Using TF-IDF on Spotify Lyrics to Predict Hit Songs with Graph Neural Networks

Course project for CSE 6240: Web Search and Text Mining, Spring 2023

Jinyoung Eum
Georgia Institute of Technology
Atlanta, USA
jeum5484@gatech.edu

Nohelia Da Silva
Georgia Institute of Technology
Atlanta, USA
nohelia.dasilva@gatech.edu

Wenchao Wu
Georgia Institute of Technology
Atlanta, USA
wwu368@gatech.edu

ABSTRACT

Spotify has become a leading music platform, and generate many hit songs. Many previous studies have looked into the relationship between song features and its popularity, and have tried to predict whether a song is a hit, to boost the development of music platforms. In this study, we collect the data of Spotify songs in recent years and make hit predictions. We add lyric feature by using Term Frequencey-Inverse Document Frequency (TF-IDF) method to the song and test whether it would give us a better prediction. Also, We compared the performance of three models: Graph Convolutional Network (GCN), Graph Attention Network (GAT), and Random Forest. We found that the addition of lyric features improved the performance of all models, with Random Forest achieving the highest accuracy of 90.21%. We also found that Random Forest outperformed GCN and GAT in both cases, with and without lyric features.

KEYWORDS

prediction, feature extraction, Machine Learning, Natural Language Processing, audio features, Spotify, Graph Convolutional Network, Random Forest, TF-IDF, Graph Attention Networks

ACM Reference Format:

Jinyoung Eum, Nohelia Da Silva, and Wenchao Wu. 2023. Using TF-IDF on Spotify Lyrics to Predict Hit Songs with Graph Neural Networks Course project for CSE 6240: Web Search and Text Mining, Spring 2023 . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As Spotify becomes a leading music streaming platform with millions of active users and billions of streams every day, studies on predicting various attributes of songs with machine learning algorithms burgeon to find insights. In these studies, audio and lyrical data became the main features used to predict hit songs and presented high accuracy on prediction tasks [1–3].

Wang and Zhang [4] conducted a study on Spotify hit prediction using audio features to construct graphs, and found that graph Convolutional networks (GCN) have the potential to achieve higher accuracy than other machine learning algorithms like Random Forest and Neural Network. However, GCN's performance was not as good, possibly due to the presence of isolated nodes in the network which resulted in shorter length of walks and a large number of nodes without neighbors. The authors suggested that improvements could be made to their GCN model by adjusting the number of edges in the graph and the number of layers in the Neural Network, which need further research

In addition, Zhao et al. [5] concluded that adding music metadata, audio features and lyrics achieved the best performance in classifying hit songs through machine learning models. In consequence, we propose to combine audio and lyrical features for better hit song prediction with the benchmark of graph neural network algorithms.

Existing feature extraction methods on hit song prediction were mainly used for prediction with machine learning classification algorithms. Kumar et al. [1] applied two natural language processing techniques namely Bag-Of-Words and Term Frequency-Inverse Document Frequency (TF-IDF) on lyrical data for feature extraction and performed music genre prediction with Random Forest, Support Vector Machine, Naive Bayes, Linear Support Vector Classifier, and eXtreme Gradient Boosting (XGBoost). TF-IDF achieved the best accuracy on genre classification with 80.16% on XGBoost.

Our project aims to explore the idea of feature extraction on lyrical data to improve the accuracy of graph neural networks on Spotify hit song prediction. We primarily focus on extracting features from music lyrics and benchmarking graph neural network models by constructing graphs for prediction tasks. To achieve our objective, we use The Spotify Hit Predictor Dataset from 1960 to 2019 which consists of 22,693 songs and 18 audio features and The 150k Lyrics Labeled with Spotify Valence dataset which consists of 150,000 songs with 5 features. The two datasets were left joined on track and artist name and sentence similarity and vocabulary wealth were computed for each song.

The methods used in this project include the Sentence Similarity Coefficient, Graph Construction, Random Forest (baseline), Graph Convolutional Network (GCN), and Graph Attention Networks (GAT). TF-IDF method was applied on lyrics of each song to extract sentence similarity feature. The graph was constructed using each song as a node and edges were added between similar tracks using cosine similarity of acoustic feature vectors and sentence similarity coefficient vectors.

Graphs with and without sentence similarity coefficient were created. Random Forest algorithm as a baseline, as well as GCN and GAT models to predict hit songs. The accuracy of models with

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

sentence similarity are 86.36% for GCN, 85.33% for GAT, and 90.21% for Random Forest; while the accuracy of models without sentence similarity are 83.26% for GCN, 82.44% for GAT, and 89.18% for Random Forest. Based on this, we can conclude that adding lyrical features, specifically sentence similarity, can improve the accuracy of hit song prediction using graph neural network algorithms.

Our project on hit song prediction using graph neural networks and lyrical features has the potential to make a significant impact on the music industry. By achieving high accuracy in predicting hit songs, we have demonstrated the effectiveness of combining audio and lyrical features in graph neural network algorithms. This can provide valuable insights to stakeholders in the music industry, such as record labels, artists, and music streaming platforms. The project can help record labels to identify potential hit songs and invest their resources in promoting them, while also helping artists to make informed decisions about which songs to release or promote. Moreover, music streaming platforms like Spotify can personalize their music recommendations to users based on their listening history and preferences, leading to better user experience and higher user engagement. Overall, our project can contribute to the growth and development of the music industry by providing a data-driven approach to hit song prediction.

2 LITERATURE SURVEY AND BASELINES

Dimolitsas et al. [6] proposed a machine learning-based method for predicting hit songs using data from Spotify. The authors collected a dataset of audio and metadata features for over 25,000 songs, including information on acousticness, liveness, energy, tempo, danceability, valence, among others. They also collected data on the popularity of the songs, as measured by their number of streams on Spotify. The authors then used these features to train several machine learning models, including Random Forest, Support Vector Machine, Logistic Regression, and k-Nearest Neighbors, to predict the popularity of songs. These models were evaluated using accuracy, precision, and recall metrics. The results showed that the Random Forest model outperformed the other models, achieving a high accuracy in predicting hit songs. The authors also found that the most important features for predicting hit songs were danceability, energy, and valence. It was concluded that using machine learning to predict hit songs could be useful for the music industry in identifying potential hits, and the use of more than one music streaming platform as well as gathering more metadata about songs, such as information about melody and harmony, could lead to better results.

Joshua et al. [3] also did similar analysis by analyzing the relation between audio attributes of the song and its popularity. They divided the data into four categories including primary, numerical, dummy, and categorical. They used three machine learning methods: Kmeans, Linear Regression, and Random Forest to train the data, and showed that Random Forest would perform better than other two methods after using an appropriate number of decision trees. They thought it would be a potential way to predict hit songs in future industries. Also, it would predict better if they use more features like artist and the track information besides the audio of the song.

Martín-Gutiérrez et al. [7] constructed HitMusicNet to predict music popularity from multimodal data accurately. Two main strategies were Bag-Of-Words and TF-IDF. They performed feature extraction from the lyrics with these strategies and obtained the total number of sentences, the average number of words per sentence, the total number of words, the average number of syllables per word, a sentence similarity coefficient, and a vocabulary wealth coefficient. The sentence similarity coefficient was defined based on the TF-IDF and the computation of the cosine distance [7]. The vocabulary wealth coefficient was defined as an indicator of the diversity of the vocabulary. The authors suggest that TF-IDF outperforms the Bag-Of-Words strategy from the lyrical data. Thus, our group will perform the TF-IDF strategy for feature extraction from the lyrics.

3 DATASET DESCRIPTION

The Spotify Hit Predictor Dataset was obtained from Kaggle [8].It contains information on songs released on Spotify, as well as a binary label indicating whether a song is a Hit or Flop. The data preprocessing and cleaning for this dataset involved combining data from two sources: Spotify's API and Billboard's API. The first step was to use the Spotify API to retrieve audio features for tracks from 1959 to 2019. These were then merged with data obtained from Billboard's API, which provided information on the track's popularity and chart position. This dataset contains information on 22,693 songs with its corresponding 18 audio features, such as danceability, energy, and tempo.

The 150k Lyrics Labeled with Spotify Valence dataset was obtained from Kaggle [9]. It was constructed using Spotify API and an existing Song/Band/Lyrics Kaggle dataset to get a sentiment analysis dataset, where valence is used to measure positiveness. Non-english songs, songs with missing values and duplicate lyrics were removed, resulting in 150k songs. The valence values can serve as a measure of the overall mood or emotion conveyed by a song, which can be an important factor in determining its popularity. The dataset contains 150,000 rows of lyrics data with the corresponding information: artist, songs lyrics, song title, and label (spotify valence) with values between 0 and 1.

To merge these two datasets, the artist name and track name columns were cleaned by converting the text to lowercase, removing numbers and punctuation, replacing whitespace with underscores, removing accents, and stripping any remaining whitespace. The cleaned artist name and track name columns were then used to left join the two datasets, resulting in a new dataset with features from both datasets for songs that had matching artist and track names. Any rows with missing values were dropped. Lyrics are also cleaned by converting to lowercase and removing any punctuation and split into sentences, which are used to compute sentence similarity and vocabulary wealth. These two are added as columns to the dataset.

This dataset is enough to achieve the goals of the project because it includes a vast amount of data on songs released on Spotify, covering both their audio features and popularity. Additionally, merging this dataset with lyrics data that provides information on the overall sentiment conveyed by a song makes it a complete set of features. Using this dataset, machine learning models can be trained

	mean	std	min	25%	50%	75%	max
danceability	0.56	0.15	0.06	0.46	0.57	0.67	0.98
energy	0.64	0.22	0.02	0.48	0.66	0.81	1.0
key	5.28	3.51	0	2.0	5.0	8.0	11.0
loudness	-8.78	3.8	-27.12	-11.26	-8.28	-5.79	-1.1
mode	0.75	0.43	0	0	1.0	1.0	1.0
speechiness	0.06	0.05	0.02	0.03	0.04	0.06	0.53
acousticness	0.28	0.28	0	0.03	0.16	0.47	0.99
instrumentalness	0.04	0.14	0	0	0	0	0.97
liveness	0.19	0.16	0.02	0.09	0.13	0.26	1.0
valence	0.59	0.24	0.04	0.39	0.61	0.79	1.0
tempo	121.6	28.3	49.19	100.18	119.99	138.09	241.42
duration_ms	227555.59	69831.69	56827.0	182460.0	220760.0	259267.0	1367093.0
time_signature	3.93	0.31	1.0	4.0	4.0	4.0	5.0
chorus_hit	39.43	17.71	0	27.7	35.72	46.21	220.49
sections	10.17	3.05	2.0	8.0	10.0	12.0	55.0
vocab_wealth	0.65	0.01	0.54	0.65	0.65	0.65	0.6
sen_sim	0.08	0.06	0	0.04	0.06	0.1	1.0

Table 1: Data statistics

and tested to predict the popularity of songs based on their features, enabling the comparison of different models' performance.

4 EXPERIMENTAL SETTINGS

Here our main task is to predict whether a song would be a hit in Spotify given different features of it, like danceability, energy, and key, as we have described in the section of dataset description. We only include the 17 features from the dataset shown in the data statistics table. High-cardinality categorical features can pose challenges for data encoding, as they can create a large number of dummy variables, increase memory usage, and reduce model performance. We choose not to use the high-cardinality categorical columns like 'track', 'artist', 'uri', 'decade' from our dataset.

Whether a song would be a hit or not in the dataset is shown in a binary feature 'target'. It would be 1 if the song is a hit and 0 otherwise. The 'target' feature is already a binary feature included in the original dataset, and we do not need to categorize some original features into two types. We would use this feature in training and not use it while testing. Then we would compare the predicting result of the test dataset and the 'target' value of the test dataset.

In our final dataset after joining two original datasets together, we have 4847 rows (songs) in total. 4099 of them are hit songs while 748 of them are not. There are 15.4% of the songs are not hit, and thus we have enough negative examples in our experiment.

We would use accuracy, precision, recall, and F1 score as evaluation metrics of our prediction, and set the parameter as 'micro'. The accuracy of our model's hit song prediction is the critical component of our project, and these evaluation metrics measure accuracy from different perspectives. Moreover, using more than one metric would give us a more comprehensive measurement of the model performance.

Here we would split the data into 5 parts evenly and use cross-validation to test its prediction accuracy using different evaluation metrics. In each cross-validation, we would use 4 parts of the data to train our model and the last part of the data to test our model accuracy. We use online CPU and GPU resources to train and test our models depending on their complexity.

System setting:
CPU: Dual Xeon Gold 6226 (24 cores/node, 2.70 GHz)
RAM: 12 cores x 4GB, 48GB memory
Language: Python 3.8.8

5 METHODS

In this project, we choose the first option to run existing methods.

5.1 Sentence Similarity Coefficient

TF-IDF strategy showed better performance than the Bag-Of-Words strategy for lyrics feature extraction in music genre classification; HitMusicNet proposed sentence similarity feature by extracting with TF-IDF and cosine similarity [1, 7]. Thus, we used Algorithm 1 proposed by Martín-Gutiérrez et al. [7] to extract the sentence similarity coefficient from our merged dataset.

As we extract the sentence similarity coefficient feature, we generated a dataset without the sentence similarity coefficient feature to benchmark the performance of the lyrics feature with the TF-IDF strategy.

5.2 Network Construction

We constructed a graph based on our dataset for GCN and GAT models. Each node in the graph represented a track. We added an edge between the two tracks if the tracks are considered similar. The similarity between the two tracks was determined by the cosine similarity of the acoustic feature vectors and sentence similarity coefficient vectors. We set our threshold of similarity to 0.85 and generated graphs. Since we want to test the effect of sentence similarity coefficient vectors, we generated a graph constructed based

Algorithm 1: Sentence Similarity Coefficient

```
Data: lyric for each song
Result: sentence similarity coefficient
if |lyric| > 1 then
    l ← lyric
    tf_idf ← computeTfIdf(l)
    m ← computeCosineSimilarity(tf_idf)
    diag ← getUpperDiagonal(m)
    t ← |diag|
    n ← 0
    for s in diag do
        if s ≥ μ then
            n ← n + 1
        sim ← n/t
    end
return sim
```

on cosine similarity involving sentence similarity coefficient vectors and a graph that was constructed without sentence similarity coefficient vectors.

5.3 Random Forest (baseline), GCN and GAT

The Random Forest algorithm [10], proposed by Leo Breiman in 2001, combines multiple decision trees to improve accuracy and reduce overfitting. It randomly selects subsets of features and data samples to create decision trees, and then aggregates the predictions to make a final prediction. Hyperparameters include the number of trees, tree depth and others. Here we set all hyperparameters as default values of random forest classifier in sklearn library. The number of trees is 100. The max depth of tree is none. The minimum number of samples required to split an internal node is 2. The minimum number of samples required to be at a leaf node is 1. The algorithm can be implemented in Python using scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. By combining feature and lyrics datasets, we can create a comprehensive dataset for predicting hit songs. Random Forest is a widely used and effective model for classification problems, making it a suitable choice for this task.

Kipf et al. [11] describes the Graph Convolutional Network (GCN) method for processing graph-structured data. GCN learns node representations by aggregating information from neighboring nodes and can be used for various tasks. The hyperparameters of GCN include the number of hidden layers, filters per layer, and dropout rate. The code for implementing GCN is available in various deep learning libraries such as PyTorch Geometric in Python: https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html#torch_geometric.nn.conv.GCNConv. GCN is a suitable choice for predicting hit songs since it captures the graph structure of the dataset and incorporates both feature and text information in a unified model. We built a 4-layer neural network from our input data size to the number of classes (hit or not hit). The detailed network structure is shown in Figure 1. We added the ReLU layer and dropout layer with a 0.2 dropout rate between convolutional layers. Then, we used LogSoftmax as our activation function.

```
GCN(
  (conv1): GCNConv(4847, 16)
  (conv2): GCNConv(16, 64)
  (conv3): GCNConv(64, 16)
  (conv4): GCNConv(16, 2)
)
```

Figure 1: Graph Convolutional Networks Structure

Velickovic et al. [12] proposes Graph Attention Networks (GAT), which is a deep learning method for processing graph-structured data, similar to GCN. However, unlike GCN, GAT uses attention mechanisms to adaptively emphasize different parts of the graph for different nodes. The hyperparameters of GAT include the number of hidden layers, the number of attention heads, and the dropout rate. GAT code is available in various deep learning libraries such as PyTorch Geometric in Python: https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html#torch_geometric.nn.conv.GATConv. GAT is a suitable choice for studying the relationship between lyrics and musical features in hit song prediction, as it can effectively capture the graph structure of the dataset and incorporate both feature and text information in a unified model while giving different importance to different parts of the graph. Similar to GCN, we built a 2-layer neural network from our input data size to the number of classes. The detailed network structure is shown in Figure 2. We added dropout layers with a 0.2 dropout rate between GAT layers and used LogSoftmax as our activation function.

```
GAT(
  (layer_1): GATConv(4847, 16, heads=8)
  (layer_2): GATConv(128, 2, heads=8)
)
```

Figure 2: Graph Attention Networks Structure

6 EXPERIMENTS AND RESULTS

Table 2: Graph Properties

Properties	Network with SS	Network with no SS
Number of Nodes	4847	4847
Number of Edges	11744281	11744281
Network Density	1	1

Table 3: Results of Models Without Sentence Similarity

Model	Accuracy	Precision	Recall	F1
GCN	84.30%	84.30%	84.30%	0.84
GAT	82.44%	82.44%	82.44%	0.82
RF	89.18%	89.18%	89.18%	0.89

Table 4: Results of Models with Sentence Similarity

Model	Accuracy	Precision	Recall	F1
GCN	84.50%	84.50%	84.50%	0.85
GAT	83.06%	83.06%	83.06%	0.83
RF	90.21%	90.21%	90.21%	0.90

Using datasets with and without sentence similarity coefficient vectors, we generated two track networks. The network properties are shown in Table 2. Sentence similarity did not affect the structure of the graph as the properties show no changes in the number of nodes, number of edges, and network density.

From the results in Table 3 and Table 4, we can see that models without sentence similarity show lower performance compared to the models with sentence similarity. Using the feature sentence similarity, the accuracy of random forest, GCN model, and GAT model improved 1.03%, 0.20%, and 0.62% separately. The result suggests that there was an improvement in the performance of GCN and GAT models from without to with sentence similarity coefficient. We can conclude that extracting the sentence similarity coefficient by TF-IDF and adding the sentence similarity coefficient helps to improve the model overall.

The random forest may have the highest accuracy because of simple prediction tasks with our dataset. GCN performing slightly better than GAT could be due to the self-attention mechanism of GAT performing similarly to the weighted average of all neighbors of GCN from our generated graph. Yet, a low-level technical assessment is required to identify the reasons for the results.

Measured by accuracy, random forest performs the best and GAT performs the worst among three models, either in the group with sentence similarity feature or without sentence similarity feature. However, in the future, significant improvement of GCN and GAT models suggests further improvement of them to predict better.

7 CONCLUSION

Even though the sentence similarity coefficient did not affect the overall structure of the graph representation, we conclude that the graph generated with the sentence similarity coefficient improved the performance of GCN and GAT in predicting the hit songs.

There are some limitations that we could improve. First, we could have more lyric data points for training. After joining two datasets, we only have 4847 songs, and we may explore more datasets to get a larger dataset for model training. With more training, test, validation, and negative samples, models may more accurately predict the hit songs. Next, we may try to test more hyperparameters to train our model to get better performance. With a limited time frame, we tested few parameters, yet testing different hyperparameters may result in higher performance.

In the future, we need to explore more datasets to get a larger dataset for model training. Moreover, we can find more aggregated features from the dataset that improve the quality of the graph and the performance of the GNN models. We can look into previous studies to think of what kind of features we still need. Also, we can try using other GNN models which are probably more suitable for classifying and predicting hit songs.

8 CONTRIBUTIONS

All team members have contributed a similar amount of effort.

REFERENCES

[1] Akshi Kumar, Arjun Rajpal, and Dushyant Rathore. Genre classification using feature extraction and deep learning techniques. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, pages 175–180. IEEE, 2018.
[2] Sebastian Raschka. Musicmood: Predicting the mood of music from song lyrics using machine learning. *arXiv preprint arXiv:1611.00138*, 2016.
[3] Joshua S Gulmatico, Julie Ann B Susa, Mon Arjay F Malbog, Aimee Acoba, Marte D Nipas, and Jennalyn N Mindoro. Spotipred: A machine learning approach prediction of spotify music popularity by audio features. In *2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T)*, pages 1–5. IEEE, 2022.
[4] Xueyu Wang and Ruixuan Zhang. Spotify hit prediction. 2021.
[5] David Cameron Frank Hopfgartner Mengyisong Zhao, Morgan Harvey and Valerie J. Gillet. An analysis of classification approaches for hit song prediction using engineered metadata features with lyrics and audio features.
[6] Ioannis Dimolitsas, Spyridon Kantarelis, and Afroditi Fouka. Spothitpy: A study for ml-based song hit prediction using spotify. *arXiv preprint arXiv:2301.07978*, 2023.
[7] David Martín-Gutiérrez, Gustavo Hernández Peñaloza, Alberto Belmonte-Hernández, and Federico Álvarez García. A multimodal end-to-end deep learning architecture for music popularity prediction. *IEEE Access*, 8:39361–39374, 2020.
[8] Farooq Ansari. The spotify hit predictor dataset (1960-2019), 2020.
[9] Edenbd. 150k lyrics labeled with spotify valence, 2020.
[10] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.
[11] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
[12] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.