KV Multimedia Search and Retrieval

Exercise 1 Group E

|  |  |  |  |
| --- | --- | --- | --- |
| Ali Ayadi  Johannes Kepler University  K12042692@students.jku.at | Luca Della Mura  Johannes Kepler University  k12241884@students.jku.at | Ruo Li  Johannes Kepler University  k12247854@students.jku.at | Sara Scheucher  Johannes Kepler University  k1648069@students.jku.at |

ABSTRACT

This paper outlines the development of a rudimentary multi-modal music retrieval system. The goal of this paper is to assess the impact of varied representations of data and similarity measures within a retrieval system. Subsequently, the results retrieved by the developed systems will be evaluated for multiple metrics. The report explains the different representations used in text-based, audio-based, and video-based music retrieval systems, accompanied by an explanation of two different data fusion techniques. The implementation details are also briefly explained.

1. Introduction

The main goal of a music retrieval system is to find similar songs based on a specific query song provided by the user. Developing a retrieval system for music can be a challenging task due to the diverse nature of musical elements. Using text-based features as a representation may lead to matches based on similar lyrics, the retrieved songs however can belong to a completely different genre. Moreover, many songs don´t have lyrics. If audio-based representations are used instead, the question arises as to which features are used to measure whether a song sounds similar to a specific query song, because the similarity of music is highly subjective.

This paper describes the approach to implementing various music-retrieval systems and subsequently evaluates the systems Within the scope of this paper 11 distinct Music-retrieval systems will be implemented using different representations. The Dataset as well as the representations will be explained in Chapter 2. The Similarity of Songs will be compared by calculating the cosine similarity of different text, audio, and video-based features and a combination of some of those features. The retrieved results will be evaluated qualitatively according to the similarity to the queried song. The data set used for testing the 11 retrieval systems is a subset of the Music4All-Onion dataset which was kindly provided by the university.

1. Methodology

In this chapter, the dataset as well as the different features we chose for the retrieval systems and the methods to calculate the similarities between the query songs and the retrieved songs will be explained.

* 1. The dataset

Music4All-Onion is a large-scale, multi-modal music data set, which expands the Music4All-dataset with additional features and meta-data. The dataset provides many audio, video and metadata features of 109,269 music pieces. The data was extracted from the platform last.fm [1]

For the implementation of the music retrieval systems, the university provided us with a subset containing 10k songs. The provided data is presented in various TSV files. One for each representation as well as one which contains the song name, artist, and album info, and one file which contains the IDs and the corresponding genres of the song, which we will need for different evaluations of our retrieval systems. To read these files we use the Pandas library which interprets the files as data frames.

* 1. Text-based features

To compare songs based on the cosine-similarity of their lyrics we chose the following 3 representations which we used as input for our retrieval systems:

BERT is a language representation model. The name stands for Bidirectional Encoder Representations from Transformers.

It is trained on a collection of words that can capture rich contextual information in natural language. This allows BERT to understand the meanings of words in a given phrase. [2]

TF-IDF: this method is the mix of two components TF (Term Frequency) and IDF (Inverse Document Frequency), this combination yields a vector capturing the relevance of words in a document relative to a corpus. TF calculates the occurrence of a word for a given document on the other hand IDF calculates a word's importance for a given collection of documents the product of IDF by TF yields the relative importance of a word to a document within the overall corpus. [3]

Word2Vec is a popular technique for constructing word embeddings in the field of Natural language processing. A word embedding is a vector representation of a word. Two methods can be used to create such an embedding with the Word2Vec technique. The first one is the CBOW Model (Common Bag of Words) which takes the context of every word as input to predict the word. The second method is the Skip Gram model which takes the target word and predicts the context of the word and generates the representation of the target word in the process. Both techniques use shallow neural networks. [4]

* 1. Audio-based features

To calculate the similarities between songs with audio-based features we chose the following four representations:

Blf-Correlation: This dataset contains data about the correlation pattern of block-level features. [1]

Block-based audio features process the frames in terms of blocks. A block can be understood as a matrix where the number of columns is defined by the width of the block and the number of rows is defined by the number of frequency bins. Block-Level features have the advantage that each block comprises a sequence of several frames and therefore allows the extracted features to better capture temporal information. [5]

Blocks are built by selecting a collection of frames and then combining the feature values of all blocks into a representation of the song. [6]

Mfcc stats: This dataset contains a statistical summarization by concatenating the mean and flattened covariance matrix of the Mel Frequency Cepstral Coefficients (MFCCs). [1]

MFCCs have been widely used for speech recognition. They can represent the speech amplitude spectrum in a compact form which also captures perceptually important information. [7]

Ivec256: This dataset contains the I-Vector of MFCCs generated with a factor analysis procedure with 256 GMM components. [1]

Musicnn: This dataset contains features which were extracted using Deep Neural Networks (DNN). [1]

* 1. Video-based features

We only implemented one retrieval system with video-based features. We decided to use the Vgg19 representation.

Vgg19: This dataset contains data which was extracted using Very deep convolutional networks for large-scale image recognition. [8]

* 1. Cosine-Similarity

In our retrieval systems we used the cosine similarity to find the songs that are most similar to the query song.

In information retrieval the cosine similarity is a widely used metric especially in text-based retrieval. It calculates the similarity between two terms which are modeled as a vector. [9]

Ein Bild, das Schrift, Text, Handschrift, weiß enthält.

Automatisch generierte Beschreibung

* 1. Data Fusion

Early fusion: As an early fusion technique we used vector concatenation. The audio and textual attributes are concatenated into one vector before the similarity matrix is calculated. [10]

Early fusion: We combined one textual and one audio feature using the BERT and the musicnn representations. We take the embeddings, convert them into matrices and then horizontally stack the matrices which returns a fused matrix of the two embeddings. Then we calculate the evaluation metrics using the fused matrix.

Late fusion: As a late fusion technique the two features were combined by taking a weighted sum of the precalculated cos-sim-matrices of the two features. The resulting late fusion matrix is than used to calculate the relevant metrics.

1. Implementation

For implementing the assignment, the programming language Python is used as it is most suitable for data analysis and data science purposes. The coding environment used is Jupyter Notebook as it supports the programming language Python. We also used the libraries NumPy and pandas as well as Scikit-learn which provide us with different similarity functions. The code repository is hosted on the platform GitHub. The coordination and integration of code contributions of each team member is therefore ensured using Git.

To ensure that new functionality as well as new algorithms can easily be added to the music retrieval system in the future, a large focus is set on making the code modular.

The input of the system is a string which contains the name and the artist of the query track. The retrieval system should output a list of k similar songs with the title and the artist. To be able to make better use of the results in further calculations this output is saved in a Python dictionary. To keep the code modular and make it reusable we first implemented some basic functions in a separate Python file.

* 1. Cos-sim function

To calculate the cosine-similarity of the different representation of the lyrics, we created a wrapper function called “cos\_sim” that takes two Numpy-arrays as input and reshapes them to 2d arrays so they can be used in the cosine similarity function which is provided by the Scikit-learn library. The result of the cos\_sim function is the similarity score of the two arrays. The similarity function cos\_sim is then passed to the text-based, audio-based, and video-based retrieval functions.

* 1. Cosine similarity Matrix

For further performance improvements the similarity of all queries to all queries was precalculated once for each embedding and then re-used for all following calculations. This was achieved by transforming each embedding into a matrix form and then using matrix multiplication according to the formula described in chapter 2.5.

* 1. Random baseline

This retrieval system randomly selects n tracks and retrieves them, regardless of the given query track. we first shuffled the songs in a random order, so we get a different result each time the function is called. Then we excluded the query song from the data frame, so it does not appear in the result list. Afterwards, we retrieve the top N random songs and stored them in the result list.

* 1. Text-based retrieval functions

The text-based function takes as input parameters the similarity function which is cosine similarity, the feature representation, and the query song id. The function searches for the query song in the representation and extracts the row vector representing the song.

Then we create an array which is called similarities to store the

similarity scores. Afterwards, the text-based retrieval function iterates through all the rows in the dataset which contains the features. The similarities between the query-vector and track-vector are then calculated using cosine similarity. The song-id as well as the similarity-score are then saved in the similarities list. Afterwards, we sort the list in decreasing order of the similarity score and retrieve the ids of the 10 most similar songs.

* 1. Audio-based retrieval functions

For the audio-based retrieval systems, we used the same 3 query songs as we used for Task 1. We used the representations explained in chapter 2.3. For all these representations we calculated the cosine similarity. The results for all 3 query tracks are displayed in the provided main.ipynb file. The function takes the query-id, representation feature, number of tracks to retrieve as well as the similarity function as input, calculates the similarity and sorts the retrieved tracks in decreasing order of their similarity score.

* 1. Video-based Retrieval functions

The video-based function follows the same implementation as the audio-based function and takes a video representation as feature.

1. Evaluation Metrics

To evaluate all the 11 music retrieval system we calculated different evaluation metrics that are described in detail in this chapter.

5.1 Precision & Recall

Recall and Precision are measures that show how well a retrieval system retrieves relevant information. The precision is the ratio of true positives and total retrieved results. The recall is the ratio of true positives and the actual number of positives. [11]

In our music retrieval system, a retrieved song is considered relevant if it has at least one genre in common with the query song.

However, we have noticed that there are songs in the dataset which are not assigned to the correct genres. An example of this is the song "Somebody's Gotta Die" by "The Notorious B.I.G." which is categorized under the genre "death metal". However, this song is clearly a hip hop / rap song. Such a wrong categorization has an impact on the accuracy.

For the calculation of precision and recall we first obtained the genres of our retrieved results and put them into a list which consists of ids and genres of retrieved songs. This list is one of the parameters for the precision\_at\_k function. We also need k as a parameter as well as the id and genre of the query track. Afterwards we store the top k results into a variable and compare the genres of the top k results with the query genre and count how many of the retrieved results are relevant (a result is relevant if it has at least one common genre with the query track). Finally, we divide the relevant retrieved results by k.

For the calculation of the recall, we need one more parameter which is the whole genres dataset. The method calculates the number of retrieved relevant songs as well as the number of relevant songs in the whole genres dataset and then divides the relevant retrieved songs by all relevant songs.

These approaches were only computed to take one specific query song and calculate the precision and recall @ k.

For the calculation of Average Precision and Recall, which should be calculated over all possible query songs, we have decided to use a different method to speed up the calculation, as our previous approach took too much time. We have created a cosine-similarity matrix for this purpose, which has already been explained in more detail in section 3.2.

5.2 Genre diversity

Diversity is the opposite of similarity, in music recommendation systems (RS), diverse genres are needed to give the user a better and broader recommendation outcome so he can choose flexibly according to his preferences. To ensure large diversity a wide range of genres must be presented in the retrieved list by the RS. Different methods have been proposed by researchers to calculate the diversity such as calculating the distance between two elements i and j in the recommended list. Cosine similarity can be also used as a distance function to calculate the diversity. [12]

To calculate this evaluation metric, we implemented a method that first measures the diversity of genres of top k retrieved tracks given a query. It calculates how the genres are evenly distributed over k retrieved tracks. We can break down the formula into two parts:

1) Genre distribution:

We initialize a vector of zeros with a length equal to the number of all genres existing in the dataset, then for every genre found in each retrieved track, we add one to the corresponding genre position in the zeros vector divided by the number of genres of the retrieved track. So, this could be considered as the normalized attribution of each genre within the retrieved track genres to the overall genres in the dataset.

2) Normalize of the distribution:

We divide the resulting vector by the number of the retrieved tracks. To get the genre diversity we calculate Shannon’s entropy of the resulting vector. This is calculated by taking the negative sum of all the items of the resulting vector multiplied by its logarithm (base 2).

5.3 Genre coverage

This metric is, similar to the Genre Diversity described in the last

Chapter, a way to measure the quality of the music retrieval systems beyond using accuracy. For this research project the Genre Coverage is defined as the proportion of the number of unique genres present within at least one the top k retrieved tracks and the number of unique genres within the dataset itself.

5.4 nDCG

Normalized discounted cumulative gain (later referred in the current study as nDCG) evaluates the results based on graded relevance, i.e. nDCG assumes the users prefer the elements in the list of retrieved results to be presented in the descending order of their degree of relevance. Its calculation can be summarized in the following four steps:

1) Calculate the degree of relevance of each element (later referred in the current study as gain) in the list of retrieved results.

2) Assign the weight to the gain obtained from each element respectively according to its position in the list of retrieved results. Calculate the weighted sum of the gains. (later referred in the current study as DCG, i.e. Discounted Cumulative Gain)

3) Generate an ideal list of the retrieved results by reordering the elements in the list in the descending order of their gains. Calculated the DCG score for the ideal list. (later referred in the current study as iDCG, i.e. ideal Discounted Cumulative Gain)

4) Divide the DCG score of the list of the retrieved results by the iDCG score. [13]

The setting of the current study is as follows:

The function written to calculate the nDCG score for currently study only consider the top k elements in the list of retrieved results. When evaluation the results of this particular study, k is set to 10. (The metric is thus referred in the result section as nDCG@10).

The Sørensen–Dice coefficient of the genres is used to compute the gain, which adopts the following formula:



Gquery refers to the set of genres of the query track. Gi refers to the set of genres of the track used to calculate the gain. The genre information is obtained from the id\_genres\_mmsr.tsv dataset.

The weight to the gain was calculated with inverse logarithm of 2. The formulae adopted by the current study to calculate the nDCG are the following:

图示, 文本

描述已自动生成

1. Evaluation

6.1 Accuracy

We obtained the highest average precision of 0.4813 calculated over all query tracks using the musicnn feature combined the cosine similarity. The worst performing retrieval system based on precision was the early-fusion system combining the textual representation bert with the audio feature musicnn only achiving a precision of 0.3202.

We received the highest average recall computed over all query tracks again using the musicnn dataset in combination with cosine similarity which was 0.0022. Both early and late fusion obtained the lowest recall of only 0.0009.

As follows you can see the Precision-Recall Curve when varying the number of retrieved tracks between 1 and 100. To make this plot we used a sample of 100 query tracks and calculated the average precision and recall for each number of k. We used a random sample to reduce the computation time.

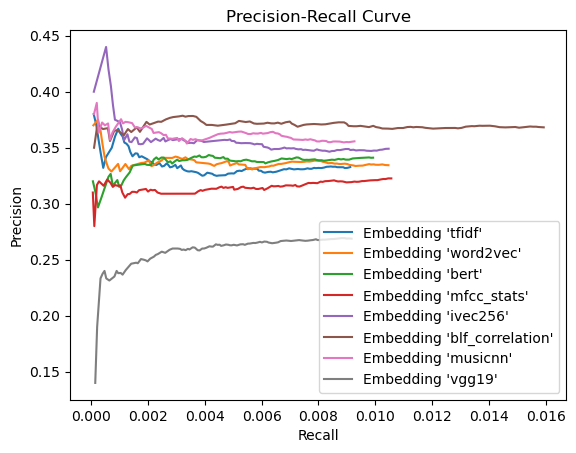


Figure 1: Precison-Recall plot

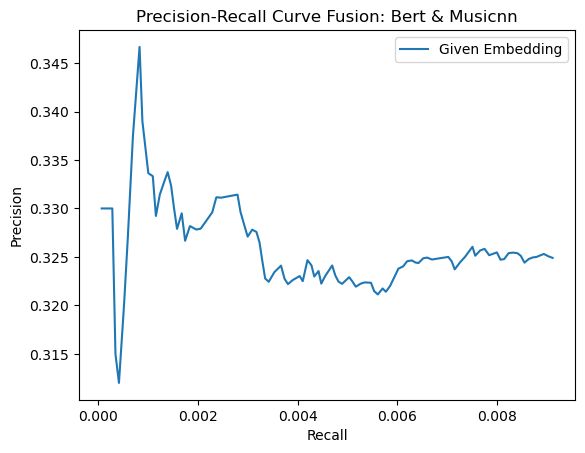


Figure 2: Precision-Recall Plot Early Fusion

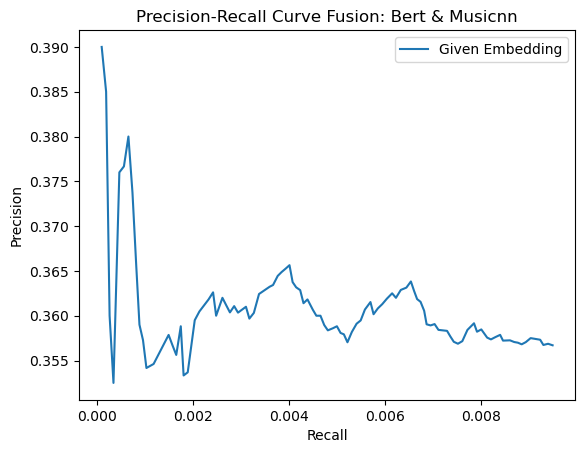


Figure 3: Precision-Recall Plot Late Fusion

**6.2 Genre Diversity@10**

The diversity formula is based on Shannon’s entropy which we  
would like to maximize this is equivalent to minimizing its negative. Consequently, a smaller diversity score indicates a more diverse retrieved list. We obtained the lowest diversity score of 4.7048 again using the musicnn representation. The highest score of 5.0797 was retrieved using the late fusion technique using BERT and musicnn as features. This could be interpreted that BERT is a powerful embedding system that led to retrieving tracks more similar to the query track and therefore very small genre diversity has resulted.

**6.3 Genre Coverage@10**

The next result section was concerned with the genre coverage@10 score. As mentioned in the methodology section, genre coverage assesses the proportion of unique genres covered in the retrieved list. Therefore, a higher genre coverage@10 score indicates a more diverse retrieved list. As can be seen from table 1, the highest genre coverage@10 score could be obtained with the audio-based retrieval system using the “musicnn” representation. Considering all implemented systems, the genre coverage only differs slightly. It ranges between 0.0343 and 0.0383. The lowest score was obtained using the text-based “bert” representation.

**6.4 nDCG@10**

Now let us shift our focus to the results concerning the nDCG@10 score. As mentioned in the methodology section, a larger nDCG@10 score indicates a better performance of the retrieval system as the lists obtained from the results are more similar to their ideal counterparts when ranking is taken into account. Due to time constraint and high computational cost, we were not able to obtain the final result from the two fusion based retrieval systems. As can be seen from table 1, the highest nDCG@10 score was obtained with the retrieval system implemented with the “MFCC” representation. Considering all implemented systems, 4 retrieval systems performed better than the random baseline, they are “MFCC”, “musicnn”, “blf-correlation” and “Vgg19.” (Ranked in descending order of nDCG@10 score.)

1. Conclusion

The results of the evaluation metrics show that the retrieval system using the musicnn dataset which includes features created with the help of deep neural networks in combination with cosine similarity achieved the best score for all metrics except for the nDCG.

We already expected that one of the audio-based representations would perform best in terms of precision and recall, as these metrics are measured against the same genres and usually the assignment to music genres is not based on text or music video features but on audio features. We also note that combining the best performing audio feature with a textual feature using an early fusion technique lowered precision and recall immensely.

Results: (rounded to 4 digits)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Avg. Precision@10 | Avg. Recall@10 | Avg. nDCG@10 | Avg. Coverage@10 | Avg. Diversity@10 |
| Mfcc stats | 0.4289 | 0.0018 | 0.1611 | 0.0382 | 4.8112 |
| Blf-Correlation | 0.4098 | 0.0017 | 0.1504 | 0.0382 | 4.8942 |
| Ivec 256 | 0.4344 | 0.0019 | 0.1184 | 0.0382 | 4.9036 |
| musicnn | 0.4813 | 0.0022 | 0.1525 | 0.0383 | 4.7048 |
| Tf-Idf | 0.3794 | 0.0014 | 0.1201 | 0.0377 | 4.9743 |
| Word2Vec | 0.3851 | 0.0017 | 0.0892 | 0.0378 | 4.8492 |
| Bert | 0.4192 | 0.0020 | 0.1201 | 0.0343 | 4.8448 |
| Random-Baseline | 0.3274 | 0.0010 | 0.1205 | 0.0373 | 5.0708 |
| Vgg19 | 0.3278 | 0.0010 | 0.1462 | 0.0373 | 4.9715 |
| Early-Fusion  (Bert, Musicnn) | 0.3202 | 0.0009 |  | 0.0373 | 5.0625 |
| Late-Fusion  (Bert, Musicnn) | 0.3301 | 0.0009 |  | 0.0373 | 5.0797 |

Figure 4: Results (averaged over all query tracks)

8. Figures & Tables

[Figure 1: Precison-Recall plot 4](#_Toc156942512)

[Figure 2: Precision-Recall Plot Early Fusion 4](#_Toc156942513)

[Figure 3: Precision-Recall Plot Late Fusion 5](#_Toc156942514)

[Figure 4: Results (averaged over all query tracks) 1](#_Toc156942515)

9. REFERENCES

[1] Moscati, M., Parada-Cabaleiro, E., Deldjoo, Y., Zangerle, E., & Schedl, M., „Music4All-Onion“, doi: 10.5281/zenodo.6609677.

[2] J. Devlin, M.-W. Chang, K. Lee, und K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. arXiv, 24. Mai 2019. Zugegriffen: 16. Januar 2024. [Online]. Verfügbar unter: http://arxiv.org/abs/1810.04805

[3] V. Jayaswal, „Text Vectorization: Term Frequency — Inverse Document Frequency (TFIDF)“, Medium. Zugegriffen: 16. Januar 2024. [Online]. Verfügbar unter: https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d

[4] D. Karani, „Introduction to Word Embedding and Word2Vec“, Medium. Zugegriffen: 16. Januar 2024. [Online]. Verfügbar unter: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

[5] K. Seyerlehner, G. Widmer, M. Schedl, und P. Knees, „Automatic Music Tag Classification Based On Block-Level Features“, Juli 2010, doi: 10.5281/ZENODO.849724.

[6] H. Eghbal-Zadeh, B. Lehner, M. Schedl, und G. Widmer, „I-Vectors for Timbre-Based Music Similarity and Music Artist Classification.“, in *ISMIR*, 2015, S. 554–560.

[7] B. Logan, „Mel frequency cepstral coefficients for music modeling.“, in *Ismir*, Plymouth, MA, 2000, S. 11.

[8] K. Simonyan und A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition“. arXiv, 10. April 2015. Zugegriffen: 16. Januar 2024. [Online]. Verfügbar unter: http://arxiv.org/abs/1409.1556

[9] F. Rahutomo, T. Kitasuka, und M. Aritsugi, „Semantic cosine similarity“, in *The 7th international student conference on advanced science and technology ICAST*, 2012, S. 1.

[10] A. Depeursinge und H. Müller, „Fusion Techniques for Combining Textual and Visual Information Retrieval“, in *ImageCLEF: Experimental Evaluation in Visual Information Retrieval*, H. Müller, P. Clough, T. Deselaers, und B. Caputo, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 95–114. doi: 10.1007/978-3-642-15181-1\_6.

[11] K. M. Ting, „Precision and Recall“, in *Encyclopedia of Machine Learning*, C. Sammut und G. I. Webb, Hrsg., Boston, MA: Springer US, 2010, S. 781–781. doi: 10.1007/978-0-387-30164-8\_652.

[12] M. Kuanr und P. Mohapatra, „Assessment Methods for Evaluation of Recommender Systems: A Survey“, *Found. Comput. Decis. Sci.*, Bd. 46, Nr. 4, S. 393–421, Dez. 2021, doi: 10.2478/fcds-2021-0023.

[13] S. Büttcher, C. L. A. Clarke, und G. V. Cormack, *Information retrieval: implementing and evaluating search engines*. Cambridge, Mass: MIT Press, 2010.