Lyric-based recommendation for music playlists

Ilai Bachrach **Marc Oliveau**
{ilaibachrach, marc.oliveau}@gmail.com

Abstract

In the decades to come, content-based platforms will have the opportunity to choose between building recommendation systems that fuel habit or those that enable users to grow and explore. Here we test the performance of different word embeddings on the task of recommending songs based off of the content of lyrics. We fine tune the models' dimensions for the best performance and evaluate the performance of these word embeddings through cosine similarity scores and real-life playlists. Our results for the reduced TF-IDF and Doc2Vec indicate user preferences are not primarily focused on the lyrics of songs. However, SBERT outperforms these embeddings, suggesting that there is a somewhat weak correlation between lyrics and preferences for songs. Lastly we provide insight into how using the methods applied in this paper can provide critical new perspectives for users that are increasingly trapped in musical echo chambers.

1 Introduction

In the past years the music industry has heavily relied on user-generated collaborative filtering approaches to drive song recommendation algorithms. However, with the advent of new and powerful word embedding tools like BERT, we aim to identify a hybrid path forward for recommendation systems that involves content-based filtering.

2 Related Work

Unlike collaborative filtering which relies heavily on user data to suggest items to users, content-based filtering does not need user generated data. Content-based recommendations rely on features of a song, such as their vibrancy and energy.

Previous research has identified the success of specifically the related genres of a song in recommending songs to users (Adiyansjah et al, 2019). A song for example, may have different genre tags that overlap with those of other songs.

In a paper from 2010, the authors look at different semantic information from a user's playlist like genre, culture and moods. Given a list of songs to represent a user's preferences, the author's paper focuses on different similarity representations of features of these songs. The authors consider both semantic distance from mean where the liked songs from the initial playlist are combined into one vector, and semantic distance from all tracks, where each track is considered individually. Our paper focuses on the former approach. The authors also look at a probabilistic model, however, even so, content-based filtering approaches did not compare against their collaborative filtering algorithm. (Bogdanov et al, 2010)

Six years later, also in the setting of content based filtering, another paper investigates different vector space representations of both items and user profiles extracted from Wikipedia. Unlike in the previous paper, here, the authors report scores similar to collaborative filtering based approaches. Therefore it is clear that the effectiveness of content based filtering depends heavily on the context of its implementation (Ding et al, 2015).

Another approach outlines the significance of the feature dimensions of a dataset one uses when applying content-based filtering. Here, the authors had access to whether a user skipped or repeated a song in addition to the social tag of a song. As such they could incorporate the degree to which a user liked a song in the form of weights (Musto et al, 2016). While we do not go this route, we see a opportunity for using more dimensional data sets in combination with the methods described in this paper to achieve even better similarity scores.

3 Dataset Description

We use two data sets. The first dataset is from a user on Kaggle. This dataset has feature information for 18000 songs found on Spotify. Features include lyrics, genres, as well as audio features. Importantly, the author added lyrics to each song using the genius library in R. This means that for some songs, it is likely that the song title may not match with exact lyrics. The second dataset is a collection of one million playlists with contained song names. Therefore, our only metric of users music preference comes

from knowing that the collection of songs in a playlist is not random but selected by a user based on taste preferences.

From these two datasets we found an intersection of +72000 playlists with + 10 songs for which we have lyrics and that exist in the Spotify playlists. We also focus on English songs only. For our evaluation we use 100 playlists for each word embedding. Since this intersected dataset may be useful for future research, we have made it available through our GitHub repository.

4 Method

Starting from an explicit set of music tracks provided by the user as evidence of their music preferences, we initially designed a recommendation procedure that 1. represents a song lyric in vector space using a TF-IDF vectorizer and 2. finds n closest songs based on cosine similarity.

4.1 Cosine Similarity of several input songs

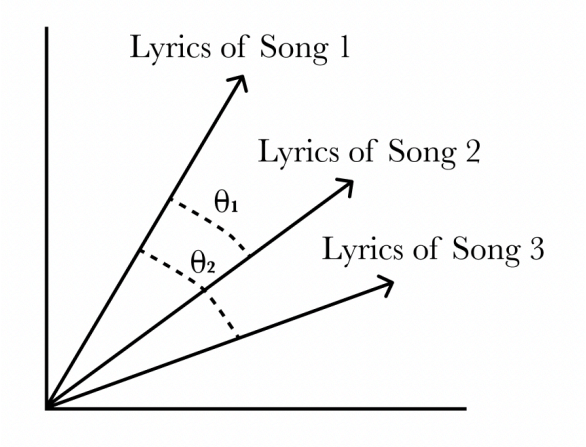

Figure 1. Cosine distance of different songs.

$$\frac{A \cdot B}{||A|| x ||B||}$$

In order to find the cosine similarity of two songs (A, B) we compute their respective dot product over the multiplication of the magnitude of the individual vectors.

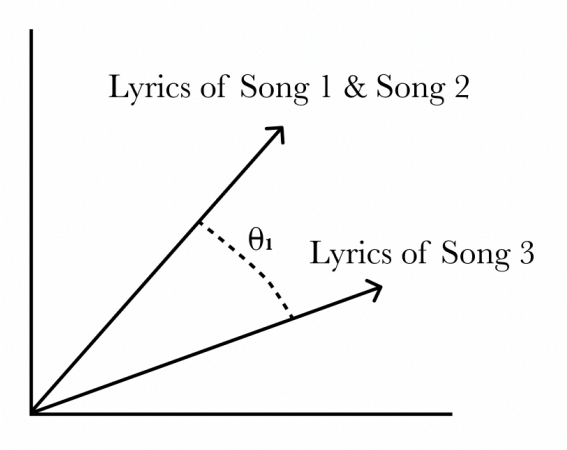

Figure 2. Mean Vector for two songs.

With a collection of songs as input, each with their own respective vector, we compute a mean vector out of these songs. This will allow us to compare the similarity score between the input playlist and songs in the initial dataset.

4.2 Different text representations

Here we describe the different word embeddings we used to represent the lyrics of songs.

TF-IDF or the term frequency inverse document frequency matrix is given by the:

$$w_{i,j} = tf_{i,j} \times log(\frac{N}{df_i})$$

where the term frequency TF(t, f), the number of times term t appears in document f is multiplied by IDF(t), the log of the number of documents over the document frequency of term t. Like the count vectorizer, the term frequency inverse document frequency matrix does not incorporate similarity between two different words. Even so, we expect the lowest results here. TF-IDF provides the baseline for our investigation.

Unlike TF-IDF, Doc2Vec works with distributional semantics.

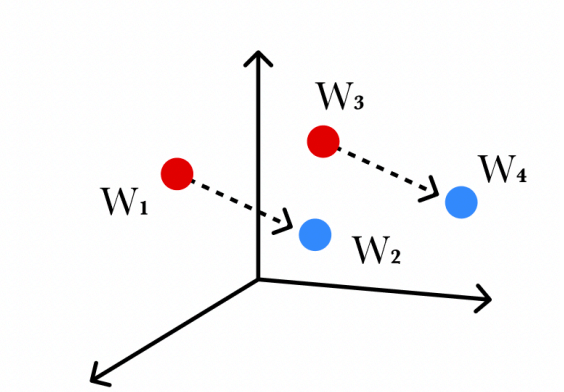

Figure 3. Words with similar meaning are connected to other words with similar relational meaning.

Doc2Vec is based on Word2Vec which uses a continuous bag of words to predict words from context and skip-gram, which does the opposite. Unlike Word2vec, Doc2Vec adds another paragraph ID vector to the CBOW model. Thus Doc2Vec can be applied to documents in the form of song lyrics.

Both TF-IDF and Doc2Vec are sparse matrices. We therefore apply truncated SVD to reduce the dimensionality to 100 to allow for faster computing. In a user setting it is important to consider the trade-off between speed and performance.

Next we use BERT Word Embeddings. Since BERT itself has a high computational demand, we use a method conveyed by Nils Reimers, Iryna Gurevych, known as Sentence-BERT. In general BERT uses a cross-encoder, given 10,000 sentences, the transformer runs through all possible sentence combinations. This causes immense computational overhead. Sentence-BERT goes about this using Siamese and triplet network structures. Siamese allows SBERT to find a fixed size vector for input sentences which can then quickly be compared using cosine similarity. We run the sentence transformer algorithm of BERT using SBERT. We then encode the list of lyrics using the transformer to receive our document embeddings (Reimers et al, 2019).

4.3 Identifying Recommendations

In order to compute recommendations we find the mean vector from a collection of songs that make up around 70% of a given playlist. The lyric vector of a collection of songs corresponds to a specific row in the matrices obtained from the word embeddings. We compare this vector to the collection of original songs that are not in the input portion of the playlist. The number of songs we recommend is the same length as the number of songs used for testing, around 30% of the original playlist length.

5. Results and Findings

Initially we test the performance of all three word embeddings using a Boolean-wise check, against the 30% that was left of the original playlists. However this yields almost no overlapping results. Next we did a cosine similarity evaluation which considers how close a song is to the songs from the test set. We do this evaluation using different dimensions for the TF-IDF and Doc2Vec. We also test whether the size of the playlist has an impact on the performance of the word embedding.

TF-IDF DIMENSIONS	Playlist Size			
	10	50	80	Average
100	0.140	0.149	0.162	0.150
300	0.159	0.159	0.163	0.160
500	0.156	0.169	**0.178**	**0.168**
Average	0.152	0.159	**0.168**	

Figure 4. TF-IDF results for different dimensions and playlist sizes.

DOC2VEC DIMENSIONS	Playlist Size			
	10	50	80	Average
100	0.171	0.167	**0.185**	**0.174**
300	0.163	0.172	0.162	0.166
500	0.153	0.158	0.171	0.161
Average	0.162	0.166	**0.173**	

Figure 5. Doc2Vec results for different dimensions and playlist sizes.

As expected, TF-IDF acts as a baseline model with the worst maximum scores. However, Doc2Vec does not perform much better. TF-IDF works best with high dimensionality and large playlist lengths. Doc2Vec in contrary, works best with low dimensionality and large playlist lengths.

SBERT	Playlist Size			
	10	50	80	Average
	0.251	0.260	0.269	0.260

Figure 6. SBERT results for different dimensions and playlist sizes.

SBERT performs substantially better in comparison to previous models, however it is still below a point in which we can conclude substantial relevance.

6. Conclusions

The results indicate that users put low importance to the lyrics of a song. However, SBERTs higher score indicates a slightly more relevant semantic importance of lyrics in music preferences of users than initially assumed. As initially described SBERT, unlike TF-IDF and Doc2Vec creates more meaning between words that are further apart.

To investigate this further we suggest using more word embeddings as well as more in depth pre-processing pipeline. Further, we suggest using different evaluation metrics to draw more general conclusions.

While the results generally do not indicate lyrical significance, the goal of music recommendation is not just to provide music that a user likes but music that makes a user explore the new and unknown, educating a user on how the same message can be conveyed throughout different genres and periods of time. Recommending songs based on lyrics therefore alludes to the poetic side of music in which corresponding songs preserve meaning to some degree while allowing users to explore new acoustics.

References

Adiyansjah, Gunawan, A. A., & Suhartono, D. (2019). Music Recommender System Based on Genre using Convolutional Recurrent Neural Networks. *Elsevier*, 99-109.

Bogdanov, D., Haro, M., Fuhrmann, F., Gomez, E., & Herrera, P. (2010). Content-based music recommendation based on user preference examples. *ACM Conference on Recommender Systems. Workshop on Music Recommendation and Discovery* (S. 7). Barcelona: ResearchGate.

Ding, L., Zheng, N., Xu, J., & Xu, M. (2015). An Improved Content-Based Music Recommending Method with Weighted Tags . *International Conference on Multimedia Modeling*. Springer.

Musto, C., Semeraro, G., Gemmis, M. d., & Lops, P. (2016). Learning Word Embeddings from Wikipedia for Content-Based Recommender Systems . *European Conference on Information Retrieval* (S. 7). Bari: ResearchGate.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *EMNLP*, 11.