A search engine for musical covers

Piyush Singh
International Institute of Information Technology, Hyderabad
piyush.singh@research.iiit.ac.in

Abstract

With the advent of "democratized" platforms such as YouTube, SoundCloud, Spotify, etc., people worldwide are increasingly interested in listening to cover versions of popular songs. Sometimes the cover versions become more popular than the original song. However, none of these platforms currently provide an option to search the cover versions of the songs. Most of the platforms rank cover versions much lower in search results owing to unverified user handle. We present a method to solve this discoverability problem by mapping each song to a database of music metadata. Moreover, we can use this database to build a reverse search system to search for songs by humming the lyrics or the main tune.

1 Introduction

Several examples[1] exist where the cover versions of songs eventually become more popular than the original recorded versions. However, these covering artists struggle to make their content discoverable, especially in their initial stages. Only when a few of their covers go viral, do they become discoverable, however good their covers might be. This is because these platforms favor record labels over independent creators. For instance, there is a big debate about whether or not YouTube is biased against individual creators.[2] There is also the problem of copyright claims wherein almost all cases, the platforms favor the record label because of the Digital Millennium Copyright Act(Wikipedia contributors, 2021).

From the standpoint of the platform, the challenge is in maintaining a fair balance between channels that will easily generate at least some traffic/views versus independent creators, which may occasionally generate unusually high traffic. This points to an exploitation-exploration tradeoff, one of the popular paradigms in reinforcement learning and control problems (Ishii et al., 2002).

YouTube and SoundCloud do not allow users to select a song from a dropdown. This is because they are not exclusively for music but instead general video and audio platforms, respectively. Consequently, they do not maintain a database of music metadata internally. Even if they do, they cannot provide a dropdown because that will require a separate interface for music and non-music videos. This is one of the problems that YouTube has sought to address with its recently launched product, YouTube music.

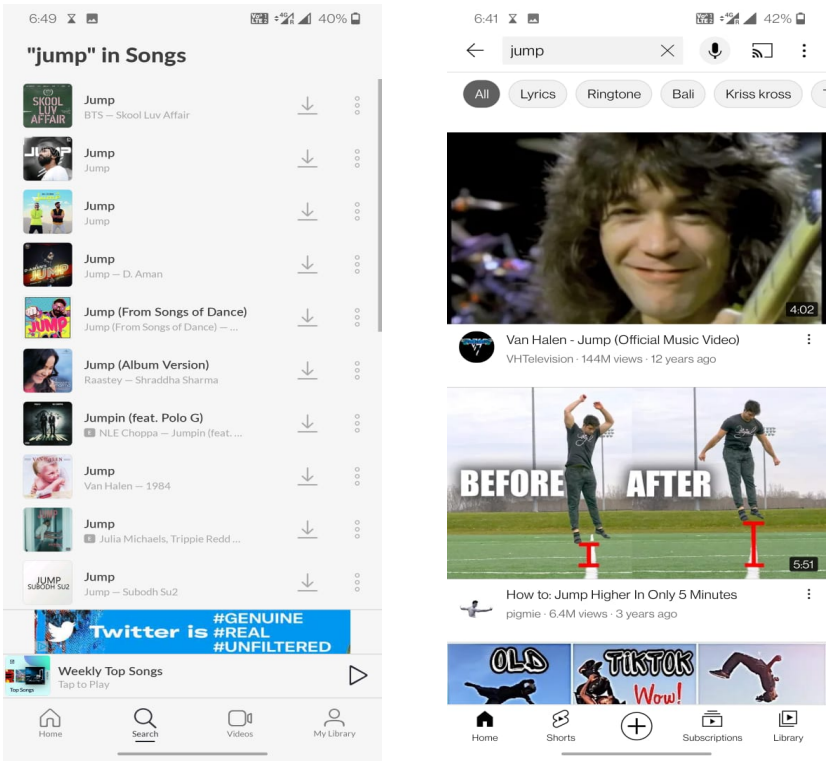
Music-only platforms such as Spotify and JioSaavn make a dropdown of song/album from a metadata DB. Still, they are mainly dedicated only to the official label version of music.

This article presents a design to map all the cover versions of a song to a unique item in the musical metadata database. We use this database to build a search engine that can search exclusively for cover versions of a specific song or cover versions of all songs of an album or artist etc. We also present a design to use this same database to build a reverse search where we can search for music through a user's voice. This is similar to the recently launched Hum-to-Search feature by Google.[3]

[1] For eg. this cover is more popular than this original.

[2] The golden age of YouTube is over: https://www.theverge.com/2019/4/5/18287318
How YouTube is rigged against independent video creators: https://medium.com/the-business-of-content/c47a7f033886
YouTubers and record labels are fighting, and record labels keep winning: https://www.theverge.com/2019/5/24/18635904

[3] Song stuck in your head? Just hum to search : https://blog.google/products/search/hum-to-search/



(a) JioSaavn (b) YouTube

Figure 1: A music-only platform can provide a dropdown menu from a music metadata DB (left), whereas a generic system offers all types of results (right)

2 Related Works

Much work has been done on the classification of mood, genre, etc., from music audio datasets. (Qin et al., 2013) present a Bag-of-Tones model for musical genre classification, which follows the conceptually similar idea of the bag-of-words model in natural language processing. They use Mel-frequency cepstral coefficients (MFCC)as features to cluster their dataset into a set of codewords referred to as ”tones”. Further they represent each song by a histogram of these tones and then use support vector machines (SVM) for the final classification step.

Figure 2: Pre-processing pipeline used by Qin et al.

3 Problem Description

3.1 Search engine for song covers

We need to create a database of original label recording songs and then map all covers of that song to a unique row in this database. This consists of two parts:

1. Create a database using the original recordings

2. Classify each musical cover to one of the items in this database

3.2 Hum to search

Once we have several audio recordings of a song, we can use this system to do a nearest neighbor (or more complicated) match of a non-expert user to the database entry for the song. This will require us to build a reverse search engine.

4 Insights

To build our database and search system, we can use the fact that songs generally have musical components that can be grouped. Also, songs generally have a definitive structure: intro followed by different verses and the same chorus multiple times followed by an outro. Hence the audio data of songs can be used to learn musical patterns, which can then be used to do the matching tasks for musical covers and non-expert song-hummings.

For our ease, we can start our work from an existing database of music metadata (without audio) such as MusicBrainz.[4]. We can fetch lyrics of all songs of this database from popular lyrics websites such as AZLyrics[5].

5 Solution

5.1 Building the database

To build the database of label music, we start with a music metadata DB to use this as a key to which covers can be mapped later.

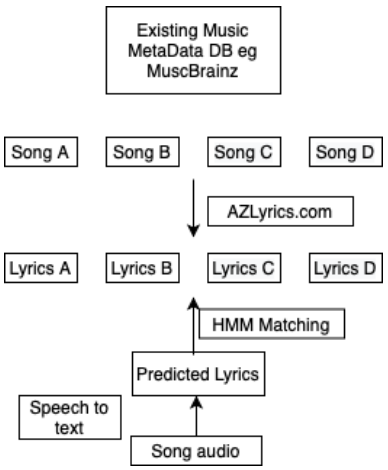


Figure 3: Matching record label audio to a unique entry in metadata DB

After this, we need to learn a vector representation of the song audio. For this, we can use the bag of tones approach similar to, (Qin et al., 2013) or we can also use deep learning.

5.1.1 Bag of Tones representation

We can use the following steps to learn the representation of any song audio:

1. Obtain the Mel spectrogram image of the song audio
2. Get the salient regions and features of the image using SIFT.
3. Cluster the features into a set of codewords
4. Find the nearest neighbor codeword for each region and append it to the histogram of that feature
5. Finally, the representation of the song audio is this histogram of codewords

[4] https://musicbrainz.org
[5] https://www.azlyrics.com

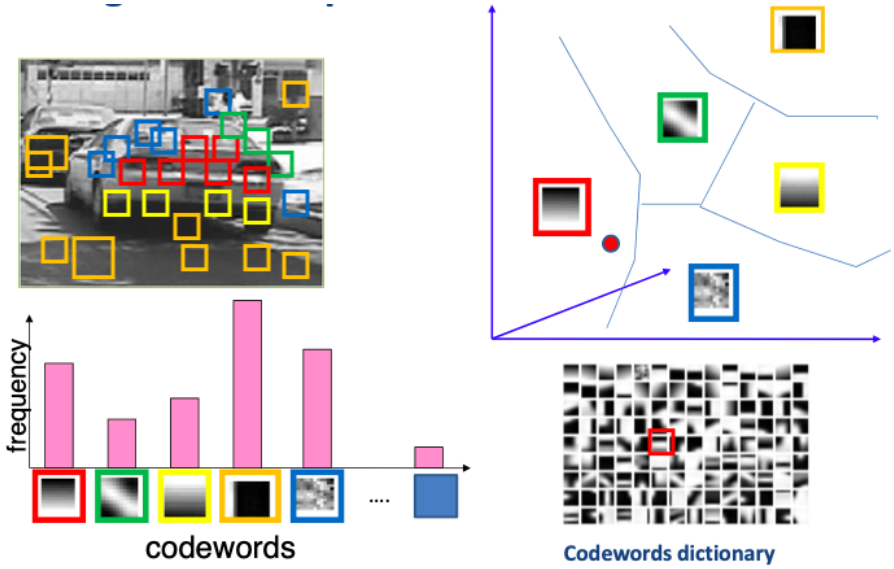


Figure 4: Bag of visual words representation of an image. We can use the Mel Spectrogram image of the song audio for this process. Image courtesy: CS231N slides

5.1.2 Representation using deep learning

Any discussion on modern-day representation learning is incomplete without deep learning. Deep learning has surpassed the traditional methods in representation learning for several modalities.

In the deep learning paradigm, autoencoder is a method to compress and decompress an original sample and learn using reconstruction error. Thus it learns the representation of any instance in its latent layer.

(Vaswani et al., 2017) have presented Transformers, which have been hugely successful in learning natural language and generating novel text in summarization, translation, and other NLP tasks. (Choi et al., 2020) have proposed a method to use transformers to learn the representation of song using audio. For this, we can follow these steps:

1. Transcribe the audio files into MIDI using "Onsets and Frames" framework(Hawthorne et al., 2018).
2. Represent the audio by a sequence of timesteps where each time step corresponds to a MIDI note or interval.
3. Train a transformer autoencoder on this audio.
4. Final representation of the song audio is the latent layer of the model.

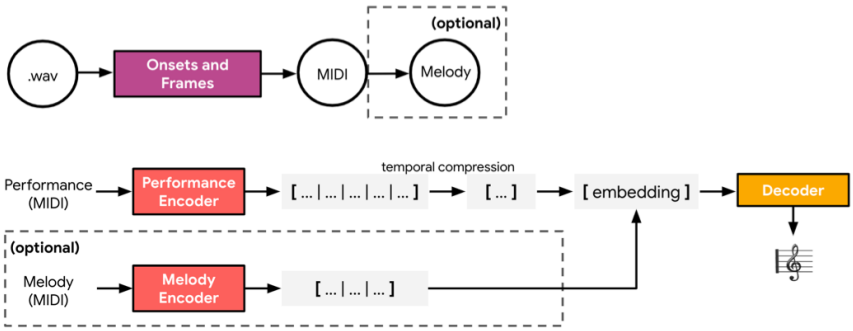


Figure 5: Architecture of transformer autoencoder by Choi et al.

5.2 Expanding the database to cover versions

We use the framework as presented above to learn the representation of the cover versions. We map each cover version to one of the items in the metadata DB using nearest neighbor match. In case of large search space, we can do approximate nearest neighbor using KD-Tree. Assuming that the cover versions sufficiently resemble the original in terms of musical features that we have learned, this matching should be near perfect.

We can use the database now to present cover-only results that we sought to generate initially.

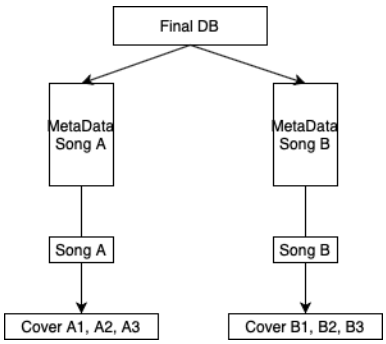


Figure 6: Database of label audio with metadata and mapped cover versions

5.3 Hum to search

Since we have one song mapped to several audios and all these versions' features, we can consider new audio from a non-expert user as a new version of an unknown song. We can obtain the feature vector for this new recording using the above method and find the nearest match. We can return this match as the song the user was looking for.

6 Evaluation

Search engines have popular evaluation methods such as PrecisionK, Mean Average Precision (MAP) and normalized discounted cumulative gain (NDCG) metrics which we can use for evaulation.

However, for most modern-day retrieval systems, K is usually set to a very small value (1 or 2) because of the impatient nature of internet users.

7 Conclusion

In this article, we presented a method to create a database of original and cover versions of music. We presented a way to map the cover version to the original song metadata. We also presented a simple process to solve the hum-to-search problem.

Learning the representation can be tricky in a real-life scenario and may require innovative loss functions and architectural changes. Moreover, with data such as audio, pre-processing becomes extremely important. So, the method will need to take into account such complexities.

The discoverability of cover versions will become more and more attractive as the reach of the internet increases and, subsequently, the number of small independent creators will increase. Hence systems like the one we discussed here will gain more attention.

References

Kristy Choi, Curtis Hawthorne, Ian Simon, Monica Dinculescu, and Jesse Engel. 2020. Encoding musical style with transformer autoencoders. In *ICML*.

Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. 2018. Onsets and frames: Dual-objective piano transcription. *ArXiv*, abs/1710.11153.

Shin Ishii, Wako Yoshida, and Junichiro Yoshimoto. 2002. Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural networks*, 15(4-6):665–687.

Zengchang Qin, W. Liu, and Tao Wan. 2013. A bag-of-tones model with mfcc features for musical genre classification. In *ADMA*.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Wikipedia contributors. 2021. Digital millennium copyright act — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Digital_Millennium_Copyright_Act&oldid=1056825909`. [Online; accessed 27-November-2021].