Music Video Clips Retrieval Based on Similarity

University of Piraeus - NCSR Demokritos
Machine Learning in Multi-Modal Data

First Author
Orfanos Stamatios
mtn2211
stamatisorfanos99@gmail.com

Abstract

With the rapid growth of online music platforms and video-sharing websites, the need for effective song recommendation systems has become increasingly important. In this paper, we present an interesting approach to finding the most similar song in a collection to a given song provided by the user. Our approach combines image classification and text classification techniques to extract meaningful features from songs. By analysing both visual and textual information in the song lyrics, we aim to provide more accurate and comprehensive recommendations to users. Experimental results demonstrate the effectiveness of our proposed approach, showcasing its potential for enhancing song recommendation systems in the modern era of multimedia content.

1 Introduction

In recent years, the widespread availability of online music platforms and video-sharing websites has led to an overwhelming amount of multimedia content being produced and consumed. As a result, there is an increasing demand for efficient and accurate systems that can assist users in discovering new songs and videos that match their preferences.

In this paper, we propose an approach that leverages image classification and text classification techniques to address the challenge of finding the most similar song in a collection to a given song provided by the user. By combining these two modalities, we aim to capture a more comprehensive representation of songs, enabling us to identify hidden patterns and similarities that may not be evident through traditional approaches.

Our approach starts by extracting visual features from song cover images using state-of-the-art image classification models. These visual features provide insights into the mood, genre, and artistic style associated with a song, which are crucial factors for users when seeking similar content.

Furthermore, we recognise the significance of lyrics in capturing the semantic meaning and thematic content of songs. Therefore, we employ text classification methods to extract relevant features from song lyrics, enabling us to consider the textual context when determining song similarity. By incorporating lyrics analysis into our approach, we can offer more personalised recommendations that align with the user's preferences and interests.

In summary, our proposed approach combines image classification and text classification techniques to provide a more comprehensive and accurate method for finding the most similar song in a collection. Through experimental evaluations and comparisons with existing methods, we demonstrate the effectiveness and potential of our approach in enhancing song recommendation systems, thereby empowering users to discover and explore a wide range of multimedia content that resonates with their individual tastes and preferences.

2 Data

In order to gather a data-set for our study, we employed a data collection method that involved obtaining the 90 most popular songs from Spotify.

Spotify, being one of the leading music streaming platforms, provided a reliable source for capturing a diverse range of songs that have gained significant popularity among users. To automate the process, we developed a script that utilised Spotify's API to retrieve the necessary video files for the identified popular songs.

2.1 Image data

When working with videos, it is possible to extract a significant amount of image data from each frame of the video. Videos are essentially a sequence of frames that are displayed at a certain frame rate, typically ranging from 24 to 60 frames per second (fps). Each frame represents a snapshot of the scene at a specific point in time. To extract image data from videos, we use frame sampling. Frame sampling involves selecting a subset of frames at regular intervals from the video. The amount of image data that can be extracted from videos depends on several factors, including the duration of the video, the frame rate, and the resolution of the frames. During the process of extracting frames from each video we realised that we had a great variety of duration for each video. That meant that a static sampling rate would create a imbalanced data-set, where the videos with greater duration would have double or even triple the amount of images. To combat this problem we set different sampling rate based on the duration of the music video.

In order to create the images from frames we use a couple of very useful libraries. The main library we used in order to achieve our goal was ffmpeg, that helped us get the basic information about the video like duration, frames per second and dimensions of frames.

1. Video duration bigger than **300sec/5min** we have a sampling rate of 0.5fps.
2. Video duration bigger than **200sec-300sec 3.3min-5min** we have a sampling rate of 1fps.
3. Video duration less than **200sec/3.3min** we have a sampling rate of 2fps.

Following this procedure we got a more balanced data-set for the images of each video. The result of this procedure are two folders train and test that include 90% and 10% of the data respectively. The only problem with this process is the fact that the frames do not have the same size in terms of height and width, but we are going to resolve this problem in the training of the model.

2.2 Text Data

Extracting text data from videos can be a more challenging task compared to image data extraction, as videos primarily consist of visual and auditory content. However, there are some methods that can be employed to extract text data from videos, albeit with certain limitations. In this case the library we used to extract the lyrics of the music video is Whisper and more specifically the Whisper English Base Model.

Some of the limitations we faced is that when using the Whisper Base Model we faced some problems. An example of such limitations would be some music videos of the hip-hop genre, where the addition of tuning to the voice would create problems to the Whisper model. In that case the model would confuse English with other languages and would start producing all of the lyrics in the other languages.

When using the model's transcribe method we were able to get the lyrics of each music video. The result of this method is a dictionary that contains the following metrics:

1. **id** : It represents the identifier or unique reference assigned to each segment or unit of speech recognized by the Whisper model.
2. **seek** : It indicates the seek position of the recognized speech segment in the input audio. This value denotes the starting point of the segment within the audio, allowing for precise alignment with the original audio source.
3. **start** : It represents the start time of the recognized speech segment in seconds. This attribute provides temporal information about when the recognized speech begins within the input audio.

4. **end** : It signifies the end time of the recognized speech segment in seconds. Similar to the ”start” attribute, the ”end” attribute provides temporal information about when the recognized speech concludes within the input audio.

5. **text** : It contains the transcribed text or speech-to-text output of the Whisper model. This attribute represents the recognized words or sentences corresponding to the input audio segment.

6. **tokens** : It represents the number of tokens used to generate the transcribed text. In models like GPT-3, text is processed in tokenized form, where each word or character is encoded as a token. This attribute provides insights into the length or complexity of the transcribed text.

7. **temperature** : It denotes the temperature parameter used during text generation. Temperature controls the randomness or diversity of the generated text. Higher values (e.g., above 1.0) produce more diverse outputs, while lower values (e.g., below 1.0) result in more focused or deterministic outputs.

8. **avg_logprob** : It represents the average log probability of the generated text. This attribute provides a measure of the confidence or likelihood assigned by the model to the generated transcriptions. Higher average log probability values indicate higher confidence in the transcriptions.

9. **compression_ratio** : It indicates the compression ratio achieved by the Whisper model. Compression ratio represents the reduction in size or representation achieved through speech recognition or compression techniques.

10. **no_speech_prob** : It signifies the probability of there being no speech detected in the input audio segment. This attribute provides a measure of the confidence or certainty of the model in identifying segments without any speech content.

Among the parameters provided by the Whisper model, the ”text” attribute holds significant importance when considering text classification tasks. Text classification involves the categorisation or labelling of textual data into predefined classes or categories. The ”text” attribute contains the transcribed text output generated by the model, representing the recognised words or sentences corresponding to the input audio segment. This transcribed text serves as the primary input for text classification algorithms, enabling the application of various natural language processing (NLP) techniques.

3 Models

In our approach, we adopt a comprehensive strategy by utilising different pre-trained models for each category of data, including image and text. This multi-modal approach allows us to leverage the unique capabilities of each model in capturing and analysing specific aspects of the data, thereby enhancing the overall accuracy and effectiveness of our system.

3.1 Image Classification Model

For image data, we employ a state-of-the-art image classification model pre-trained on the data-set we created. The model we are going to be using is ResNet-50 from Torchvision. ResNet-50 is a deep CNN architecture, comprising 50 layers, including convolutional layers, pooling layers, and fully connected layers. Its depth allows for the extraction of complex features and enables better representation learning.

ResNet-50 introduced the concept of skip connections and residual learning. These connections facilitate the propagation of gradients and help address the vanishing gradient problem, making it easier for the network to learn and optimise deeper layers. Residual learning allows the network to focus on learning the residual or the difference between the input and output of a layer, which aids in faster convergence and improved performance.

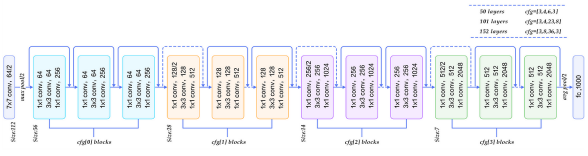

Figure 1: Resnet-50 Model Architecture

ResNet-50's depth and skip connections contribute to its excellent **generalisation** ability. It can learn high-level features that are transferable to various image recognition tasks, even when trained on different data-sets. This makes ResNet-50 a great choice for the classification task in this project.

In order to train the model we need to transform the frame images with the following steps:

1. Resize images: 224x224
2. Normalise mean: [0.485, 0.456, 0.406]
3. Normalise standard deviation: [0.229, 0.224, 0.225]

In this case, by unfreezing only the last linear layer, we are effectively discarding the classification aspect and obtaining the feature vector that captures the high-level representation of the input image data. This feature vector can be used for various downstream tasks such as similarity matching, clustering, dimensionality reduction, or feeding into another machine learning model for further processing or classification. It encodes the extracted features from the ResNet50 model in a compact and meaningful representation.

3.2 Text Classification Model

In the text classification task we use the BERT, which is a transformer-based model that refers to one of the pre-trained variants of BERT available in the Hugging Face Transformers library. BERT-base-uncased is built on the Transformer architecture, which employs a multi-layer bidirectional self-attention mechanism. It consists of 12 transformer layers with a total of 110 million parameters, making it a medium-sized BERT model.

BERT uses a WordPiece tokenization approach, where words are split into sub-words or tokens. The "uncased" aspect of BERT-base-uncased indicates that the model has been trained on lowercased text, treating uppercase and lowercase letters identically.

The resulting embedding from the last hidden state is a fixed-length vector representation of the input text. Each token in the input sequence is associated with a corresponding vector in the embedding. These vectors capture the contextual information, including the meaning and relationships between words.

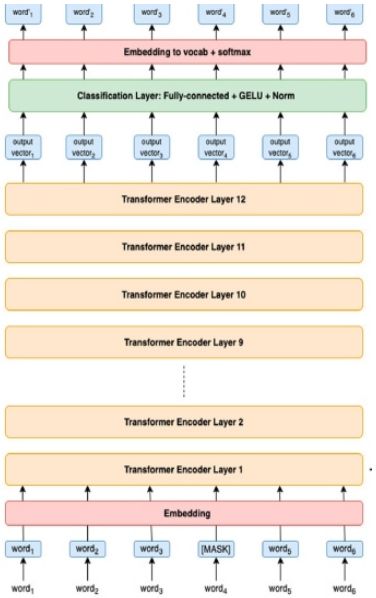

Figure 2: BERT-base-uncased Model Architecture

The dimensionality of the embedding depends on the BERT model variant, where in this case for the "bert-base-uncased" model, the embedding typically has a dimensionality of 768. The BERT embedding can be useful for various downstream tasks such as text classification, sentiment analysis, question answering, or any other task that involves understanding and processing textual data. By leveraging the contextualized representations learned by BERT, you can capture intricate linguistic patterns and semantics in the text, leading to improved performance in NLP tasks.

4 Training

For the training process we have to take into consideration the fact that each model was trained with different hyper-parameter configurations, we were able to explore a broader space of model behaviours and identify the settings that yielded the best performance for each specific task. This approach allowed us to optimise the models' capabilities and tailor them to the unique characteristics of the image and text data we were working with. The flexibility provided by adjusting hyper-parameters played a crucial role in achieving the desired accuracy and efficiency in our project.

After completing the training of the models we are able to calculate the **Cosine Similarity** between the created Feature Vectors and the test data.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 3: Cosine Similarity

4.1 Resnet-50 Training

Initially we have to import the data in a format that is going to be used by the model and create a data augmentation pipeline in order to get better results.

The data augmentations techniques for the training data are:

1. Resize image 224x224.
2. Apply affine transform on an image.
3. Apply Gaussian Blur.
4. Apply random horizontal flip.
5. Normalise the mean and standard deviation.

The data augmentations techniques for the testing data are:

1. Resize image 224x224.
2. Normalise the mean and standard deviation.

The **hyper-parameters** of the Resnet-50 models are the following:

1. Learning Rate: 0.0001
2. Epochs: 5
3. Batch size: 32
4. Loss Function: Cross Entropy Loss
5. Optimiser Function: Stochastic gradient descent

When training this model we have to take into consideration the fact that we have 90 different videos, which means that we expect to have some imbalanced results for some classes or videos in this case. In order to get a more representative result of the model training we are going to get the average accuracy of the top 10 classes of data. The results of the model training are the following:

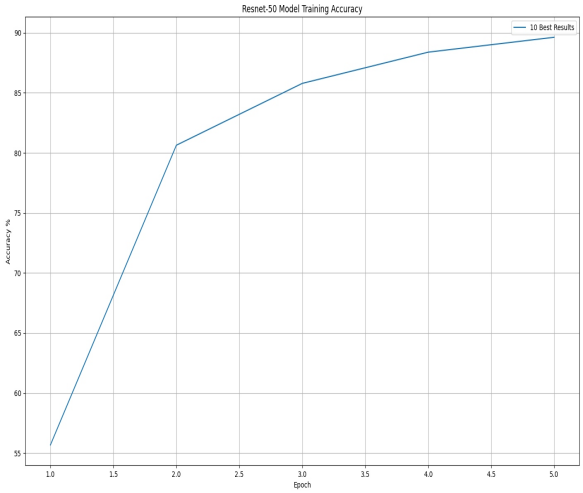

Figure 4: Resnet-50 Model Training Accuracy

When using the best 10 classes from the data we can understand that the results are close to 90%, which means that some videos have a very recognisable and different music video that is helpful for our model. Training the model allows us to fine-tune the weights of the pre-trained model based on your specific data-set. This fine-tuning process adapts the model to better represent the characteristics of your data, potentially improving the quality of the embeddings.

Of course using the linear layer as an embedding and applying cosine similarity can be more effective than directly comparing raw pixel values or intermediate layer activations. The linear layer acts as a feature extractor that maps high-dimensional input data to a lower-dimensional feature space where similarities are more meaningful. The cosine similarity metric, when applied to these embeddings, can provide a more reliable measure of similarity between images.

Obviously we want to get a multiple frames from different time-stamps with the aim of having a good representation of the video. More precisely we are going to take 10 frames from the test video and get the most similar results for each one of them.

In order to test the performance of the model and cosine similarity we use a frame from a popular song Star Boy by a popular artist The Weeknd.

Figure 5: Test Image - The Weeknd Star Boy

As we can see the cosine similarity based on the Feature Vector of the Resnet-50 Linear layer is quite successful, since we are able to retrieve similar results bsed on the colour of the background, being primarily dark blue background with neon colours.

4.2 BERT-base-uncased Training

In this case we have to import the text files and use the Bert-uncased tokeniser in order to create the tokens for the corpus of all the songs. In order to prepare the data for the model we have to follow the steps below:

1. Add padding.
2. Add truncation.
3. Set maximum sequence length to 512 tokens.

The next step in order to get the most similar results is to use the Pytorch's Cosine Similarity to the embeddings of each lyrics compared to the embedding of the test song. In this case we do not have the accuracy metrics like we did before, due to the fact we are not using a model but rather a similarity metric.

5 Conclusion

In addition to the utility and performance of pre-trained models, this project has successfully addressed the challenge of retrieval based on music video clips in a satisfying manner. By employing a multi-modal approach that incorporates image and text data analysis, we have effectively tackled the complexity of recommending similar videos to users. Using pre-trained models such as ResNet-50 for image classification and BERT-base-uncased for text processing, we have extracted meaningful features from music video clips. These features have allowed us to capture visual and thematic elements of the videos, enabling us to identify similarities and recommend relevant content.

By combining the outputs of the pre-trained models and implementing retrieval algorithms, we have achieved a robust and satisfactory solution for music video retrieval. The pre-trained models provide a strong foundation for feature extraction, while the retrieval algorithms efficiently match and recommend videos based on their similarities.

Furthermore, the seamless integration of these pre-trained models into our retrieval system has significantly reduced the need for extensive custom code implementation.Leveraging existing implementations and pre-trained weights has streamlined the development process and allowed us to obtain impressive results with minimal computational resources resurces.

In conclusion, this project has successfully solved the problem of music video retrieval by utilising pre-trained models and a multi-modal approach. The combination of image and text analysis, along with efficient retrieval algorithms, has enabled us to recommend similar videos to users in a satisfying manner.

References

Spotify

Hugging Face Transformers

ffmpeg

Hugging Face BERT Model

Whisper Base Model

Whisper

Whisper English Base Model