to similarity score by using the following formula:

$$sim\text{-}score(u_i, u_j) = 1 - dist(u_i, u_j) \quad (4)$$

In our model, we use the KD-Tree with neighborhood size $N = 20$. For each test user u_t, we get a list of 20 nearest neighbors $N_{u_t} = (u_1, u_2, \dots)$ with corresponding $sim\text{-}score$ vector $S_{ut} = (s_{u_1}, s_{u_2}, \dots)$ where $s_{u_i} = sim\text{-}score(u_t, u_i)$.

Context-aware Hybrid Model: Our system performs the steps discussed in Algorithm 1 to find the final ranked list. The first block (line no #1 to #9) of the algorithm finds a collection of (size = $|N_{u_t}| * k$) preferred artists by considering the neighborhood, N_{u_t}, of the test user u_t. k is some positive integer. The second block (line no #12 to #18) prepares the final rank of the preferred artists by considering the number of times the artists appear in the neighborhood list and by taking the maximum preference score for each artist. Function $Top_k(D)$ returns top k keys from a given dictionary D after sorting the items in decreasing order of their values.

Algorithm 1: Context-aware Model

Input: u_t, U, A, (N_{u_t}, S_{u_t})
Output: $(a_1, a_2, \dots, a_k)$

1 Initialize an empty list, L_{u_t}
2 **for** $u_i \in N_{u_t}$ **do**
3 Initialize an empty dictionary, E_{u_i}
4 **for** $a \in A$ **do**
5 $v_{a_{u_i}} = hy\text{-}score(u_i, a) * s_{u_i}$
6 $E_{u_i} = E_{u_i} \cup (a, v_{a_{u_i}})$
7 **end**
8 $L_{u_t} = L_{u_t} \cup Top_k(E_{u_i})$
9 **end**
10 L_{u_t} contains $|N_{u_t}| * k$ entries possibly with some duplicate artist entries as some artists may appear in the favorite lists of multiple neighbors of u_t
11 We convert the list L_{u_t} to a dictionary of lists where the artist id is the key to combine the multiple entries for some artists
12 Initialize an empty dictionary, RA
13 **for** $(a, \{v_{a_1}, v_{a_2}, \dots\}) \in L_{u_t}$ **do**
14 $s_a = max(v_{a_1}, v_{a_2}, \dots)$
15 $c_a = \log(count(v_{a_1}, v_{a_2}, \dots))$
16 $rank_a = s_a + c_a$
17 $RA = RA \cup (a, rank_a)$
18 **end**
19 return $Top_k(RA)$

6 Performance Analysis

In order to evaluate the performance of our design, we split the dataset into different train-test ratio (90-10, 80-20, 70-30, 60-40) and calculate *precision, recall,* and *F-measure* on the test users with fifteen or more tweets. The detailed reports are presented in table 3 and 4. Test no $1, 2, 3, 4$ use test users only found on the test set and test no 5 uses all users from the dataset for performance evaluation. In all of these test cases, CAARS shows significant improvement over the basic CF and HY models.

#	Train-Test	# of tweets	# of users	# of users with tweets ≥ 15
1	**90-10**	107472	47689	250
2	**80-20**	214943	78547	812
3	**70-30**	322414	103785	1621
4	**60-40**	429886	125008	2463
5	**60-40**	1074713	214741	6840

Table 3: Statistics of Performance Analysis

#	Train-Test	Scale	CF	HY	CAARS
1	90-10	Precision	0.005371	0.005371	0.182933
		Recall	0.001512	0.001512	0.041596
		F-measure	0.002234	0.002234	0.062847
2	80-20	Precision	0.004961	0.004961	0.147209
		Recall	0.00194	0.00194	0.048172
		F-measure	0.002628	0.002628	0.068338
3	70-30	Precision	0.003717	0.003717	0.126918
		Recall	0.001818	0.001818	0.05374
		F-measure	0.002303	0.002303	0.070773
4	60-40	Precision	0.003581	0.003581	0.117932
		Recall	0.001922	0.001922	0.057593
		F-measure	0.002382	0.002382	0.072448
5	60-40	Precision	0.003163	0.003163	0.106881
		Recall	0.001887	0.001887	0.06040
		F-measure	0.002242	0.002242	0.071867

Table 4: Detailed Results

Note that with our implementation and the current dataset, the CF system and the HY system always predict the same set of songs for each user with slight difference in ranks. Hence, their performance scores were the same.

7 Conclusion & Future Directions

"Music is the shorthand of emotion."

– Leo Tolstoy

Recommending music is a very challenging task due to the complex nature of the human mind and emotions. Nonetheless, in recent times, with the advent of robust and scalable systems, human activities can be observed (with proper consent) and analyzed to produce useful contextual information. Researchers working in the music information retrieval and recommender systems are using these information in several directions for providing better music recommender systems. Our model is a modest contribution to these efforts. CAARS is an artist recommender system for Twitter users and it shows better accuracy in recommending artists.

In this work, we show that implicit feedback (e.g., tweet count) is a promising way of modeling users to recommend

artists and songs. We also find that context inference (even on a very small scale) shows great improvement in the performance of artist recommender systems. Our main contributions are:

- to combine MMTD and MusicBrainz dataset for extracting artists information that we use for finding user biases.
- to determine a relevant (and dominant) context (users tweet-time patterns) that we use successfully along with the hybrid model.

We can extend CAARS to recommend songs. One simple approach will be to pick the most popular song of a recommended artist. Other possible approaches will be to use the genre-bias of the user, to align the release date/year and current time, etc. to find possible songs of the recommended artists.

We plan to expand the research in several other dimensions. In terms of expanding context, we may correlate users' current locations to find local artists. It would be interesting to know which languages users are acquainted with, as multi-lingual users listen to songs in almost all those languages. We can expand the content by including more user information (age, gender, birthplace, current job, relationship status) and artist information. We may use lyrics and genre of the songs as features. The other major step will be building a generic system for collecting contextual data and implicit feedbacks of music lovers from various social media (e.g., YouTube, Twitter, Facebook, Pinterest, etc.) in order to offer a better recommender system.

References

Adomavicius, G.; Mobasher, B.; Ricci, F.; and Tuzhilin, A. 2011. Context-aware recommender systems. *AI Magazine* 32:67–80.

Aggarwal, C. C. 2016. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 1st edition.

Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18(9):509–517.

Bertin-Mahieux, T.; Ellis, D. P.; Whitman, B.; and Lamere, P. 2011. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Celma, O. 2010. Music recommendation. In *Music recommendation and discovery*. Springer. 43–85.

Elahi, M.; Ricci, F.; and Rubens, N. 2016. A survey of active learning in collaborative filtering recommender systems. *Comput. Sci. Rev.* 20(C):29–50.

Felsenstein, J. 1997. An alternating least squares approach to inferring phylogenies from pairwise distances. *Systematic biology* 46(1):101–111.

Goldberg, D.; Nichols, D.; Oki, B. M.; and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35(12):61–70.

Hauger, D.; Schedl, M.; Košir, A.; and Tkalčič, M. 2013. The million musical tweet dataset: What we can learn from microblogs. International Society for Music Information Retrieval.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. *2008 Eighth IEEE International Conference on Data Mining*.

Metabrainz, F. 2015. Musicbrainz database.

Oard, D., and Kim, J. 1998. Implicit feedback for recommender systems. In *in Proceedings of the AAAI Workshop on Recommender Systems*, 81–83.

Shardanand, U., and Maes, P. 1995. Social information filtering: algorithms for automating " word of mouth". 95:210–217.

Verbert, K.; Manouselis, N.; Ochoa, X.; Wolpers, M.; Drachsler, H.; Bosnic, I.; Member, S.; and Duval, E. 2012. Context-aware recommender systems for learning: A survey and future challenges. *IEEE Trans. Learn*.