Song Recommendation Based on Lyrical Similarity

Skandan Senthil Nathan
Northeastern University, Boston
senthilnathan.s@northeastern.edu

Abstract

This project aims to develop a recommendation system for songs based on their lyrics using the Doc2Vec algorithm. The problem is to recommend relevant songs to users based on a phrase or an existing song they provide. The solution involves training a Doc2Vec model on a dataset of song lyrics, preprocessing the data, and using the trained model to generate vector representations for the user's input. These vector representations are then compared with the song vectors to recommend the most similar songs. The results demonstrate the effectiveness of the Doc2Vec model in capturing semantic similarities between lyrics and providing relevant recommendations.

Introduction

Music recommendation systems have become increasingly important in the age of streaming services and vast music libraries. Recommending songs based on lyrics can provide a more personalized and meaningful experience for users, as lyrics often convey the essence and emotions of a song. This project explores the use of the Doc2Vec algorithm, a neural network model with natural language processing techniques, to create a recommendation system that suggests songs based on their lyrical content. By leveraging the semantic relationships captured by Doc2Vec, the system can recommend songs with similar themes, emotions, or narratives, enhancing the user's discovery of new music.

The importance of lyric-based recommendation systems lies in their ability to capture the nuanced meanings and contexts present in song lyrics. Unlike traditional recommendation systems that rely solely on metadata or audio features, a lyric-based approach can understand the deeper themes, narratives, and emotional resonances that resonate with listeners. By analyzing the semantic content of lyrics, the system can identify songs that share similar concepts, metaphors, or storytelling elements, providing users with recommendations that align with their interests and preferences on a closer level.

Furthermore, lyric-based recommendations can foster a deeper appreciation and understanding of music by exposing users to songs they may have overlooked based solely on genre or artist preferences. It allows for discoveries of songs that resonate on an emotional or thematic level, regardless of their musical style or popularity.

Developing an effective lyric-based recommendation system presents several challenges. Mainly, it requires robust natural language processing techniques and a powerful model to accurately extract and represent the semantic content of lyrics.

This project aims to address these challenges by leveraging the Doc2Vec algorithm, a powerful technique for learning dense vector representations of text documents. By training the Doc2Vec model on a large corpus of song lyrics, the system can capture the semantic relationships between songs, enabling accurate and relevant recommendations based on lyrical content.

The remainder of this report will delve into the background of the Doc2Vec algorithm, related work in the field of music recommendation systems, a detailed description of the project methodology, experimental results, and analysis, and concluding remarks on the effectiveness and potential future directions of this approach.

Background

Word vectors, also known as word embeddings, are dense vector representations of words that capture their semantic and contextual relationships. These vector representations are learned from large text corpora using neural network-based models, such as Word2Vec or GloVe. The key idea behind word vectors is that words with similar meanings or contexts tend to have similar vector representations, allowing them to be positioned close together in the high-dimensional vector space. For example, the word vectors for "king" and "queen" would be closer in the vector space compared to the vectors for "king" and "apple".

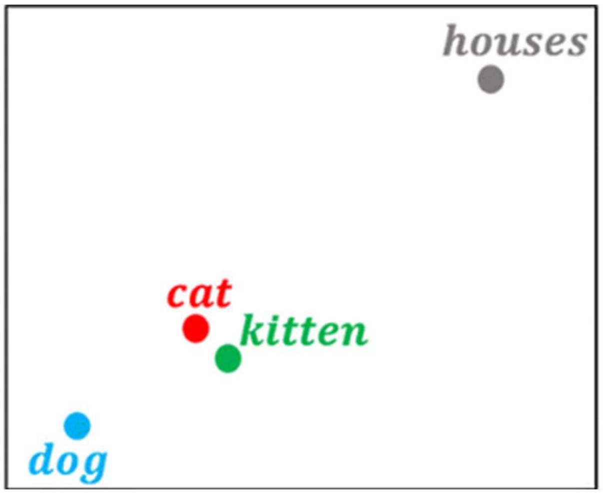


Figure 1: Words represented in a 2d space by their word vectors.

The Doc2Vec algorithm is a variant of the popular Word2Vec algorithm, which learns vector representations of words based on their context in a corpus. While Word2Vec focuses on learning word embeddings, Doc2Vec extends this concept to learn vector representations of entire documents, capturing the semantic relationships and contextual information present in larger textual units.

The key idea behind Doc2Vec is to treat each document as a single unit and learn to predict the words in the document based on the document vector and the context words. This is achieved by introducing a document vector, which acts as a memory that captures the overall semantic content of the document. During training, the algorithm tries to maximize the log probability of correctly predicting the target word given the surrounding context words and the document vector.

There are two main variants of the Doc2Vec algorithm: Distributed Memory Paragraph of Vectors (DM-PV) and Distributed Bag of Words (DBOW). The DM model considers both the context words and the document vector to predict the target word, while the DBOW model ignores the context words and relies solely on the document vector to predict the target words. This exploration is focused on the DM-PV model.

Related Work

Several other methods have been explored for music recommendation systems, such as collaborative filtering, content-based filtering, and hybrid approaches. Collaborative filtering techniques rely on user ratings or listening behavior to recommend songs like what other users with similar preferences have enjoyed. While these methods can provide accurate recommendations, they suffer from the cold-start problem, where new songs or users with limited data cannot be effectively recommended.

Content-based filtering methods analyze the audio features or metadata (e.g., genre, artist) of songs to recommend similar items. These methods can provide recommendations without relying on user data but may struggle to capture the deeper semantic and emotional aspects of music, which are often conveyed through lyrics.

Hybrid approaches combine collaborative filtering and content-based filtering techniques to leverage the strengths of both methods. However, these approaches still largely rely on user data or audio features and may not fully capitalize on the rich information present in song lyrics.

The Doc2Vec approach used in this project aims to address these limitations by directly learning dense vector representations of song lyrics, capturing the semantic and contextual information in a more comprehensive manner.

Project Description

The problem of recommending a song based on lyrics can be formally defined in the following way. Let D = {d_1, d_2, ..., d_N} be the set of N lyric documents (songs) in our dataset, where each lyric document di is represented as a sequence of words (w_1, w_2, ..., w_M). The goal of the Doc2Vec model is to learn a mapping function f that projects each lyric document di into a dense vector representation vi in a continuous vector space V as f: D → V. The vector representation vi should capture the semantic and contextual information present in the lyric document di, such that similar lyrics have vectors that are close in vector space V. Once the Doc2Vec model is trained, it can be used to generate vector representations for new, unseen lyrics or phrases v_p. The problem of recommending songs based on lyrics can then be formulated as finding the top K lyric documents d_k in the dataset D whose vector representations v_k have the highest cosine similarity with the vector representation $v_{p.}$

Working of the Doc2Vec Architecture: This project uses DM-PV version of the Doc2Vec model. It uses neural network architecture to learn the weights and train. The inputs to the model are the word vectors and the document vector. The values for them are randomly initialized at the start. These document vectors and the word vectors are concatenated together and sent as input to the hidden layers in the neural net. The hidden layers do their summation, and the output is returned after a SoftMax function is applied to it.

This is the predicted target word. Once this is done, back propagation and gradient adjustments are done for every epoch. Finally, the weights for the word vectors and the document vector are learned.

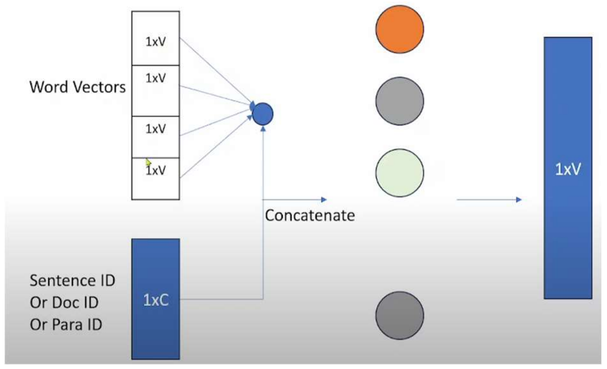


Figure 2: Doc2Vec PV-DM Architecture

Data Preprocessing: The song lyrics data is preprocessed to prepare it for the Doc2Vec model. This involves several steps:
Lowercasing: All tokens are converted to lowercase to ensure consistency and reduce the vocabulary size.
Removing Special Characters: Special characters, punctuation, and digits are removed from the lyrics as they do not contribute to the semantic content.
Stop Word Removal: Common stop words (e.g., "the", "is", "and") are removed as they do not carry significant meaning.
Lemmatization: Lemmatization can be applied to reduce inflected words to their base or root form, further reducing the vocabulary size.

Preparing Data for Doc2Vec: After preprocessing, the lyrics are converted into a format suitable for the Doc2Vec algorithm. This involves creating tagged documents, where each document (lyric) is associated with a unique tag representing the song or artist. The tagged documents take the form:

TaggedDocument(words=[‘this’, ‘is’, ‘the’, ‘song’], tags=[‘song1’])

These tagged documents serve as the input to the Doc2Vec model during training.

Training the Doc2Vec Model: The Doc2Vec model is trained on the tagged documents using the following steps:
Model Initialization: The Doc2Vec model is initialized with hyperparameters such as vector size, window size, minimum word count, and other configuration settings.
Model Training: The training process involves feeding the tagged documents into the Doc2Vec algorithm, which learns to predict the target words based on the surrounding context words and the document vector. The objective function being optimized is:

$$\frac{1}{T} \times \sum_{t=k}^{T-k} \log P(w_t | w_{t-k}, \dots, w_{t+k}, D)$$

where T is the total number of samples for training, w_t is the target word, w_{t-k} and w_{t+k} are the context words, and D is the document (lyric) vector. The algorithm tries to maximize the log probability of correctly predicting the target word given the surrounding context words and the document vector.
Optimization: The model parameters are optimized using techniques like stochastic gradient descent.

User Input and Vector Generation: When a user provides an input, either a phrase or an existing song from the dataset, the trained Doc2Vec model is used to generate a vector representation for the input:
Song Input: If the user provides an existing song from the dataset, the corresponding document vector learned during training is used as the input vector.
Phrase Input: For a phrase input, the algorithm computes the phrase vector in a different way. It treats the trained Doc2Vec model as frozen, and considers the input phrase as a new, previously unseen document. It initializes a random vector for this new document, and then uses a training-like process to optimize this vector to best predict the words in the input phrase, given the existing word and document vectors in the trained model. Essentially, it tries to find the document vector that, when combined with the word vectors in the model, maximizes the likelihood of generating the words in the input phrase.

Similarity Calculation and Recommendation: Once the input vector is obtained, the system calculates the cosine similarity between the input vector and the vectors of all songs in the dataset. The cosine similarity between two vectors x and y is calculated as:

$$\cos(x, y) = \frac{x \cdot y}{||x|| \times ||y||}$$

The top N songs with the highest cosine similarity scores are recommended to the user as the most relevant songs based on lyrical content.

Various techniques were employed to improve the performance and robustness of the system, such as:

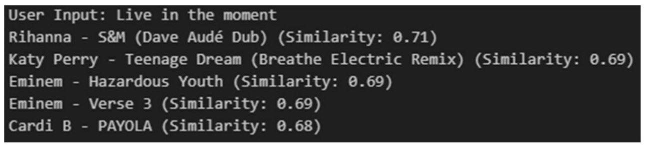
Hyperparameter Tuning: Grid search and cross-validation were used to find the optimal hyperparameters for the Doc2Vec model, including vector size, window size, and minimum word count.
Negative Sampling: Negative sampling techniques were employed to improve the training efficiency and quality of the learned vectors.

The implementation of the project involved the use of popular Python libraries and frameworks for natural language processing and machine learning, such as Gensim, NLTK, scikit-learn, and NumPy.

Experiments

To evaluate the performance of the recommendation system, several experiments were conducted:

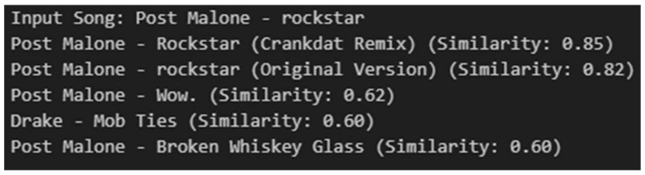
Phrase-based Recommendations: Users provided various phrases, and the system recommended songs based on the semantic similarity of their lyrics to the input phrase. For example, phrases like "love song," "upbeat summer," or "Live in the moment" were used as input, and the system recommended songs with lyrics that matched those themes or emotions.



```
User Input: Live in the moment
Rihanna - S&M (Dave Audé Dub) (Similarity: 0.71)
Katy Perry - Teenage Dream (Breathe Electric Remix) (Similarity: 0.69)
Eminem - Hazardous Youth (Similarity: 0.69)
Eminem - Verse 3 (Similarity: 0.69)
Cardi B - PAYOLA (Similarity: 0.68)
```

Figure 3: Recommending songs based on user input.

Song-based Recommendations: Users provided an existing song from the dataset, and the system recommended similar songs based on lyrical content. This experiment aimed to evaluate the system's ability to identify songs with comparable themes, narratives, or emotional resonances. As shown in Figure 4, most lyrically like the song “rockstar” by “Post Malone” were covers and remix version of it, followed by songs with similar lyrical meaning.



```
Input Song: Post Malone - rockstar
Post Malone - Rockstar (Crankdat Remix) (Similarity: 0.85)
Post Malone - rockstar (Original Version) (Similarity: 0.82)
Post Malone - Wow. (Similarity: 0.62)
Drake - Mob Ties (Similarity: 0.60)
Post Malone - Broken Whiskey Glass (Similarity: 0.60)
```

Figure 4: Recommending songs based on a given song.

Lyrical Similarity Analysis: To further investigate the capability of the Doc2Vec model in capturing semantic similarities and differences between song lyrics, an analysis was performed on the learned vector representations of various artists. The t-SNE (t-Distributed Stochastic Neighbor Embedding) technique was used to project the high-dimensional lyrical vectors into a 2D space for visualization purposes.

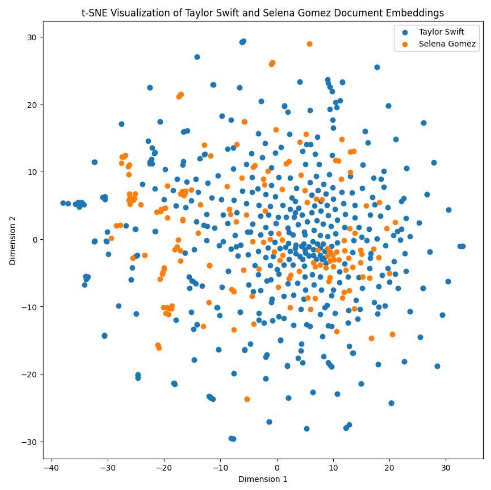


Figure 5: T-SNE Visualization of lyrically similar artists.

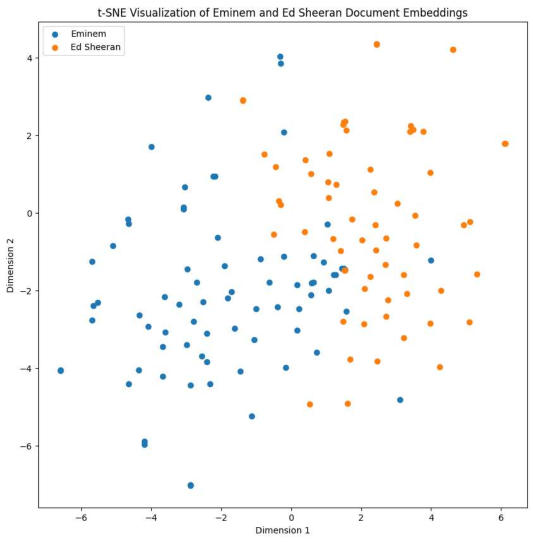


Figure 6: T-SNE Visualization of lyrically dissimilar artists

This visualization reveals interesting patterns in the lyrical content of different artists. As expected, artists like Taylor Swift and Selena Gomez, who often create similar genres of music, exhibit strong lyrical similarities, as indicated by their proximity in the 2D space. Conversely, artists like Eminem and Ed Sheeran, who come from vastly different musical backgrounds, have lyrical content that is more dissimilar, as represented by their separation in the visualization. These findings suggest that the Doc2Vec model can effectively capture the semantic nuances and contextual relationships present in song lyrics, enabling accurate representations of lyrical similarities and dissimilarities.

Comparison with Baseline Methods: The performance of the Doc2Vec-based system was compared to methods, such as recommendations based on artist or genre. Recommendation of songs in the opposite genre were also observed in the lyric based model. This allowed for an assessment of the added value provided by the lyric-based approach compared to more traditional recommendation methods.

Parameter Tuning: Various hyperparameters of the Doc2Vec model, such as vector size, window size, and minimum word count, were tuned to optimize the performance of the recommendation system. Grid search and cross-validation techniques were employed to find the best combination of parameter values.

Qualitative Analysis: In addition to quantitative metrics, a qualitative analysis was performed by examining specific examples of recommendations and assessing their relevance and interpretability. This analysis provided insights into the strengths and weaknesses of the system and helped identify areas for improvement.

Source Code for the Project:
https://github.com/skandansn/Lyric-recommender

Conclusion

The Doc2Vec-based recommendation system successfully captured semantic similarities between song lyrics and provided relevant recommendations to users. The results demonstrated the effectiveness of the approach, particularly in recommending songs with similar themes, emotions, or narratives. While at times it retained the ability to suggest songs like those recommended by traditional genre-based or collaborative filtering approaches, the system also excelled in offering unique recommendations based on the semantic similarities captured in the lyrics.

However, there were instances where the system struggled, such as recommending songs from a niche topic or failing to capture certain contextual nuances in lyrics. Future work could explore hybrid approaches that combine lyrical content with other metadata or audio features to further improve the recommendation quality. Additionally, incorporating advanced natural language processing techniques, such as context-aware word embeddings or transformer models like GPT, could potentially enhance the system's ability to capture the complex semantics and contextual information present in lyrics.

Another area for future exploration is the incorporation of user preferences and feedback into the recommendation system. By combining the lyric-based approach with collaborative filtering techniques, the system could learn to personalize recommendations based on individual user tastes and listening histories, further enhancing the relevance and accuracy of the recommendations.

Overall, this project showcased the potential of using natural language processing techniques, specifically the Doc2Vec algorithm, for music recommendation systems. By leveraging the rich semantic information present in song lyrics, the system was able to provide meaningful and relevant recommendations, leading to more personalized and enriching music discovery experiences.

References

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. in Proceedings of Workshop at ICLR.

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. in Proceedings of Workshop at NIPS.

Gemsim Word2vec package
https://radimrehurek.com/gensim/models/word2vec.html

Gemsim Doc2vec package
https://radimrehurek.com/gensim/models/doc2vec.html

Stack Exchange
https://datascience.stackexchange.com/questions/37488/doc2vec-how-does-the-inference-step-work-in-pv-dbow?rq=1