Musical Semantic Embeddings (MuSE)

CS 229: Machine Learning

Eric Lee
Department of Computer Science
Stanford University
ericlee7@stanford.edu

Akshar Sarvesh
Department of Computer Science
Stanford University
asarvesh@stanford.edu

Abstract

Personalized music streaming services require computationally efficient and scalable recommendation systems, but traditional methods relying on audio feature extraction often face scalability challenges. Thus, this paper presents Musical Semantic Embeddings (MuSE), a novel approach inspired by NLP models like GloVe, to generate song embeddings using only playlist data. By leveraging a global co-occurrence matrix of songs, MuSE captures semantic relationships without requiring audio analysis. Enhancements such as Laplace smoothing, artist playlist augmentation, and contrastive learning improve embedding robustness and quality. Experiments highlight MuSE's effectiveness, achieving strong classification metrics and consistent embedding clusters. This approach offers a scalable and accurate solution for modern music recommendation systems.

1 Introduction

With the rise of personalized music streaming platforms like Spotify and Apple Music, song recommendation systems have become integral to user experience, driving engagement and user satisfaction. However, current recommendation systems are often computationally expensive, requiring complex models and extensive feature extraction for each individual song. In this paper, we introduce Musical Semantic Embeddings (MuSE), a novel approach inspired by NLP models such as GloVe, to generate embeddings for songs using playlist data alone. Given a set of playlists, our method generates a unique embedding for each song, such that similar songs are close in the embedding space. We propose that these embeddings can be effectively utilized for downstream tasks central to music platforms, such as song recommendation and user profile evaluation.

2 Related Work

Song recommendation systems have traditionally relied on computationally intensive models that extract high-dimensional features from audio data using techniques like recurrent [Sun, 2022] and convolutional neural networks [Yang, 2022]. While effective, these methods are resource-intensive and challenging to scale across the enormous catalogs of today's music streaming platforms.

Inspired by natural language processing (NLP), our approach simplifies embedding generation by leveraging playlist data alone, similar to how NLP techniques create embeddings for words in text. This lightweight method uses co-occurrence patterns of songs in playlists to produce embeddings that can power downstream tasks without requiring feature extraction from audio content. Initially, we explored Word2Vec [Mikolov et al., 2013], which embeds songs based on their neighbors within playlists. Applying Word2Vec's continuous bag-of-words model has been seen in prior work, such as Song2Vec [Tong, 2018], attempting to capture semantic relationships between adjacent songs. However, Word2Vec's reliance on localized context limits its effectiveness for playlists, where song order is often arbitrary due to features like "Shuffle."

To overcome these limitations, we propose a GloVe-based approach [Pennington et al., 2014], which uses a global co-occurrence matrix to more appropriately encode relationships between songs. Unlike Word2Vec, GloVe emphasizes overall membership in playlists rather than localized context, making it more suited for large-scale music recommendation. Additionally, we explored techniques such as data augmentation and contrastive learning [Khosla et al., 2021] to improve embedding robustness and capture additional information from our song-playlist membership data.

3 Dataset and Features

We sampled 527 playlists containing 21747 unique songs from a Spotify Playlists dataset [Shkurenko, 2022] found on Kaggle. The dataset includes information about playlist names, song names, song artists, and maps each playlist to the list of songs it contains. To avoid an excessively sparse co-occurrence matrix, we sampled the 527 playlists containing at least one of 10 songs that were found across many playlists. Given the straightforward nature of this membership data, we performed no additional pre-processing. For fast querying, we stored the data in a MySQL database.

We split our data into training, validation, and test sets across playlists distributed as 80%-10%-10%, respectively. This practice allows us to evaluate generalizability on the test set while tuning hyperparameters on the validation set, such as our learning rate, scaling term α, number of epochs, and embedding dimension size.

4 Method

To generate robust embeddings, we implemented the following machine learning techniques:

1. **Global Co-Occurrence Matrix:** We constructed a co-occurrence matrix M based on the presence of songs in playlists. Specifically, if songs i and j appeared in λ playlists together, they would have a co-occurrence score $M_{(i,j)} = \lambda$. Note that for S unique songs, $M \in \mathbb{R}^{S \times S}$. From this matrix, we computed the weighted least squares objective and optimized to create initial embeddings.
2. **High Co-occurrence Scaling:** With precedent taken from the GloVe [Pennington et al., 2014], we included a scaling term ($\alpha < 1$) to reduce the high weight of extremely high co-occurrence values for balance across other pairs. Namely, for co-occurrence $P_{(i,j)}$:

$$\lambda_{(i,j)} \leftarrow \left(\frac{\lambda_{(i,j)}}{\lambda_{\text{max}}}\right)^{\alpha}$$

   Unlike GloVe, we did not treat songs as "context" or "main" songs because in this application regarding playlists, the order of songs is not critically important.
3. **Laplace Smoothing:** We found better results by implementing Laplace smoothing, which involves incrementing all co-occurrence values by 1:

$$M_{(i,j)} \leftarrow M_{(i,j)} + 1, \forall i, j \in [S]$$

   This helps reduce the sparsity of our co-occurrence matrix, and helps reduce over-fitting of the most frequent co-occurrences by placing a small amount of importance even on rare song pairs.
4. **Optimization and Gradient Descent:** We used gradient descent to minimize the following objective by taking its gradient:

$$J = \sum_{i,j}^{n} \left(\frac{M_{(i,j)}}{M_{\text{max}}}\right)^{\alpha} (w_i^T w_j + b_i + b_j + \log M_{(i,j)})^2$$

   Here, w_i is the embedding of song i, b_i is the bias of song i, and α is as defined above. J is optimized with batch gradient descent, with stopping upon convergence ($< 10^{-5}$) or maximum iterations (specified as hyperparameter).
5. **Artist Playlist Augmentation:** To account for the musical similarity of songs by the same artist, we augmented our dataset with playlists consisting exclusively of songs from

individual artists. After creating one such playlist for each artist, we add these augmented playlists to the training set such that the co-occurrence matrix will reflect the relationships between these songs more clearly. Namely, M follows:

$$\Big(M_{(i,j)} \leftarrow M_{(i,j)} + 1\Big) \text{ if Artist}(i) = \text{Artist}(j) \text{ and } i \neq j, \forall i, j$$

6. **Constrastive Learning from NLP:** To enhance performance on downstream classification tasks (e.g., song similarity), we incorporated contrastive learning to refine our embeddings [Khosla et al., 2021]. To maintain a lightweight approach using only accessible playlist data, we introduced a novel method leveraging LLM-generated embeddings of playlist titles to construct positive and negative song pairs. Specifically, we created an embedding representation for each playlist by tokenizing its title and averaging BERT embeddings [Devlin et al., 2019] across tokens. Then, positive song pairs were sampled from playlists with the highest cosine similarity between their embeddings, while negative song pairs were drawn from playlists whose embeddings had the lowest similarity.

   We then optimized loss function J, where w_i, w_j are MuSE embeddings for songs i, j, $y \in \{-1, 1\}$ is the label denoting a positive (1) or negative (-1) example, and margin m is a hyperparameter enforcing a lower bound on the distance between negative pairs:

$$J = \sum_{\{(w_i, w_j, y)\}} y \cdot \Big(1 - \frac{w_i^T w_j}{||w_i||_2 \cdot ||w_j||_2}\Big) + (1 - y) \cdot \max\Big(0, \frac{w_i^T w_j}{||w_i||_2 \cdot ||w_j||_2} - m\Big)$$

   Intuitively, optimizing J pushes the embeddings of positive examples towards each other and pushes negative examples away from each other, maintaining a margin of at least m.

5 Experiments

To evaluate these song embeddings at the end of training, we opted to implement both quantitative evaluation of the model's ability to classify song similarity and qualitative evaluation of proximity in the embedding space.

- **Quantitative Evaluation:** We assessed model performance in classifying whether labeled song pairs are similar or dissimilar across 4 metrics: accuracy, precision, recall, and F1 score. Noting that MuSE is likely to be used for recommendation systems, our priority was to maximize precision and F1 score. We generated these labeled song pairs from our validation and test sets. In particular, positive examples were generated by sampling pairs in the same playlist and negative examples were created by sampling songs not co-occurring in any playlist. Labels were predicted by simply taking the cosine similarity between learned song embeddings. We considered song pairs with absolute value cosine similarity > 0.05 as a minimum threshold for a meaningful recommendation.
- **Quantitative Evaluation:** To evaluate whether the embeddings for similar songs led to appropriate clusters, we used t-SNE to reduce the embeddings to two dimensions and plotted sample songs from well-known artists. This allowed us to visually assess whether songs that are known to be similar (ex. Applause and Paparazzi by Lady Gaga) have embeddings which are close in N-dimensional space. We chose t-SNE over PCA due to its better ability to deal with high dimensional data like we have in our MuSE embeddings.

We began by training a basic MuSE model to generate baseline embeddings (Method 4.2). Then, for hyperparameter tuning, we experimented with the scaling constant $\alpha \in \{0.6, 0.75, 0.9\}$. Our findings were consistent with the initial GloVe experiments [Pennington et al., 2014], where $\alpha = 0.75$ yielded the best results in striking a balance between reducing over-fitting on high frequency pairs and minimizing noise from infrequent pairs.

Then, we conducted grid search with $\alpha = 0.75$ to determine our remaining hyperparameters. Specifically, we evaluated 12 combinations between the number of epochs $N \in \{50, 100, 150, 200\}$ and learning rate $\eta \in \{0.025, 0.05, 0.1\}$. After evaluation on the validation set, we found the highest F1 score on the hyperparameters $N = 150$ and $\eta = 0.05$. To train our model, we performed batch conventional gradient descent, with an early stopping convergence threshold of 10^{-5}.

Table 1: Classification metrics for experiments with different dimensions on validation set.

Embedding Dimension	Accuracy	Precision	Recall	F1
25	48.54%	52.75%	51.61%	52.17%
75	53.30%	57.45%	63.53%	60.34%
150	51.13%	53.42%	55.71%	54.55%
250	60.15%	68.82%	72.73%	70.72%
500	**63.21%**	**75.32%**	**74.36%**	**74.83%**

We separately determined the hyperparameter of embedding length, due to its importance in determining the information captured within the embeddings. As seen in Table 1, we found that in general, the model with embedding dimension 500 showed the best performance across all metrics. Further, performance generally seemed to improve as the embedding dimension increase. This likely occurs because the longer embeddings are able to more deeply capture the complex relationships between songs, though at the risk of over-fitting with the additional parameters.

We also discovered for all models that the inclusion of Laplace smoothing uniformly improved performance across all metrics. This improvement can be attributed to the reduction of noise from extremely rare song pairs, whose song embeddings capture little information without smoothing.

6 Results and Discussion

After tuning hyperparameters, we trained models for each combination of implementing artist playlist augmentation (Method 4.5) and contrastive learning from NLP (Method 4.6). Then, to assess the impact of these additions, we evaluated each finalized model on the test set (Table 2).

Table 2: Classification metrics for final models on test set with dimension 500.

Model	Accuracy	Precision	Recall	F1
MuSE	65.30%	71.79%	82.35%	76.71%
MuSE + Artist Augmentation	72.22%	77.92%	82.19%	79.98%
MuSE + Contrastive Learning	69.61%	74.07%	**85.71%**	79.47%
MuSE + Augmentation + Contrastive	**74.55%**	**79.75%**	84.00%	**81.82%**

Based on these metrics, we observe that including both playlist augmentation and contrastive learning results in the best trained embeddings. Adding playlist augmentation to the model leads to improvements in accuracy and precision with a slight drop in recall, as embeddings distinguish between different artists more clearly and err on the side of classifying false negatives. Meanwhile, adding contrastive learning leads to measured improvements across all metrics, implying success in pulling similar embeddings closer while driving dissimilar embeddings apart. The final model with both of these features seems to combine the benefits of both individual additions, with high performance across all metrics by the most significant margin.

As seen in Figure 1, the qualitative results also support the conclusion that using both contrastive learning and playlist augmentation generates the best performing model with regard to the embedding space. Augmentation seems to create more well-defined clusters at the expense of distance, as seen by the grouping of colors being in similar regions but being somewhat further apart. Meanwhile, contrastive learning clearly pulls similar embeddings closer together, to an extent determined by margin m. In the final combined model, we see the best of both features where clusters are both well-defined and relatively tight.

Moreover, beyond groupings by artists, note that the embeddings for the combined model (bottom right) encapsulate distinctions in audio features and genre despite not having trained on such information. Specifically, rap/hip-hop songs are in the top left, and dance music aggregates toward the bottom left. Pop music clusters on the right side of the plot, with more energetic pop music near the bottom and increasingly sad / relaxed songs near the top of the plot.

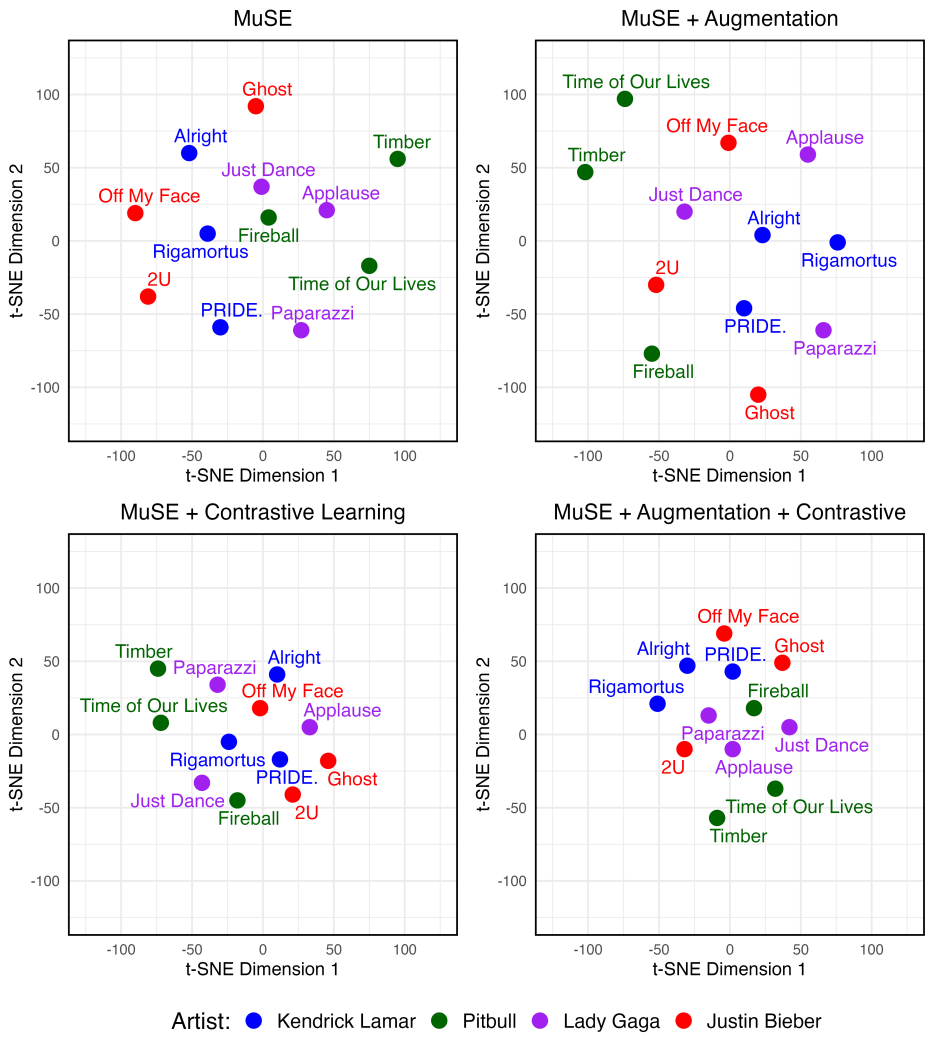

Figure 1: Two dimensional visualizations of similar songs in embedding space (reduced by t-SNE) for each model with embedding length 500.

7 Conclusion

In this paper, we introduced Musical Semantic Embeddings (MuSE), a novel machine learning technique designed to efficiently generate meaningful song embeddings using playlist data alone. By leveraging a global co-occurrence matrix to map songs into an embedding space, MuSE eliminates the need for feature extraction from audio data. Our findings demonstrate that MuSE effectively groups similar songs in the embedding space, with techniques like playlist augmentation and contrastive learning. These advancements highlight MuSE's potential as a computationally efficient solution for recommendation tasks, enabling platforms like Spotify and Apple Music to continuously refine embeddings with minimal overhead. The ability to train embeddings using only playlist metadata represents a significant step toward faster, scalable music recommendation systems.

Despite its advantages, we note that MuSE faces limitations in generalizing to unseen songs, particularly those with limited playlist representation. While playlist augmentation helps mitigate this challenge, future research could explore complementary strategies to address this limitation. Leveraging NLP techniques to analyze song titles or training a neural network atop the existing embeddings to predict unseen embeddings are promising directions. Such extensions would broaden MuSE's applicability to a wider range of songs, including less popular and newly released tracks.

8 Contributions

- Eric and Akshar worked together to ideate, read papers, and conduct background research to formulate the project methodology.
- Eric processed the dataset, splitting it to generate training, validation, and test sets.
- Akshar implemented the fundamental training framework with gradient descent.
- Akshar and Eric jointly worked on training models in parallel, and Eric led hyper-parameter tuning with grid search.
- Eric implemented the artist playlist augmentation feature to modify the co-occurrence matrix.
- Akshar implemented contrastive learning with the generation of positive/negative pairs and additional training with contrastive loss function.
- Eric implemented the evaluation framework with qualitative evaluation with clustering and quantitative evaluation on labeled positive and negative song pairs.
- Akshar led in-depth results analysis and error analysis after training to evaluate model performance.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2019. URL https://arxiv.org/abs/1810.04805.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. 2021. URL https://arxiv.org/abs/2004.11362.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013. URL https://arxiv.org/abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *Stanford Computer Science*, 2014. URL https://aclanthology.org/D14-1162/.

Viktoriia Shkurenko. 6k spotify playlists. *Kaggle*, 2022. URL https://www.kaggle.com/datasets/viktoriiashkurenko/278k-spotify-songs/data?select=final_playlists.csv.

Pengfei Sun. Music individualization recommendation system based on big data analysis. *Computational Intelligence and Neuroscience*, 1, 2022. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC9276508/.

Weiqi Tong. From word2vec to song2vec: An embedding experimentation with context learning of dense vectors. *Medium, Weiqi Tong*, 2018. URL medium.com/@weiqi_tong/from-word2vec-to-song2vec-an-embedding-experimentation-9215279c9d7a.

Jingzhou Yang. Personalized song recommendation system based on vocal characteristics. *Mathematical Problems in Engineering*, 2022. URL https://doi.org/10.1155/2022/3605728.