Artist Recommendation based on Association Rule Mining and Community Detection

Okan Çiftçi, Samet Tenekeci and Ceren Ülgentürk
Department of Computer Engineering, İzmir Institute of Technology, İzmir, Turkey

Keywords: Association Rule Mining, Community Detection, Recommender Systems, Graph Databases.

Abstract: Recent advances in the web have greatly increased the accessibility of music streaming platforms and the amount of consumable audio content. This has made automated recommendation systems a necessity for listeners and streaming platforms alike. Therefore, a wide variety of predictive models have been designed to identify related artists and music collections. In this paper, we proposed a graph-based approach that utilizes association rules extracted from Spotify playlists. We constructed several artist networks and identified related artist clusters using Louvain and Label Propagation community detection algorithms. We analyzed internal and external cluster agreements based on different validation criteria. As a result, we achieved up to 99.38% internal and 90.53% external agreements between our models and Spotify's related artist lists. These results show that integrating association rule mining concepts with graph databases can be a novel and effective way to design an artist recommendation system.

1 INTRODUCTION

With the widespread use of the Internet, music distribution channels have diversified. As a result, more and more music becomes consumable every day. At this point, just as listeners want to discover new songs and artists, music streaming platforms seek to make the right recommendations to increase the time their users spend on the application. For this purpose, many recommendation systems have been developed that make automatic music and artist suggestions using users' activities, common behavior patterns, or other metadata.

These systems adopt state-of-the-art methods of graph theory, statistics, data mining, machine learning, and deep learning. They can be classified by the type of input features they used. In the literature, music and artist recommendation systems have been designed as content-based (based on text and audio features) (Cano et al., 2005a; Cano et al., 2005b; Yoshii et al., 2006; Chen et al., 2011), context-aware (based on playlist and category metadata) (Han et al., 2010; Hariri et al., 2012; Pichl et al., 2015), location-aware (based on geospatial data) (Schedl and Schnitzer, 2014; Kaminskas et al., 2013; Cheng and Shen, 2016), culture-aware (based on cultural metadata) (Baumann and Hummel, 2003; Baumann and Hummel, 2005; Zangerle et al., 2018), graph-based (based on topological features) (Cano et al., 2006; Celma and Herrera, 2008; Yin et al., 2012), or integrating multiple features (Neumayer and Rauber, 2007; Wang et al., 2012; Schedl, 2013).

A collection of related artists can be modeled as a social network where each node represents an artist and each edge represents the strength of the relationship between an artist pair. Graph-based models are effective in representing such complex networks. Additionally, the topological properties of graphs can be used to detect communities in these networks. In this context, we proposed a graph-based approach for the problem of artist recommendation.

Our approach can be included in the class of context-aware methods since it relies on playlist data. It utilizes well-known graph theory and data mining techniques on Spotify playlists to extract association rules and corresponding artist communities in a graph database. The workflow of our method includes 4 main steps: (1) identify frequent artist sets and association rules in Spotify playlists, (2) construct a graph database using the support and confidence values calculated in Step 1, (3) discover related artist communities by running state-of-the-art community detection algorithms on the constructed artist network, (4) perform internal and external cluster validations on discovered communities. Among the similar artist recommendation systems, our approach is novel as it is

Çiftçi, O., Tenekeci, S. and Ülgentürk, C.
Artist Recommendation based on Association Rule Mining and Community Detection.
DOI: 10.5220/0010678600003064
In *Proceedings of the 13th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2021) - Volume 1: KDIR*, pages 257-263
ISBN: 978-989-758-533-3; ISSN: 2184-3228

the first attempt to perform community detection on graph databases built using the concepts of support and confidence.

The remainder of this paper is organized as follows: Section 2 introduces the related work. Section 3 describes materials, methods, and evaluation metrics used, along with the running environment and performance. Section 4 includes experimental results as well as the discussions and threats to validity. Finally, Section 5 concludes the paper and presents the future work.

2 RELATED WORK

Our related work consists of approaches that adopt graph-based algorithms, with a few exceptions that are noted.

In (Cano et al., 2006), the authors conducted a study on different music recommendation systems by means of complex network analysis. They examined the common and distinctive topological features of AllMusicGuide, MSN Entertainment, Amazon, and Launch Yahoo! Music. Their results showed that despite some common features, such as small worldness, different network characteristics exist, such as the link degree distribution.

In (Celma and Herrera, 2008), two graph-based approaches, namely Item- and User-centric, have been proposed to evaluate the quality of novel recommendations. The authors tried to detect whether the network topology has any pathology that hinders novel recommendations and measure users' perceived quality of novel, previously unknown, recommendations. They also compared the content-based and social-based recommendation methods.

In (Anglade et al., 2011), the authors designed a recommendation system based on the music preferences of users' social connections. To this end, they used the social shuffle principle in graphs representing the social interactions between the users. They developed an application called *Starnet* to verify their claims. As a result, they proved the effect of social communication networks on music preferences.

In (Yin et al., 2012), the authors proposed a graph-based artist recommendation system that utilizes listening and trust preference networks (LTPN). They combine listening and trust information provided by users and unfold his/her reliable friends with similar tastes to make better recommendations. Their experimental results demonstrate LTPN can not only provide better recommendation but also help relieve the cold start problem caused by new users.

In (Wang et al., 2014), the authors worked on a graph-based recommendation system for social networks. They used rating and tag information and charted the relationship based on the co-tagging behavior of users. They aimed to design a more accurate recommendation system by supporting the random walk with restart algorithm on tags. As a result, they were able to achieve good performance and accurate propositions.

In (Turnbull and Waldner, 2018), the authors tried to find a solution to the task of local music event recommendation. It is difficult to find the relationship between local music artists since they tend to be obscure long-tail artists with a small digital footprint. To address this problem, they utilized *Latent Semantic Analysis* (LSA). They embedded artists and tags into a latent feature space and effectively modeled artist similarity. They also introduced the concept of a *Music Event Graph* that makes it easy and efficient to recommend events based on user-selected genre tags and popular artists.

Lastly, in (Yakura et al., 2018), the authors proposed a system to make background music suggestions based on users' feedback. They especially worked on the concentration-enhancing background music used while working. They did not use a graph-based approach but emphasized the power of the recommendation system as they proved the focus-enhancing effect of the suggested songs.

3 MATERIALS AND METHODS

3.1 Dataset & Preprocessing

In 2018, Spotify helped organize the RecSys Challenge 2018, a data science research challenge focused on music recommendation. As part of that challenge, Spotify introduced The Million Playlist Dataset (Chen et al., 2018) a dataset of 1 million playlists consisting of over 2 million unique tracks by nearly 300,000 artists. This represents the largest public dataset of music playlists in the world. From this challenge we accessed a sample. It contains approximately 663,000 tracks in 9999 playlists with 172,000 unique tracks and 36,000 unique artists. It contains track, artist, album, playlist ids, track name, album name and artist name. In this task we group by each artist by playlist id then used each playlist as a transaction to find association rules between artists therefore we only use playlist id and artist name from given playlist.

3.2 Frequent Itemsets & Association Rule Mining

We utilize the well-known Apriori algorithm (Agrawal and Srikant, 1994) to obtain frequent itemsets and association rules for our dataset. Apriori algorithm is often used for the analysis of co-occurring items over relational databases. Essentially, it uses Boolean association rules to evaluate the features and transactions. It starts with identifying the common individual items in the database and proceeds by extending them to larger itemsets as long as the frequency of these itemsets is higher than a certain threshold (i.e. minimum support criterion). For example, given two items, X and Y, Apriori algorithm defines the support and confidence values as:

$$Supp(X,Y) = \frac{Frequency(X \cup Y)}{N}$$
$$Conf(X,Y) = \frac{Supp(X,Y)}{Supp(X)} \quad (1)$$

where N is the total number of transactions in the database. According to the Apriori algorithm, if a k-itemset (i.e. an itemset with k elements) provides the minimum support value, the subsets of this set also satisfy the minimum support criterion. The frequent itemsets determined by Apriori can be used to determine association rules which highlight general trends in the database.

3.3 Graph Clustering

We perform dimensional reduction on our dataset to extract the 100 artists with the strongest interactions. By trial and error, we select a threshold of 0.23 for support and confidence values and filter out links with weights below this threshold. In this way, we both speed up the experiments and increase the reliability of the results. For convenience, we call the 100 artists we selected *source artists*.

We use both support and confidence values as edge weights on artist networks. To generate clusters on each graph, we run Louvain community detection algorithm (Zachary, 1977; Lu et al., 2015) and Label Propagation algoritm (Xing et al., 2014) which are available on Neo4j[1] graph database platform. For each algorithm, we perform two different clustering using both support and confidence values. For convenience, we call these methods *LouSupp*, *LouConf*, *LabSupp*, and *LabConf*.

[1]http://neo4j.org

3.3.1 Louvain Algorithm

Louvain algorithm aims to detect communities in large networks. It maximizes a modularity score for each community, where the modularity quantifies the quality of an assignment of nodes to communities. Louvain uses a greedy optimization method that runs in time $O(n \log n)$, where n is the number of nodes in the network. In Louvain algorithm, first the small communities found by optimizing modularity locally on all nodes, and then each small community is grouped each other.

3.3.2 Label Propagation Algorithm

LPA is a fast algorithm for finding communities in a graph. It detects these communities using a graph structure. Steps of the algorithm are as follows: (i) every node initialize with unique community label, (ii) labels propagate through the network, (iii) in every iteration on propagation, each node updates its label to the one that the maximum numbers of its neighbours belongs to. Algorithm reaches convergence when each node has the majority label of its neighbours. It can stop either convergence or maximum number of iterations is achieved. At the end of the propagation process only few labels remain. Nodes that have the same community label at convergence are said to belong to the same community.

3.4 Retrieving Ground Truths from Spotify

As ground truth, we use related artists from Spotify. We retrieve 20 related artists for each of 100 artists (i.e. the *source artists*) in our dataset using Spotipy API[2]. Since we are only interested in the relationships between our *source artists*, we exclude other related artists from this collection. After filtering, we remove clusters that contain a single element. Further details about Spotify's related artist collection is given in Section 4.2.

3.5 Evaluation Metrics

We use the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) and the Pairwise Overlap Coefficient (POC) which is a modification of the Overlap Coefficient (OC) (Vijaymeena and Kavitha, 2016), for internal and external cluster validation. ARI works only on disjoint clusters, while POC can be applied to both disjoint and overlapping clusters. Thus, we can utilize both metrics to calculate pairwise agreement ratios

[2]https://spotipy.readthedocs.io

Table 1: The contingency table.

	Y_1	Y_2	...	Y_s	sums
X_1	n_{11}	n_{12}	...	n_{1s}	a_1
X_2	n_{21}	n_{22}	...	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	...	n_{rs}	a_r
sums	b_1	b_2	...	b_s	

of LouSupp, LouConf, LabSupp, and LabConf, that all consist of disjoint clusters. This is called internal validation (see Section 4.1). On the other hand, we can use only the POC to calculate similarity between our graph communities and Spotify's related artist sets, because of the fact that these sets are overlapping. The comparison with Spotify data is called external validation (see Section 4.2)

We utilize `adjusted_rand_score` from *scikit-learn* library (Pedregosa et al., 2011) for internal validation of clusters. Given a set S of n elements, and two clusterings of these elements, namely $X = \{X_1, X_2, \ldots, X_r\}$ and $Y = \{Y_1, Y_2, \ldots, Y_s\}$, the overlap between X and Y can be summarized in a *contingency table* (Table 1) where each entry n_{ij} denotes the number of elements in common between X_i and $Y_j : n_{ij} = |X_i \cap Y_j|$. Using the *contingency table*, ARI can be formulated as:

$$\frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \quad (2)$$

where n_{ij}, a_i, and b_j are values from the *contingency table*. Typically, ARI score is between -1.0 and 1.0. It is close to 0.0 for randomly labeled clusters. In case of perfect match (i.e. identical clustering) ARI is 1.0. A negative ARI represents that the agreement is less than what is expected from a random result. Further details about ARI is available in (Hubert and Arabie, 1985). The internal validation results are presented in Section 4.1.

We utilize POC, which is a modification of OC, for both internal and external validation of clusters. Given two sets of overlapping clusters, $X = \{(a,b,c),(c,d,e)\}$ and $Y = \{(a,b,c),(b,d),(b,e),(c,a)\}$, the traditional OC is defined as:

$$OC(X,Y) = \frac{|X \cap Y|}{min(|X|,|Y|)} \quad (3)$$

and equal to 0.5, since only the (a,b,c) cluster is matching. To calculate POC, on the other hand, we first generate pairs of elements (i.e. tuples) using the elements in the same clusters. Then, we filter out the duplicated and symmetric pairs in each set and obtain reduced sets of pairs. Finally, we calculate the OC between these sets. For X and Y, these sets are $X' = \{(a,b),(a,c),(b,c),(c,d),(c,e),(d,e)\}$ and $Y' = \{(a,b),(a,c),(b,c),(b,d),(b,e)\}$, respectively. Hence, $POC(X,Y) = OC(X',Y') = 0.6$.

Table 2: Execution Times (ms) of Clustering Methods.

	LouSupp	LouConf	LabSupp	LabConf
Time	94	200	12	24

In case of a set of disjoint clusters, $X = \{X_1, X_2, \ldots, X_n\}$, the total number of pairs is:

$$|X'| = \sum_{i=1}^{n} \frac{|X_i|(|X_i|-1)}{2} \quad (4)$$

In the case of overlapping clusters (as in the example above), duplicated and symmetrical tuples are excluded from this set. After we form the sets of artist pairs for both clusterings, we calculate the POC. In this way, we determine to what extent the two given clusterings agree in associating the given artist pairs. The internal and external validation results are presented in Section 4.1 and Section 4.2, respectively.

3.6 Running Environment & Performance

In this work, we mostly use Python programming language and Jupyter Notebooks[3] for development. We utilize Apyori[4] library to create association rules, Cypher query language and neo4j platform to store and manage our graph database, and scikit-learn[5] library to validate our clusters. We use netgraph[6] for visualization. We run the experiments on a MacOS Catalina device with i7 9700 3.6 GHz Intel processor, 16GB RAM, and 512GB SSD. Table 2 shows the execution time of each clustering method. Two conclusions can be drawn from our performance analyses: (1) the clustering algorithms run about twice as fast when support is used instead of confidence, (2) the Label Propagation algorithm is about 8 times faster than the Louvain algorithm.

4 EXPERIMENTAL RESULTS

In this section, we present the experimental results in three parts. First, we provide some important statistics about graph-based communities generated by Louvain and Label Propagation algorithms and

[3]https://jupyter.org
[4]https://github.com/ymoch/apyori
[5]https://scikit-learn.org
[6]https://github.com/paulbrodersen/netgraph

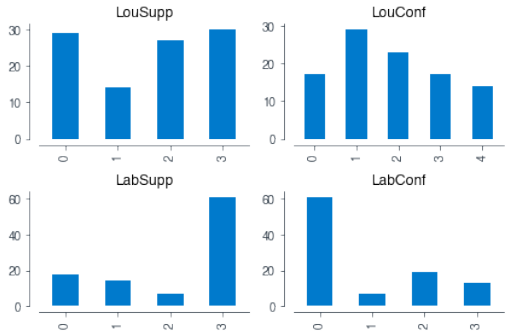

Figure 1: Data distribution of four clustering methods.

give internal validation results based on ARI and POC metrics. Then, we provide data statistics about Spotify's related artist collection and present external validation results based on POC. In the last part, we discuss the results and list the threats to validity.

4.1 Graph-based Communities (Internal Validation)

Among our 4 community detection methods, LouConf detected 5 communities while each of the others (LouSupp, LabSupp, and LabConf) detected 4. The smallest and largest communities were generated by the Label Propagation algorithm, with 7 and 61 artists, respectively. On the other hand, Louvain's communities ranged in size from 14 to 30. Note that the communities generated by each method are disjoint among themselves. Figure 1 presents the data distribution obtained by each clustering method. Considering the community sizes, the Louvain algorithm provides a more balanced distribution. Figure 2 illustrates the communities generated by the LouConf algorithm. Considering the edge densities, the artist network has a scale-free degree distribution. In other words, it has a small number of highly connected hub nodes (these are famous American rappers like Drake, Kanye West, Kendrick Lamar, and Future) and a large number of weakly connected nodes.

Table 3: ARIs for Internal Validation.

	LouSupp	LouConf	LabSupp	LabConf
LouSupp	1	0.7313	0.4485	0.4355
LouConf	0.7313	1	0.3228	0.3099
LabSupp	0.4485	0.3228	1	0.9872
LabConf	0.4355	0.3099	0.9872	1

Table 4: POCs for Internal Validation.

	LouSupp	LouConf	LabSupp	LabConf
LouSupp	1	0.8943	0.8239	0.8137
LouConf	0.8943	1	0.7789	0.7661
LabSupp	0.8239	0.7789	1	0.9938
LabConf	0.8137	0.7661	0.9938	1

Table 5: POCs for External Validation.

	# of pairs	# of overlaps	POC
LouSupp	1283	379	0.5279
LouConf	1022	320	0.4457
LabSupp	2095	647	0.9011
LabConf	2100	650	0.9053

* Number of artist pairs in Spotify clusters is 718.

4.2 Related Artists from Spotify (External Validation)

For external cluster validation, we compare Spotify's related artist collection with clusters generated by Louvain and Label Propagation algorithms. We select 20 related artists for each *source artist* and obtain a collection of 2100 artists (including the *source artists*), 730 of which are distinct. We can also think of this collection as 100 clusters, each consisting of 21 artists. After filtering out the related artists that are not in the *source artists*, we remove clusters that contain a single element. As a result, we have 87 clusters that range in size from 2 to 17.

To calculate POCs, we generate sets of artist pairs for 100 artists featured in 87 overlapping clusters. After we form all artist pairs, which are 1649 pairs according to Equation 4, we filter out the duplicated and symmetrical tuples and obtain a final set of size 718.

4.3 Discussions & Threats to Validity

In internal validation, both ARI and POC analyses show that agreement between support-weighted and confidence-weighted clusters is lower in the Louvain algorithm (ARI = 73.13%, POC = 89.43%) compared to the Label Propagation algorithm (ARI = 98.72%, POC = 99.38%). On the other hand, the highest cross-algorithm agreement is achieved using support values (ARI = 44.85%, POC = 82.39%). As an interesting side note, LouConf shows higher agreement with LabSupp than LabConf, although the results are very close for both metrics.

In external validation, regardless of the edge weighting method used, the Label Propagation algorithm (90.11% and 90.53% for support and confidence, respectively) significantly outperforms the Louvain algorithm (52.79% and 44.57% for support

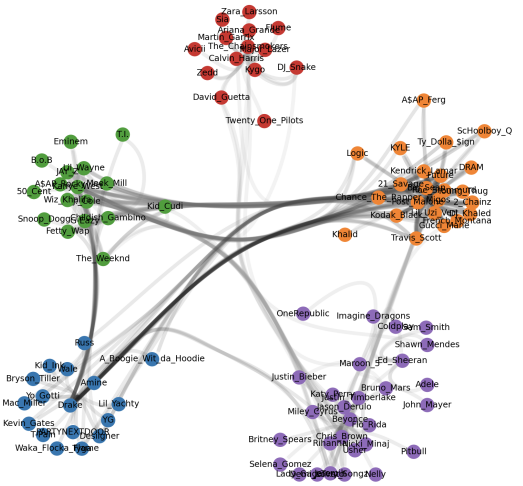

Figure 2: Communities generated by LouConf algorithm.

and confidence, respectively), and it shows a higher agreement with Spotify's related artist clusters.

Both ARI and POC analyses show that the Label Propagation algorithm on a graph with confidence-weighted edges has the highest cluster agreement, both internally and externally. On the other hand, our performance analyses show that Label Propagation executes significantly faster than Louvain (see Section 3.6). The statistical superiority of the Label Propagation algorithm in clustering performance can be attributed to the fact that it contains a large cluster with 61 artists.

The unbalanced cluster distribution in the Label Propagation algorithm could be a threat to the validity of our results. Similarly, the date mismatch between the dataset we work on (September, 2018) and the related artist collection that we retrieved from Spotify (June, 2021) could be a threat to the validity. Methodologically, limiting the dataset to a subset of 100 artists in this work would be a threat when scaling our model up to larger and more complex networks. Lastly, using only the contextual features (playlist metadata) to generate association rules could be another threat to validity.

5 CONCLUSIONS & FUTURE WORK

Recent advances in internet technology have greatly increased the accessibility of music streaming platforms and the amount of consumable audio content. This has made automated recommendation systems a necessity for both listeners and streaming platforms. As a result, various models have emerged that use different input features and computational methods to detect related artists and music collections.

In this work, we proposed a graph-based model that relies on contextual features (i.e. playlist data) and association rules. We used support and confidence metrics as edge weights in artist network. We utilized Louvain and Label Propagation community detection algorithms to identify clusters of related artists. We performed internal and external validations of clusters using the Adjusted Rand Index (ARI) and Pairwise Overlap Coefficient (POC).

We achieved clustering agreements up to 98.72% between support and confidence metrics, 44.85% between Louvain and Label Propagation algorithms, and 90.53% between our model and Spotify's related artists. These results show that integrating association rule mining concepts with graph databases can be a novel and effective way to design a recommendation system.

In future work, association rules and links in the artist network can be semantically enriched by integrating contextual data with other input features like textual, categorical, cultural, or geospatial metadata. Additionally, this model can be extended using other datasets and community detection algorithms.

ACKNOWLEDGEMENTS

We would like to thank Dr. Damla Oğuz from İzmir Institute of Technology for their comments and suggestions on this study.

REFERENCES

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of 20th international conference on very large data bases*, pages 487–499, Morgan Kaufmann Publishers Inc. VLDB.

Anglade, A., Celma, Ò., Fields, B., Lamere, P., and McFee, B. (2011). Womrad: 2nd workshop on music recommendation and discovery. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 381–382.

Baumann, S. and Hummel, O. (2003). Using cultural metadata for artist recommendations. In *Proceedings Third International Conference on WEB Delivering of Music*, pages 138–141. IEEE.

Baumann, S. and Hummel, O. (2005). Enhancing music recommendation algorithms using cultural metadata. *Journal of New Music Research*, 34(2):161–172.

Cano, P., Celma, O., Koppenberger, M., and Buldu, J. M. (2006). Topology of music recommendation networks. *Chaos: An interdisciplinary journal of nonlinear science*, 16(1):013107.

Cano, P., Koppenberger, M., and Wack, N. (2005a). Content-based music audio recommendation. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 211–212.

Cano, P., Koppenberger, M., and Wack, N. (2005b). An industrial-strength content-based music recommendation system. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 673–673.

Celma, Ò. and Herrera, P. (2008). A new approach to evaluating novel recommendations. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 179–186.

Chen, C. W., Lamere, P., Schedl, M., and Zamani, H. (2018). Recsys challenge 2018: Automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 527–528.

Chen, Z. S., Jang, J. S. R., and Lee, C. H. (2011). A kernel framework for content-based artist recommendation system in music. *IEEE Transactions on Multimedia*, 13(6):1371–1380.

Cheng, Z. and Shen, J. (2016). On effective location-aware music recommendation. *ACM Transactions on Information Systems (TOIS)*, 34(2):1–32.

Han, B. J., Rho, S., Jun, S., and Hwang, E. (2010). Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications*, 47(3):433–460.

Hariri, N., Mobasher, B., and Burke, R. (2012). Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 131–138.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.

Kaminskas, M., Ricci, F., and Schedl, M. (2013). Location-aware music recommendation using auto-tagging and hybrid matching. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 17–24.

Lu, H., Halappanavar, M., and Kalyanaraman, A. (2015). Parallel heuristics for scalable community detection. *Parallel Computing*, 47:19–37.

Neumayer, R. and Rauber, A. (2007). Integration of text and audio features for genre classification in music information retrieval. In *European Conference on Information Retrieval*, pages 724–727, Berlin, Heidelberg. Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *The Journal of machine Learning research*, 12:2825–2830.

Pichl, M., Zangerle, E., and Specht, G. (2015). Towards a context-aware music recommendation approach: What is hidden in the playlist name? In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1360–1365.

Schedl, M. (2013). Ameliorating music recommendation: Integrating music content, music context, and user context for improved music retrieval and recommendation. In *Proceedings of international conference on advances in mobile computing & multimedia*, pages 3–9.

Schedl, M. and Schnitzer, D. (2014). Location-aware music artist recommendation. In *International conference on multimedia modeling*, pages 205–213. Springer, Cham.

Turnbull, D. and Waldner, L. (2018). Local music event recommendation with long tail artists. *arXiv preprint arXiv:1809.02277*.

Vijaymeena, M. K. and Kavitha, K. (2016). A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3(2):19–28.

Wang, H., Li, G., and Feng, J. (2014). Group-based personalized location recommendation on social networks. In *Asia-Pacific Web Conference*, pages 68–80. Springer.

Wang, X., Rosenblum, D., and Wang, Y. (2012). Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 99–108.

Xing, Y., Meng, F., Zhou, Y., Zhu, M., Shi, M., and Sun, G. (2014). A node influence based label propagation algorithm for community detection in networks. *The Scientific World Journal*.

Yakura, H., Nakano, T., and Goto, M. (2018). Focusmusicrecommender: a system for recommending music to listen to while working. In *23rd International Conference on Intelligent User Interfaces*, pages 7–17.

Yin, C. X., Peng, Q. K., and Chu, T. (2012). Personal artist recommendation via a listening and trust preference network. *Physica A: Statistical Mechanics and its Applications*, 391(5):1991–1999.

Yoshii, K., Goto, M., Komatani, K., Ogata, T., and Okuno, H. G. (2006). Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. *ISMIR*, 6:296–301.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473.

Zangerle, E., Pichl, M., and Schedl, M. (2018). Culture-aware music recommendation. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 357–358.

APPENDIX

Relevant datasets and source codes are available at *https://github.com/okanvk/ArtistRecommendation*.