Multimodal Fusion Genre-Based Music Recommender

Daniel Öman
Georgia Institute of Technology
daniel.oman@gatech.edu

Wonho Choi
Georgia Institute of Technology
wchoi323@gatech.edu

Abstract

Music is presented in multiple modes of data for the user. The user understands music not only based off of its sound, but also its lyrics along with the context of the song. Therefore, recommending this music requires an understanding of all of these different dimensions of data. In this study, we propose a deep neural network content-based fusion model to predict and recommend music using a comprehensive method using details of the song. Using 6 training classes in the classification task for the DNN, we achieve a 56.7% prediction accuracy but beat baseline recommendation metrics with a mean average precision (mAP) of 0.04323. We demonstrate that reducing the number of training classes forces higher model accuracy, but lower recommendation quality. For instance, with 3 training classes, we reach 77.9% prediction accuracy, but yield an mAP of 0.02263.

1. Introduction

Music is an expressive content that, as a form of data, enables various insights into people's moods and behavior. Our project aims to predict and recommend songs for listeners in playlists using multimodal data. Using the details of the songs along with the users' music listening habits, this model will output recommendations and playlists for the user. We used a data set with user listening histories and one million songs. Afterwards, we combine both existing pre-trained and new neural networks for each mode of input data. Our model performs this combination through an alignment between the shared space of these features. Therefore, aligning the different factors in the given multimodal data, our model will produce a set of recommendations that encompasses understanding and prediction of these data.

2. Related Work

There are three related works that we referenced in our architecture for this project.

The paper "Deep Learning Recommendation Model for Personalization and Recommendation Systems" from Facebook researchers introduces an open-source Deep Learning Recommendation Model (DLRM) using different embedding schemes to process categorical data with multi-layer perceptrons [4]. Specifically, this is a model that inspires our recommender system.

Moreover, Spotify, as a music streaming company with large amounts of music data, provides insight in the factors taken into account for a recommendation model. In the paper "Personalized Audiobook Recommendations at Spotify Through Graph Neural Networks," Spotify explains its alignment of user preferences and audio while maintaining low latency and efficient computing [2].

Lastly, in the NeurIPS paper "Deep content-based music recommendation," the authors propose a neural network model that uses our dataset to provide different deep learning based approaches to learn latent representations of songs. For our project, we intend to restrict the representation to that of a genre-based encoding, but we use their results as a baseline for our end-to-end recommender system's performance [5].

3. Technical Approach

We built a music recommendation system trained on a multimodal set of datasets, taken from the Million Song Dataset (MSD) [1]. From this source, we draw song-level metadata and lyrical data. Unfortunately, due to enterprise-only API access restrictions from 7digital,[1] the audio snippet provider service, we were not able to extract audio snippets to incorporate in the model architecture. However, the proposed methodology still implements a fusion model, although missing this data source. At a high level, we leverage a fusion model, with input components trained on each dataset individually to generate embeddings for each modality, which processes them through a "fusion step", and train a deep neural network with weighted cross-entropy loss on the fused embeddings to generate genre predictions. We use the fusion model to generate genre cluster

[1] https://us.7digital.com/

embeddings for each user, and rank songs in the test set by dot product similarity in the genre embedding space. The top songs in the user's ranking are the song recommendations.

3.1. Preprocessing

A significant challenge in the model training is the existence of ground-truth labels for the DNN multi-class classification stage of our fusion model. The MSD provides song-level lists of human-labeled genre tags, supplied by MusicBrainz.[2] The goal is to convert this multi-label classification problem to a multi-class classification problem so that all songs can be concisely represented in a lower dimensional feature space. Let n be the number of data points (songs) in our dataset. For each $i \in \{1, \ldots, n\}$, we have a list of training tags $\tau^i \subseteq \mathbf{T}$, where $\mathbf{T}$ is the set of all possible tags. For instance, the song with MSD id `SOVVDCO12AB0187AF7` (say it exists at index i) has

$$\tau^i = \{\text{fusion}, \text{jazz fusion}, \text{classic pop and rock}\}.$$

As the title implies, out of the 1 million songs in the MSD, there are $n = 116,479$ songs with tags. This subset represents the parent set of our train/test split. Out of these songs, there are 1,859 unique labels (that is, $|\mathbf{T}| = 1,859$). The total number of unique descriptions per each song is $3,063$ – that is, if $\mathcal{T} = \{\tau^i : i \in \{1, \ldots, n\}\}$ is the set of all distinct tag lists in the dataset, then $|\mathcal{T}| = 3,063$). If we were to use these $3,063$ unique descriptions as our training classes, our model would underfit, as a direct result of having too many training classes for the multi-class genre classification problem. To overcome this problem, we implement a genre-list embedding and clustering scheme to reduce the number of classes to a reasonable amount. At a high level, we intend to learn a function $f : \mathcal{T} \to \mathbb{R}^m$ for an embedding dimension m, and perform a clustering algorithm (K-Means) on these embeddings to reduce the number of training classes to k where, $k << |T|$. Then, we use these cluster labels as the training classes for the multi-class classifcation problem. By reducing the dimensionality of the training class space, our fusion model's classification step will be much less likely to underfit. This essentially groups genre tags into broader "genre categories" vector representations, which the recommendation algorithm component later uses to characterize songs and users to compute similarity scores for ranking purposes. There are two techniques we use to reduce dimensionality of the label space: one baseline method, and another deep-learning based approach.

[2] https://musicbrainz.org/

3.1.1 Multihot Genre Tag Embedding

An initial and baseline embedding technique is to generate a multi-hot encoding for song's genre tags. We define the function $f_{\text{multihot}} : \mathcal{T} \to \mathbb{R}^m$ as $f(\tau)_j = 1$ (jth entry of τ) if $T_j \in \tau$, and 0 otherwise, for all $\tau \in \mathcal{T}$ and for all $j \in \{1, \ldots, |T|\}$ – that is given that $\mathbf{T}$ is an ordered set that we can index into. Note that for this function to be fully defined, $m = |\mathbf{T}|$, and so in our MSD subset, our function f_{multihot} maps to a vector with $|\mathbf{T}|$ entries.

3.1.2 Word2Vec Genre Tag Embedding

We use the pre-trained Word2Vec deep-learning model with the skipgram architecture to learn embeddings for each individual genre tag per song, then aggregate the embeddings (our specific approach uses average aggregation) to create a per-song genre embedding in space $\mathbb{R}^m$, where m is a hyperparameter to our model. Our experimentation uses $m = 100$ as the embedding size. Word2Vec leverages a 2-layer neural network to position words with similar meanings and contexts close in a dense vector space. So, to generate embeddings, we employ the following routine: For each tag $t \in \mathbf{T}$, learn an embedding $\text{Word2Vec}(t \mid \mathcal{T}) \in \mathbb{R}^m$. We use the pre-trained model and treat each element in $\mathcal{T}$ as a "sentence" to provide context for each word to embed. Then, define the following embedding-generation function $f_{\text{w2v}} : \mathcal{T} \to \mathbb{R}^m$: for each $\tau \in \mathcal{T}$,

$$f_{\text{w2v}}(\tau) = \frac{1}{|\tau|} \sum_{j=1}^{|\tau|} \text{Word2Vec}(\tau_j \mid \mathcal{T}).$$

This computes the average Word2Vec embedding of all tags in the tag list per song.

3.1.3 Embedding Technique Tradeoffs

The benefits of using f_{multihot} as our label embedding function is that it is computationally efficient and captures the similarity of songs with overlapping genre labels well when clustering with K-Means. However, it fails to capture the relationship between the elements of $\mathbf{T}$, since it relies entirely on the position of the element within $\mathbf{T}$. Furthermore, f_{multihot} maps tag lists to a very high dimensional vector space, which is detrimental when clustering with K-Means.

On the other hand, using f_{w2v} is computationally more expensive, but allows us to better represent similar lists of genres, resulting in stronger clustering.

3.1.4 Metadata Feature Embedding

Song-level metadata exists in the form of song title, release (album) title, artist name, release year, duration, an artist popularity metric provided by MSD, and release year.

For the song title, album title, and artist name, a pretrained Word2Vec model was used to generate embeddings for each category, fine-tuned with the vocabularies formed by each domain. For example, Word2Vec was fine tuned using only song titles, without considering album titles or artist names in the embedding generation. These embeddings are generated in a 100-dimensional space by default, which was the size used in the model architecture. The output metadata vector is a concatenated vector of all component embeddings.

3.1.5 Lyrical Feature Embedding

For the derivation of features from the lyrical data of the songs, we use term frequency-inverse document frequency (TF-IDF) weighted averaging and then process the given matrix into a pre-trained RoBERTa model. Specifically, the TF-IDF processing is done through the TfidfVectorizer from the scikit-learn library, which returns a vector of the features based on the TF-IDF score of each word. Given that the lyrics are presented as a frequency table of the words, TF-IDF effectively calculates its significance due to its ability in leveraging both the frequency of the term t in the single document along with its rarity across the documents. It is calculated through a method that multiplies the two metrics as shown below, where t is the term, d is one document, and D represents all the documents.

$$\text{TF} - \text{IDF}(t, d, D) = \text{TF}(t, d) \cdot \log\left(1 + \frac{\text{average}(t, D)}{\text{freq}(t)}\right)$$

Through this method, we determine the weight of the words in the lyrics then process these weights into a given number of features in the vector.

The secondary step of the embedding is passing in the TF-IDF significance vectors into a pre-trained RoBERTa model. We used the Hugging Face transformers library to call the base RoBERTa model and tokenizer. We process the word weights from the TF-IDF by passing in each word in the format of "word:count" and generating CLS classifications tokens using RoBERTa. We also process the words in batches to reduce compute load. Using this method, we effectively generate feature embeddings from the lyric significance and frequencies.

3.2. Fusion Model Architecture

Figure 1 describes the high-level fusion model architecture that we implement. The chosen fusion step is concatenation, in which the output embedding representations are concatenated. Other options for fusion are element-wise operations, such as addition, multiplication, or min/max/mean pooling. In the fusion step, different embedding outputs may have different dimensionalities. Using element-wise operation, a fusion step may pad or truncate vectors, potentially losing important features. This rationalizes the choice to use concatenation as the fusion operation.

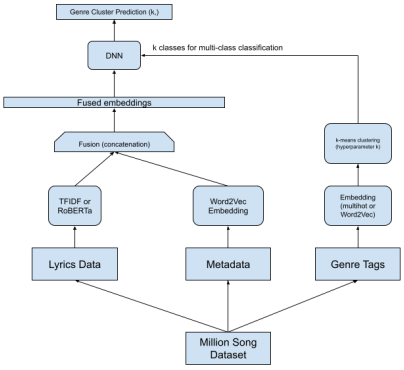

Figure 1. Fusion Model Architecture

The deep neural network has hidden sizes 256, 128, 128, 256, and 512, using fully connected layers with dropout probability p. Each fully connected layer is followed by a residual block to improve network stability to combat the vanishing gradient problem, with batch normalization to improve stability. The rectified linear unit (ReLU) is used as the activation function between layers. The output size is k, the number of chosen classes after clustering, which passed through the softmax function to generate probabilities for each class prediction. This architecture takes inspiration from Jafar and Lachir [3] which employs a similar fusion technique across modalities, which then passes output features through a deep neural network (DNN) to generate predictions. The DNN's residual blocks are illustrated in figure 2.

The complete DNN architecture is illustrated in figure 3. The fusion model was implemented using the PyTorch framework, which allows for intuitive construction of this architecture.

3.3. Model Evaluation

The model is evaluated at multiple stages. First, we assess the label embedding and clustering task using the K-Means "elbow method," by calculating the sum of squared distances of embeddings to their nearest cluster center (Scikit-Learn's "inertia" metric).

The fusion model outputs are evaluated using pointwise accuracy with the cluster ground-truth labels from the test set. We evaluate the model's performance using its recommendation capabilities against a baseline from van den Oord et al. [5], comparing the mean average precision (mAP) of their recommendation outputs, using a fixed top 500 recom-

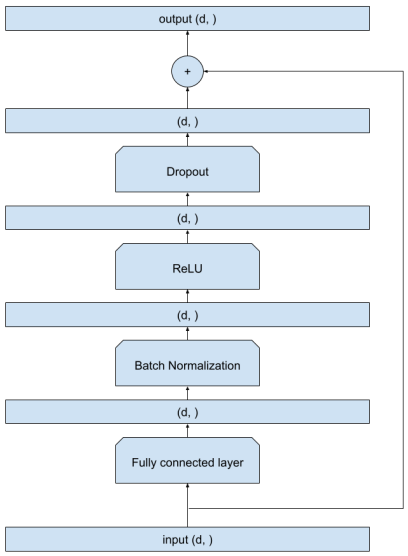

Figure 2. Residual block architectural component used in the DNN

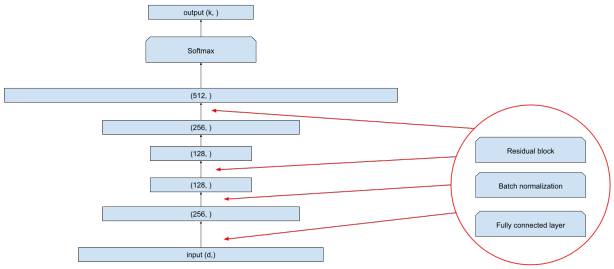

Figure 3. DNN architecture for the final fusion model component. The module circled in red is repeated between hidden layers.

mendations. The mAP is taken as a value from 0 to 1, where 0 indicates poor precision and 1 indicates perfect precision.

3.4. Inference and Recommendation

Suppose S is the set of all songs and we have k classes. Then, our fusion model is a function $F : S \to \mathbb{R}^k$ such that each vector in the output is a probability vector generated by softmax. User listening data is given in the MSD as triplets of user id, song id, and listen count. Suppose we have a set of users U and listening history set H compromised of triplets (u_i, s_i, c_i) where $u_i \in U$ is the user of the ith triplet, s_i is the song of the ith triplet, and $c_i \in \mathbb{Z}^{\geq 0}$ is the number of times u_i listened to s_i. Then for all $u \in U$, set $S_u \subseteq S$ be the songs in S listened to by u. Get all corresponding triplets (u, s_j, c_j) for all $s_j \in S_u$. For each triplet and fixed u, let

$$v_u = \frac{1}{|S_u|} \sum_{(u,s,c) \in H} cF(s),$$

i.e., the v_u is the weighted average genre prediction for all songs listened to by u. When implementing the model, we will ensure that each song in the chosen H is not in the training set. Each $v_u \in \mathbb{R}^k$. Then, for each song $s' \in S \setminus S_u$ (songs not listened to by u), such that s' is also not in the training set, compute $v_{s'} = F(s')$, and compute $v_u \cdot v_{s'}$. Rank the top songs s' in terms of this dot product, and the best K of them will be the recommendation to the users.

4. Data

We used the Million Song Dataset for our different modes of data. The Million Song Dataset is a dataset initially created by The Echo Nest, a music data platform, in conjunction with LabROSA, the Laboratory for Recognition and Organization of Speech and Audio from Columbia University. The dataset contains a public collection of multiple audio and song datasets, largely contributed by The Echo Nest. Sponsored by the National Science Foundation, the dataset was actually created under a grant from the NSF, project IIS-0713334. The main objective of the dataset is to provide a resource and encourage further research on algorithms in the music industry. Due to the lack of research provided in this area of algorithm development, the contributors aim to promote large scale development and research performed in the music industry. Moreover, for companies such as The Echo Nest that provide the song details in the dataset, the Million Song Dataset is a means to access their large scale datasets using an API.

For the composition of the dataset, the dataset contains instances of songs along with the corresponding features extracted along with details of the songs. It also contains user data regarding specific users along with their music tastes. For the actual songs, the dataset contains observable details such as the genre, date, lyrics, and audio of the songs, the dataset also contains inferred details such as a "similarity" tag and multiple genres. There are 1,000,000 total instances of all songs with different attributes of data, but there are 116,479 instances with tags with the genres of the songs that we actually used. Some songs did not have specific genre tags or did not have lyrics attributed to them. Moreover, there is a smaller subset of 10,000 songs that was used for experimentation. For the user data, there are 1,000,000 instances of unique users with given data. Due to the property of the dataset that it is a cluster of datasets, some complementary aspects of the dataset were from external links, such as the audio which was from a music service called 7digital. However, the main metadata along with lyrics were provided by companies that officially partnered with Million Song Dataset and could be directly downloaded.

The data was largely collected by The Echo Nest, which as a music intelligence service got access to the data using web scraping, data mining, and digital signal processing. The dataset was published in 2011, and it has been used

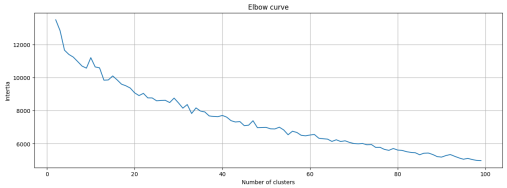

Figure 4. Clustering multi-hot encoded genre labels

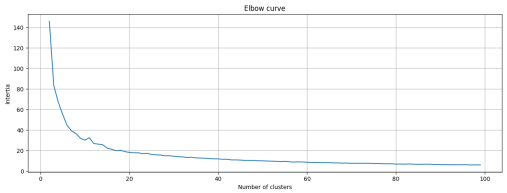

Figure 5. Clustering Word2Vec encoded genre labels

in multiple studies including one of the related works cited above, the "Deep content-based music-recommendation" from NeurIPS which we reference as a baseline later. [5].

5. Experiments and Results

We performed experimentation in the genre tag embedding task to transform the multi-label classification problem to a multiclass classification problem to identify the ideal number of training classes and clustering method for best model performance, as well as experimentation and evaluation for the fusion network and recommendation application relative to baseline models.

5.1. Genre Tag Embedding and Clustering Results

We computed embeddings for the list of genre tags per each song in the dataset to reduce the dimensionality of the training label space with K-Means clustering. As a baseline, we use multi-hot embeddings generated using the embedding function f_{multihot} and then ran K-Means (using the Scikit-Learn implementation) in $\mathbb{R}^{|\mathbf{T}|}$ space. Figure 4 is the elbow curve for the multi-hot embedding approach after running clustering from $k = 2$ to $k = 100$ clusters.

The deep learning approach uses Word2Vec, resulting in the embedding function f_{w2v}. Figure 5 is the elbow curve for the Word2Vec embedding approach.

This curve is much smoother than that of the multi-hot approach, with a clear elbow at $k = 10$ to $k = 20$, indicating an optimal cluster range. The multi-hot approach shows a rough elbow around $k = 50$. Since fewer clusters benefit training, Word2Vec significantly outperforms multi-hot, requiring 80% fewer labels. Additionally, inertia for multi-hot clusters is around 7,000, while for Word2Vec, it is about 20–a 99% improvement, highlighting Word2Vec's superiority in clustering strength and dimensionality reduction.

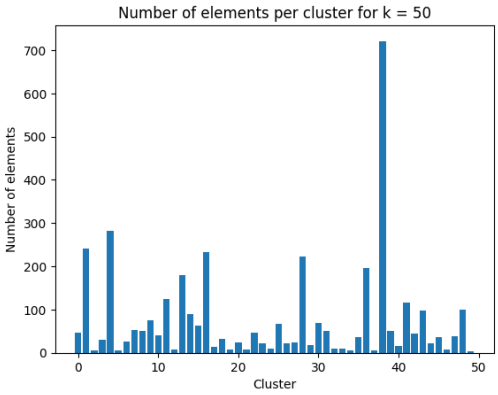

Figure 6. Multi-hot encoded cluster sizes

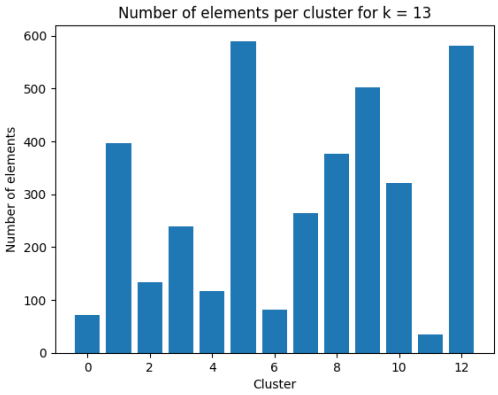

Figure 7. Word2Vec encoded cluster sizes

Further exploration was done specifically in $k = 50$ clusters for the multihot approach and $k = 13$ for the Word2Vec approach. Figures 6 and 7 show the number of elements per cluster.

In the multihot approach, cluster 38 encompasses roughly 25% of all elements. This behavior was experienced for most values of k tested in the multihot approach – that is, one cluster tends to collect a significant portion of all data points. This can imply that the fusion model classification tasks will be skewed towards predicting label 38 (in this instance of clustering). The Word2Vec cluster sizes are more uniformly distributed which will be beneficial when training the multimodal fusion model. Overall, with the given metrics and analysis, Word2Vec is the clear choice for generating ground-truth training classes.

5.2. Lyrical Data Embedding Results

As a means to measure the significance of each lyrical data present, we derive a feature embeddings with the given lyrics of the songs in the Million Song Dataset. Given the frequency table of different words present in the lyrics of each song, we compute a feature vector using term fre-

quency - inverse document frequency (TF-IDF), and then process the TF-IDF matrix using the pretrained word processing model RoBERTa to derive features from the most frequent words of the lyrics.

Initially, due to the nature of lyrical sentences and the presence of context within the lyrics, we were intending on only using a pre-trained word-processing model (e.g. RoBERTa) to successfully encompass the entire context of the words. However, through an attempt at processing the dataset using RoBERTa, this method required an overwhelming amount of compute power due to the size of the dataset. Moreover, given the dataset, the lyrics were presented as a frequency table of each word in the lyrics, rather than cohesive sentences or chunks. Therefore, a method to extract appropriate context from the large size of lyrics was passing in only the significant lyrics into the pre-trained model. Therefore, we used a TF-IDF to initially derive features and frequency from the large lyrics models, and then passed in the corresponding embeddings to the pre-trained RoBERTa. RoBERTa, based on Google's BERT model for language processing, is an optimized approach that also processes in larger batches. This proved sufficient due to the lack of presence in whole sentences but rather simply word weights and significant, so we proceeded with RoBERTa as a secondary step after the TF-IDF. Therefore, with the significant weights given for each frequent word of the lyrics, the RoBERTa was able to process the lyrics and the given contexts through the weight in which it appeared.

5.3. Fusion Model Training Configuration

We trained our fusion model using a 20% train/test split on the dataset. To train our model, we used weighted cross-entropy loss as the loss function and employed the Adam optimizer with learning rate η and dropout probability p as hyperparameters. We used weighted cross-entropy loss since the distribution of the training labels is not uniform across the train set. This allows for underrepresented training classes to have higher importance. On input, features are normalized using standard scaling from Scikit-Learn's preprocessing library. This is important since the dimensionality of the inputs is high, and the distribution of features varies significantly throughout the dataset. Additionally, principal component analysis (PCA) was applied to the test and train data to reduce dimensionality using the Scikit-Learn implementation, parametrized to retain 95% variance. This improves model accuracy and reduces underfitting. To accelerate training, batch training was used with a fixed batch size on a T4 GPU.

5.4. Fusion Model Results

To achieve best prediction results, we tuned various the following hyperparameters and architectural components: number of training labels as determined by clustering k; learning rate η; dropout probability p; Hidden layer sizes and network depth; and embedding techniques for metadata embeddings. Table 1 summarizes the training results for different hyperparameter values.

k	η	p	**Train Accuracy**	**Test Accuracy**
6	0.0002	0.2	65.5%	52.6%
6	0.001	0.5	65.4%	56.7%
4	0.001	0.5	76.4%	66.8%
3	0.001	0.5	86.6%	77.9%

Table 1. Fusion model final training train and test accuracies after 2000 epochs for different hyperparameter configurations.

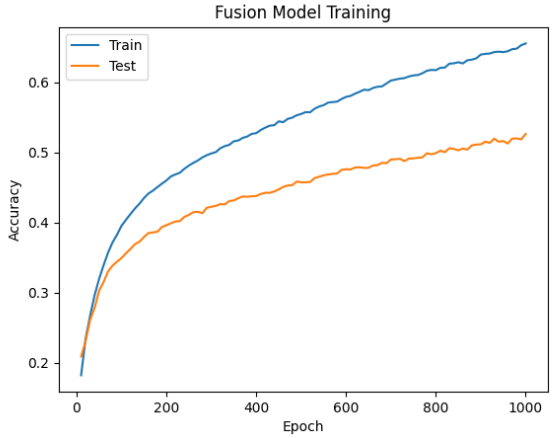

Figure 8. Initial experiment test and train accuracies over 1000 training epochs. The model overfits, which drives further hyperparameter tuning. This run corresponds to the first row of table 1.

Figure 8 shows the train/test progression over 1000 training epochs of an initial run of the model. Train accuracy exceeds test accuracy, signifying the model is over-fitting – this is a common failure point of this architecture. Train accuracy exceeds 65.5%, while test accuracy plateaus at around 53%.

To reduce model over-fitting, different hyperparameter configurations were attempted. Notably, dropout probability p was increased to improve regularization. Figure 9 shows an improved run with lower over fitting. This is the run used in the best-case recommender model evaluation, as it yielded a high mAP score.

Another set of experiments conducted involves reducing the number of training classes, meaning reducing the number of clusters performed during the genre tag clustering stage. With fewer classes, the training data become more easily separable in space, and so we expect higher train and test accuracy. This improved performance is seen in the third and fourth rows of table 1. Figure 10 shows

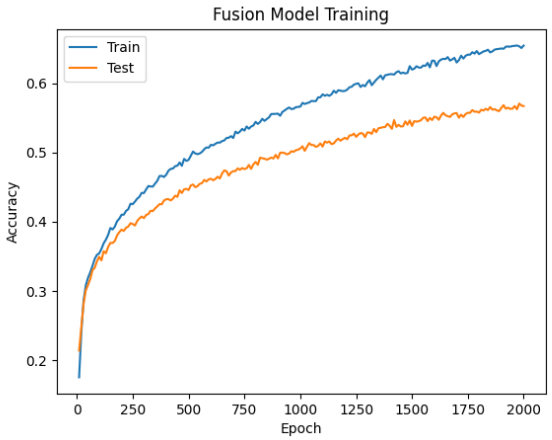

Figure 9. This experiment corresponds to the second row of table 1, trained through 2000 epochs. Note that the gap between train and test data is smaller, meaning that over-fitting was reduced by introducing a higher dropout rate.

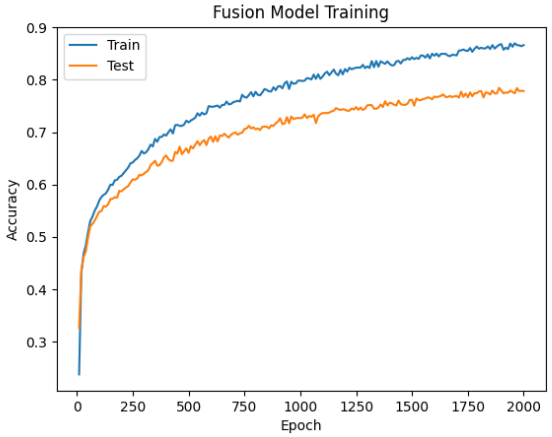

Figure 10. This experiment corresponds to the fourth row of table 1, trained through 2000 epochs. This run uses $k = 3$ training classes, resulting in higher train and test accuracies.

the train/test accuracy progression over 2000 epochs of the best-case $k = 3$ result. These accuracy metrics are comparable to the approximately 86% best-case accurate multimodal-fusion aggression detection network proposed by [3], whose architecture inspires our approach. Their approach classifies datapoints into 3 distinct classes. However, the model still struggles to generalize, as evidenced by the train accuracy being higher than test accuracy.

5.5. Recommendation Model Evaluation

Our recommendation model was evaluated under the same conditions as van den Oord et al. to use as a baseline for performance [5], using a subset of MSD with 9,330 songs and listening history for 20,000 users. In their research on deep content-based music recommendation, van den Oord et al. perform experiments against the 'metric learning to rank' (MLR) algorithm as a baseline for content-based recommendation. They propose various deep-learning based techniques to learn latent representations of songs in the MSD. In a manner very similar to our approach, the latent representations are used to score similarities between a user's listening history and songs in order to provide recommendations. They experiment with linear regression, multi-layer perceptron (MLP), and two CNN approaches trained with mean squared error (MSE) or weighted prediction error (WPE) as loss functions.

Model	Accuracy	mAP
MLR	–	0.01801
Fusion (metadata only)	54.0%	0.02050
Fusion ($k = 3$)	77.9%	0.02263
Fusion ($k = 4$)	66.8%	0.02304
Linear regression	–	0.02389
MLP	–	0.02536
Fusion ($k = 6$)	56.7%	0.04152
CNN with WPE	–	0.04323
CNN with MSE	–	0.05016

Table 2. Recommendation model evaluation for best-performing fusion model results. This table shows the test accuracy of the underlying models used for the content-based recommendation task for the models we implemented. Italicized models are results from [5], which serve as a baseline for evaluation.

Table 2 compares the mean average precision (mAP) of the recommender algorithm's output cut off at the top 500 songs per user for different model approaches. We aimed to maximize both the fusion model's test accuracy and mAP of the recommender – this is the heuristic for a successful model given a hyperparameter configuration.

When trained with metadata embeddings as the input data, the fusion architecture's DNN achieved a 54% accuracy on 6 training classes in the classification task. However, this configuration outperformed the MLR baseline in terms of recommendation mAP. With limited features, however, the model fails to outperform later model architectures. This gap in performance is expected, as van den Oord et al. leverage a CNN's strength in computing latent features rather than predicting a particular feature such as genre. Additionally, not all features (including lyrics embeddings) are used in training for this experiment.

When introducing lyrics features, best-case test accuracy increased to 56.7% and mAP increased to 0.04152, result-

ing in the model only being outperformed by the CNN approach with latent representation. These results are promising, however, as the CNNs are trained on the audio modality, which our model lacks due to the resource access limitations – in fact, the next leading model, the CNN with WPE, only achieved around a 4% improvement in mAP. Although the underlying model's test accuracy is not very high, the learned genre representation of the song is sufficient to make solid song recommendations comparable to a robust baseline. Ultimately, the strengths of our approach is the robustness of our embeddings. Leveraging deep neural networks through pretrained Word2Vec and the transformer-based RoBERTa architecture, we generate useful embeddings for our recommendation algorithm use case. However, the fusion model tends to overfit with lower prediction accuracies. A likely cause of this is the lack of audio features, which would be a compelling extension to our work. Our fusion model architecture, along with our preprocessing steps (normalization, dimensionality reduction) will likely highlight audio features in the fused embedding, resulting in more separable data, perhaps giving higher accuracy.

Interestingly, as the number of training classes k increases, the best-case mAP achieved by the fusion model decreases. A possible explanation for this observation is that the dot-product similarity computation in the recommendation algorithm is less sensitive to changes in the entries of the softmax vector, as there are less terms in the dot-product's sum. So, songs are more likely to be incorrectly scored as "similar" to a user. If audio features are introduced, it is possible that higher accuracies along with higher mAP can be achieved with more training classes. It is notable that for this recommendation system architecture, a fine balance must be hit between the number of training classes for more characteristic and accurate genre predictions and a high mAP score for higher-quality recommendations.

6. Conclusion

Overall, we achieve results that outperform conventional recommendation models with one dimensional song details. Given our fusion model architecture, using only the metadata and given features from the Million Song Dataset, our model resulted in a mean average perception of 0.02050, which was higher than that of the 'metric learning to rank' algorithm from the NeurIPS paper "Deep content-based music recommendations," [5]. With fusion with other modes of data including our lyrics, the model returned results more accurately, achieving up to 77.9% in test accuracy. Therefore, our architecture of the model exceeded current baselines along with conventional algorithms. This indicates that the multi-modal inputs along with processing of the data using given labels was significant when predicting music choices. However, a possible deterrent in the accuracy of the model was the lack of audio data and features. Due to the fact that audio samples required enterprise partnerships to access, our model did not use any specific audio features or characteristics. A possible future extension would be further modes of music properties beyond given data, such as audio, image components (i.e. the album cover), or the music video associated with different songs. Our study encourages further possible potential with the ability to use data to understand human behaviors with music listening.

Task	**Contributor**
Genre tag embedding and clustering	Daniel
Lyrical feature processing	Wonho
Audio feature exploration	Wonho
Metadata feature processing	Daniel
Fusion model implementation	Daniel
Song recommendation implementation	Daniel
Paper introduction and abstract	Wonho
Paper related work	Wonho
Paper technical approach	Daniel, Wonho
Paper data section	Wonho
Paper experiments section	Daniel
Paper conclusion	Wonho

Table 3. Project Contributions

7. Team Contributions

Table 3 describes the contributions to the project per person.

References

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011. 1

[2] Marco De Nadai, Francesco Fabbri, Paul Gigioli, Alice Wang, Ang Li, Fabrizio Silvestri, Laura Kim, Shawn Lin, Vladan Radosavljevic, Sandeep Ghael, David Nyhan, Hugues Bouchard, Mounia Lalmas, and Andreas Damianou. Personalized audiobook recommendations at spotify through graph neural networks. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 403–412, New York, NY, USA, 2024. Association for Computing Machinery. 1

[3] Noussaiba Jaafar and Zied Lachiri. Multimodal fusion methods with deep neural networks and meta-information for ag-

gression detection in surveillance. *Expert Systems with Applications*, 211:118523, 2023. 3, 7

[4] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, abs/1906.00091, 2019. 1

[5] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. 1, 3, 5, 7, 8