Music Recomendation System through LLM Summary

Noah Tekle
ntekle@usc.edu
USC
Los Angeles, California

Alline Ayala
allineayala@tamu.edu
Texas A&M
College Station, Texas

Jonathan Haile
hailej@usc.edu
USC
Los Angeles, California

Abstract

Recommendation systems have become fundamental to streaming services like Apple Music and Spotify[9], evolving into an increasingly significant area for research; therefore ensuring the highest possible hit rate is being utilized when developing these music recommendation systems is imperative. While many platforms focus on the title of the song as a differentiating factor, research has supported that utilizing summaries of the song, generated by LLMs, provides more valuable information to correctly identify the appropriate song to recommend [2]. This paper wishes to expand on this idea by investigating whether song summaries, opposed to truncated lyrics, improve the accuracy of recommendations. By comparing the effectiveness of implementing song summaries against truncated lyrics in a recommendation system, we are able to detect whether the additional context provided by these summaries are in fact what drives better results. Our analysis aims to reveal that leveraging song lyrics through lyrical summaries can significantly improve recommendation performance, potentially reshaping the approach to generating music recommendations.

CCS CONCEPTS

• Information Systems → Recommender systems; Data mining; Music data analysis;
• Computing methodologies → Machine learning; Natural language processing

Keywords

Fairness in Recommender Systems, Music Recommendation, Lyrics Analysis, Hot-Cold Data Balance, Bias Mitigation, Cross-Cultural Music Analysis

1 Introduction

Recommendation systems of all types have become essential due to the wide variation of applications it provides in a plethora of industries including but not limited to: education, healthcare, and entertainment[13]. These systems are critical in music platforms where the employment of an exceptional recommendation system can play a pivotal role in shaping user fidelity to a given platform and overall user experiences. The framework of building recommendation systems can be broken down to three large sub areas, Content-based filtering, Collaborative Filtering, and a Hybrid approach that adopts both the former filtering methods[14]. Content-based filtering gives recommendations based on the item and user preferences. This involves comparing an item's properties to a user's past behaviors and matching them accordingly. If a feature exits in the recommendation system it may utilize the cosine similarity

$$\text{cosine\_similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

to assess similarity between a potential item and user features recommending the song most closely correlated to said features [20]. The difficulty with content-based filtering is if a user has limited known preferences then it becomes increasingly difficult to give relevant recommendations[16]. Collaborative filtering is comprised of two different branches, user-based filtering where predictions are made based on user similarity, and item-based filtering where users' common items are matched together. Both forms of collaborative filtering are commonly calculated using the Pearson correlation as results in a calculated similarity between the two users or two items [20].

$$r_{XY} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}$$

Similar to Content-based filtering, Collaborative filtering also faces the same cold-start problem stemming from the absence of interactions between user and items. By combining the strong points of the two filtering methods into one hybrid recommender. It will lead to the combined machine learning model increasing its' overall accuracy[14]. If a user or item can only operate with one method then their filtering of choice can be used but if they have access to both, their recommendations can be improved as they have more data to pull from [14].

In this work we utilize the two tower Model [6], to form three distinct models: one utilizing song name, one utilizing the song summary, and one using the truncated lyrics of the song to build on the findings of Abu Mohammad Taief [2]. Going in a similar direction we decide to take the next step to justify if the summary caused the increase in accuracy or if the length of the summary was the true cause.

2 Methods

For the first model, we leveraged the Last.fm API to create our user database [23], then we downloaded the Kaggle Spotify [22] and Apple [21] datasets for our item datasets. For our second model we used the Groq API, running their supported LLM to gather song summaries [1], and for our third model we applied Lyrics OVH to receive truncated song lyrics[28].

2.1 Datasets

Last.fm was utilized to create our dataset of choice for 3 major reasons

- **Loved Tracks**: Last.fm enabled us to compile every user's last 200 loved tracks[11].
- **Tags**: By incorporating tags into our content-based filtering we expected an enhancement in our recommendation model as tags have displayed consistent reliability in classification accuracy (0.71)[12].
- **Location**: This dataset encompasses users with country details–allowing for more context-aware and nuanced recommendations in our system.

For Model 1, the user dataset contains about 8000 users with each user containing a name, country, and their last 200 liked songs. Our items dataset contains 10000 tracks, each also including the accompanying artist name and the tag attached to each song. Model 2 contains the same features as Model 1 with the substitute of song name for song summary (limited to 25 words) while Model 3 substitutes song name with the truncated lyrics (25 words).

While there are other user features that would likely increase the model accuracy such as age, or gender we decided against inclusion of said features as we wished to follow best privacy practices and not cause concern to the users, additionally it was not necessary to add in pursuit of our goal of seeing if summary based recommendation systems would perform better than a truncated lyrics recommendation system. However it would be interesting to see this topic re-explored by future researchers while ignoring privacy concerns to attempt the highest possible recommendation accuracy.

Our interaction dataset was created by mapping every user to each of their songs individually and if any songs was interacted by less than 5 users or if any users interacted with less than 5 songs they were filtered out to remove noisy data and allow the model to make more accurate predictions.

2.2 Datareader

These datasets are then converted to categorical in our datareader as our two tower model only operates on numerical values allowing us to begin directly setting up our model.

However the song name/ song summary/ truncated lyrics serves as an exception to this convert to categorical rule as each word is put through a sentence transformer which embeds each word into a vector of numbers

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

which is fed through the transformer layers [9].

In the tranformer layers a self-attention mechanishm is utilized to view every other vector and see how relevant it is to them allowing the phrase or sentence to understand its own meaning which is fundamental to this experiment.

3 Model Architecture

We begin set up for the model by developing an Interaction Graph that tracks user, item, and interaction data in order to calculate the model edges and see when multiple users and items interact [25]. To calculate the edges our model does

$$e.m = \alpha \cdot CF_s + (1 - \alpha) \cdot CF_r$$

where CFs denotes the collaborative filter model on the sender's side, CFr represents the collaborative filter model for the recipient, and alpha serves as the weight to balance the recipient and sender models [25]. Since these edges are returned as either 1 or 0, we minimize the unneeded parameters that store a majority of model memory. We then implement a sparse neural network that only stores the edges that exist and treat 0's as a non-existent[17]. Our model then splits our items into two groups: warm songs and cold songs. If a song is considered one of the least listened to songs in a given dataset (bottom x%) it becomes a cold song, otherwise it's seen as a warm song.

3.1 Set up computing

To accurately train our model and begin predictions we start developing negative edges which are user-item pairings that don't occur in our interactions dataset.

$$p_j \in P, \quad n_k \in N, \quad \text{and } p_j, n_k \in S$$

By creating interactions that don't exist it helps our model see predictions that can't exist. Then we collect the negative edges and the positive edges and once they've been trained and utilize them to generate our final score which will be used by our model to make predictions. Our model's final step before running the model is to calculate the dot product on the user and item

$$\mathbf{a} \cdot \mathbf{b} = \sum_i a_i b_i$$

which gets compressed and sent to the MLP to run the model.

3.2 Hyperparameters and Implementation

In our model we set the warm-threshold=0.2, characterizing cold items as songs found in the bottom 20% of our dataset. We concatenate our embedding features setting the embed dimension to 96, the output embedding to 192, and our dense features embedding dimension to 384. For the model we employ the activation layer 'gelu', apply add-bias, but don't apply layer normalization or elemwise-affine refinements.

4 Results and Discussion

4.1 Evaluation

To calculate hit rate, we observe the negative and the positive edges.

$$\text{Hit Rate@10} = \frac{1}{N} \sum_{i=1}^{N} [\text{true\_positive}_i \in \text{top\_k}_i]$$

which is the percentage of users that have liked at least one song out of their recommended 10 songs.

We also calculate the NDCG as follows:

$$\text{NDCG@10} = \frac{\text{DCG@10}}{\text{IDCG@10}}$$

where the Discounted Cumulative Gain (DCG) at rank 10 is given by:

$$\text{DCG@10} = \sum_{i=1}^{10} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}$$

and the Ideal Discounted Cumulative Gain (IDCG) is the DCG of the ideal (best possible) ranking at rank 10:

$$\text{IDCG@10} = \sum_{i=1}^{10} \frac{2^{\text{rel}_i^*} - 1}{\log_2(i+1)}$$

which is a percentage to identify how much the model's ranking of songs stack up against the users ranking of the same ten songs. These percentages are calculated for the users items, all the users cold items, and all the users warm items.

4.2 Apple vs Spotify

We note a few observations through from our best stats dataset

- The differentiating factor between the apple music and spotify datasets were that Apple Music had about a little less than double the amount of relevant (5 interactions minimum) items present compared to Spotify. (6789) vs (3628) This points to the idea that by increasing dataset size doesn't increase general Hit rate as Apple Muisc was worse on the first 2 models but when ran on the better performing model the possible Hit Rate the dataset could achieve was higher.

Category of Best	Best HR@10	Best NDCG@10
Apple (song name):	3.170	1.46
Apple (song summary):	6.221	2.903
Apple (song lyrics):	17.667	8.549
Spotify (song name):	12.291	6.077
Spotify (song summary):	12.139	6.025
Spotify (song lyrics):	15.630	6.914

Table 1: Apple and Spotify Best stats

Epoch	Summary		Name	
	HR@10	NDCG@10	HR@10	NDCG@10
0	0.455	0.167	1.81	1.225
20	5.462	2.52	1.710	0.812
40	4.09	2.04	1.674	0.822
60	5.463	2.339	1.567	0.736
80	5.463	2.395	1.425	0.640
100	4.704	2.426	1.639	0.730

Table 2: Apple with song name and summary

Epoch	Summary		Name	
	HR@10	NDCG@10	HR@10	NDCG@10
0	6.069	3.537	6.221	3.632
20	9.408	4.511	9.408	4.480
40	10.318	4.332	10.167	4.346
60	9.863	4.365	10.015	4.372
80	10.622	4.761	10.166	4.655
100	11.381	5.050	10.774	5.103

Table 3: Spotify with song name and summary

Epoch	Apple Lyrics		Spotify Lyrics	
	HR@10	NDCG@10	HR@10	NDCG@10
0	5.281	2.082	6.221	2.887
20	9.114	4.126	14.871	6.615
40	7.525	3.472	10.015	4.230
60	7.735	3.503	10.167	4.552
80	7.993	3.587	10.470	4.822
100	8.062	3.618	10.773	5.022

Table 4: Apple and Spotify lyrics comparison

- The hit rate and the Normalized Discounted Cumulative Gain are linearly proportional in all examples
- The best hit rate isn't usually best at the end of a predetermined number as the best Hit rate was usually some point in the middle of the epochs (supporting the idea of early stopping) [29]

4.3 Model performance

While analyzing the Hit Rate of the three models it becomes easily apparent that the best performing model is the truncated lyrics model as it exceeds on all levels, and while on the Spotify dataset the Song name model and the Summary model provide similar results, we believe summary model is the better performing model. Firstly the summary model on the Apple dataset achieved over double the Hit Rate of the Song model and while it did do slightly worse on the Spotify dataset we attribute that to the insufficient size of the dataset to preperly be trained on which is why the Apple music dataset which is significantly larger than Spotify saw better results in terms of the summar model.

4.4 Summary vs Truncated Lyrics

Our data quite favorable points to the idea that while Summary may be an improved feature compared to the Song Name a far superior feature compared to both would be utilizng truncated lyrics. Despite the fact many might assume the Summary model should've performed better there is a list of notable reasons our team believes could've caused this result.

- As the summaries were generated by the same LLM, the LLM will phrase its summaries in a very similar manner even if the summaries have no relation which can cause inaccuracy and confusion for the model.
- The summaries weren't extremely in depth summaries as they were limited to 25 words and given that the summaries were generated by feeding the song and artist name it's possible that the LLM could incorrectly summaries specific songs, which may even hurt the LLM.
- Additionally as we did the truncating utilizing the first section of the song, there's a chance the truncated lyrics formed connections it wasn't supposed to simply because the beginning section was similar even if the rest of the song wasn't related to that area.

The improvement seen in the truncated lyrics model points to the benefits of being incredibly accurate with your descriptions with as much information as possible since its that was likely a major contributor in the success of the model.

5 Conclusion and Fairness

In the future a few avenues that could be taken would be to have a large variety of different model sizes as our results lead could lead one to the conclusion that dataset size has a very big affect on how much affect changing the features of

the datasets can have. Also to improve the LLM model future researchers could utilize song lyrics to get the song summary as opposed to the song name and artist name as that would likely make the summaries more accurate. One final big avenue that could be investigated is the aspect of fairness in our model investigating fairness in this ml model to make sure certain user groups or locations don't get improperly advantaged in the model [4]. One specific direction that could be taken is watching artist popularity to ensure popular artists aren't being disproportionately advantaged, or diving deep into the cold and warm item data to ensure an equal proportion of both times appear in the recommendation.

Acknowledgments

This research was funded in large part due to the financial and educational support of the University of Southern California (USC), the National Action Council for Minorities in Engineering (NACME), and Apple Inc. Their contributions were instrumental in the success of this research.

References

Jin, H. (2024) A comprehensive survey on process-oriented automatic ..., arXiv. Available at: https://arxiv.org/pdf/2403.02901 (Accessed: 01 August 2024).

Taief, A. (2024) Application of LLMs and Embeddings in Music Recommendation Systems, Munin. Available at: https://munin.uit.no/bitstream/handle/10037/34168/thesis.pdf?sequence=2&isAllowed=y (Accessed: 01 August 2024).

Wang, Y. (2023) A Survey on the Fairness of Recommender Systems, ACM Digital Library. Available at: https://dl.acm.org/doi/pdf/10.1145/3547333 (Accessed: 31 July 2024).

Jeckmans, A. (2013) Privacy in Recommender Systems. Available at: https://ris.utwente.nl/ws/portalfiles/portal/5352108/Privacy_in_Recommender_Systems.pdf (Accessed: 01 August 2024).

Yang, J. (2020) Mixed Negative Sampling for Learning Two-Tower Neural Networks in Recommendations. Available at: https://yangli-cs-ucsb.github.io/paper/vldb13_li.pdf (Accessed: 01 August 2024).

Seshadri, P. (2023) Leveraging Negative Signals with Self-Attention for Sequential Music Recommendation, arXiv. Available at: https://arxiv.org/pdf/2309.11623 (Accessed: 01 August 2024).

Trainor, A. (2023) Popularity Degradation Bias in Local Music Recommendation, arXiv. Available at: https://arxiv.org/pdf/2309.11671 (Accessed: 01 August 2024).

Chen, Y.-X. (2010) How Last.fm Illustrates the Musical World: User Behavior and Relevant User-Generated Content, Medien. Available at: http://www.medien.ifi.lmu.de/pubdb/publications/pub/chen2010VISSW2/chen2010VISSW2.pdf (Accessed: 01 August 2024).

Bischoff, K. (2008) Can All Tags be Used for Search?, ACM Digital Library. Available at: https://people.uta.fi/~kostas.stefanidis/dbir16/papers/bischoff08.pdf (Accessed: 01 August 2024).

Ko, H. et al. (2022) A survey of recommendation systems: Recommendation models, techniques, and application fields, MDPI. Available at: https://www.mdpi.com/2079-9292/11/1/141 (Accessed: 01 August 2024).

Geetha, G. (2018) A hybrid approach using collaborative filtering and ..., IOPscience. Available at: https://iopscience.iop.org/article/10.1088/1742-6596/1000/1/012101/pdf (Accessed: 01 August 2024).

Shen, J. and Zheng, B. (2009) Solving the cold-start problem in recommender systems with social tags, Europhysics Letters. Available at: https://iopscience.iop.org/article/10.1209/0295-5075/88/28003 (Accessed: 01 August 2024).

Srinivas, S. (2017) Training Sparse Neural Networks, OpenAccess. Available at: https://openaccess.thecvf.com/content_cvpr_2017_workshops/w4/papers/Srinivas_Training_Sparse_Neural_CVPR_2017_paper.pdf (Accessed: 01 August 2024).

Sindhwani, V. (2010) Recommender Systems, Semantic Scholar. Available at: https://www.vikas.sindhwani.org/recommender.pdf (Accessed: 01 August 2024).

Kanchana1990 (2024) Song dataset: 10,000 Apple Music tracks, Kaggle. Available at: https://www.kaggle.com/datasets/kanchana1990/apple-music-dataset-10000-tracks-uncovered (Accessed: 01 August 2024).

MaharshiPandya (2022) Spotify tracks dataset, Kaggle. Available at: https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset (Accessed: 01 August 2024).

API Docs (2024) Last.fm. Available at: https://www.last.fm/api (Accessed: 01 August 2024).

Setiowati, S., Adji, T. B., and Ardiyanto, I. (2018) Context-based awareness in location recommendation system to enhance recommendation quality: A review, 2018 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, pp. 90-95. doi: 10.1109/ICOIACT.2018.8350671

Kotsogiannis, I. (2017) Directed edge recommender system, ACM Digital Library. Available at: https://www.cs.uic.edu/~elena/pubs/kotsogiannis-wsdm17.pdf (Accessed: 01 August 2024).

Morris, J. (2019) CompHD: Efficient Hyperdimensional Computing Using Model Compression. NSF. Available at: https://par.nsf.gov/servlets/purl/10237210 (Accessed: 01 August 2024).

Kooi, J. (2022) Beating Spotify's Algorithm: Towards an Improved Emotion Label for Billboard Songs, Student Theses, Google. Available at: https://scholar.google.com/scholar_settings?sciifh=1&hl=en&as_sdt=0%2C5 (Accessed: 01 August 2024).

Ji, Z. (2021) Early-Stopped Neural Networks Are Consistent, Proceedings NeurIPS. Available at: https://proceedings.neurips.cc/paper/2021/file/0e1ebad68af7f0ae4830b7ac92bc3c6f-Paper.pdf (Accessed: 01 August 2024).

Appendix

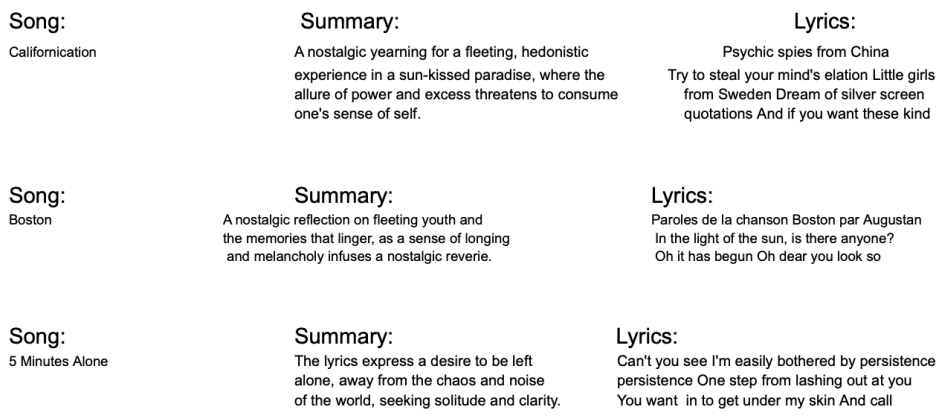

Figure 1: Image of song + summary + lyrics.