Indexing and Retrieval of Scores by Humming based on Extracted Features

Hilda Romero-Velo, Susana Ladra, José R. Paramá, and Fernando Silva-Coira

Universidade da Coruña, Laboratorio de Bases de Datos, Faculty of Computer Science, 15071 A Coruña, Spain
Universidade da Coruña, Centro de Investigación CITIC, 15071 A Coruña, Spain
Correspondence: h.rvelo@udc.es

DOI: *https://doi.org/10.17979/spudc.000024.06*

Abstract: In order to be able to conduct searches over large collections of music scores with queries provided in audio format, this article considers recent literature in the field and proposes an implementation to extract specific features from music pieces. Afterwards, we index those features using modern Lempel-Ziv (LZ)-based data structures. These data structures take advantage of the intrinsic repetitiveness within music to reduce space consumption and, at the same time, to index the information optimizing the search time per query. Furthermore, taking advantage of this property-based representation framework, which does not depend on the way the music is portrayed, we enable the possibility to perform melodic searches by simply providing a query audio. This research branch is known as "query by humming" and has commonly been applied to audio sources. A preliminary study for its application in other forms of music representation is presented in this research.

1 Introduction

This paper presents a proposal to perform searches over large score collections using queries provided in audio format. Literature offers an extensive list of studies in the field of music information retrieval. Recently, Zhu et al. (2022) published an article that proposes a new approach: extracting three features describing a composition directly from the score and then, in their scenario, indexing them with the Elasticsearch search engine. However, this paper defines the theoretical framework for obtaining such characteristics, but does not specify the practical procedure of how to obtain them. In this article, we provide an implementation that allows such features to be derived. Then, those features are stored and indexed using modern Lempel-Ziv (LZ)-based data structures (Fariña et al., 2019; Kreft and Navarro, 2013). More specifically, from this family of auto-indexes, we use those based on LZ77 and LZ-End. These data structures take advantage of the intrinsic repetitiveness of music to save space consumption and, at the same time, to index the information so as to optimise the search time per query. Within this context, the ability to perform approximate searches is of utmost importance, since the query (audio excerpt) may not necessarily reflect exactly the same specifications as the original score does.

The most common approach is to take a specified audio query (analogue signal) and look at the stored audio data to find its equivalents. On the other hand, Zhu et al. (2022) depart from queries that specify the notes of the fragment to be compared on a score, which, nevertheless, is not very practical. This document explores the feasibility of offering a new querying method based on audio search. Contrary to widely used solutions such as Shazam (Wang,

2003), which requires the audio query to be a replica of a fragment of the original song's audio, or like Google's proposal (Kumar, 2020), which works with the fingerprint of each tune, our searches are independent of the format in which the original piece is collected, allowing queries over collections of scores. Therefore, by applying the principle of feature extraction on the query audio itself, score searches are enabled in the field referred to as "query by humming".

For this study, we work with 1,275 scores obtained from the Folkoteca Galega[1] under the MusicXML format. The corresponding melodic sequences are inferred from each one of them, and afterwards the proposed features are extracted, resulting in a set of 1,686 melodies to be indexed in our system. These three characteristics are the chromatic distance as well as the diatonic interval between notes, and the rhythm difference ratio between figures. In order to easily manipulate the scores, a conversion from MusicXML[2] format to **kern format is made using the Humlib library (Sapp, 2023), and then the previously mentioned features are extracted through the toolkit provided by Humdrum (Huron, 2022).

A similar approach is followed for queries generated from audio. First the audio is transformed into Musical Instrument Digital Interface (MIDI[3]) format. This task is a complex domain of study by itself, and even though we are dealing with a simple case (conversion of a monophonic sound of a single instrument), inaccuracies can be raised during the translation process between formats. For this entry, the *audio-to-midi* converter developed by Spotify is used (Bittner, 2022). Once the query is obtained in MIDI format, we transfer it to **kern representation using the Humdrum tools and, then again, we extract the features corresponding to that fragment to run the query against our index.

2 Previous Concepts

In this section, we will review the background knowledge and define certain aspects needed to understand the course of the case study. We will first describe the features to be extracted from the scores and then we will consider the musical representation formats involved.

2.1 Features

Three defining characteristics of a musical piece are extracted from the scores:

- **Chromatic Distance:** The first feature to be obtained is the chromatic distance between notes. This is calculated according to the chromatic scale, which consists of 12 notes. We start measuring from zero and proceed by counting the exact distance in semitones between the two notes involved. For instance, from C to high C there would be a distance of 12 semitones.
- **Diatonic Distance:** This second feature describes the distance between notes according to their position in the diatonic scale. The diatonic scale is formed by 7 notes. An example would be the natural scale of C major: C, D, E, F, G, A, B. In this case the distance is calculated starting at 1. Therefore, the unison, i.e. the distance between two consecutive notes which have the same pitch (for instance, a F followed by another F), in this scale would be represented by 1, whereas in the chromatic scale it would be marked as 0. Bringing the previous example, the diatonic distance from C to high C would be 8, which is commonly known as an octave.
- **Rhythm Ratio:** The rhythm of the piece is shown by indicating the ratio of difference in duration between a note and the preceding figure. In Figure 1, we observe that the value given to each figure doubles in length as we climb up the tree. Thus, a whole note is equivalent to 2 half notes, 4 quarter notes and 8 eighth notes.

1 *https://folkotecagalega.gal/pezas*
2 *https://www.musicxml.com/*
3 *https://www.midi.org/*

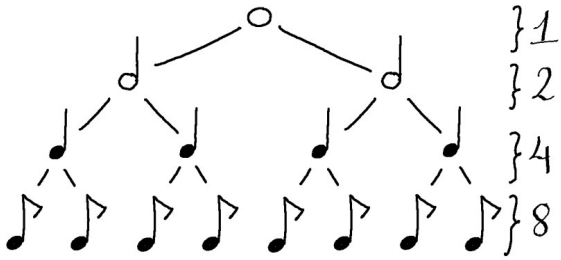

Figure 1: Rhythmic Figures Tree with Numbered Equivalences from 1 to 8

In the example of Figure 2, we distinguish 4 notes inside a score, accordingly, we can compute 3 distances. Those intervals within the notes are marked as 1, 2, 3 beneath them. Taking a piano as our perspective for the computation, the calculus of the chromatic distance considers the black notes and starts calculating at 0. Meanwhile, for the diatonic distance of this example we do not take into account the black notes, we only reckon the white ones and start counting at 1. The chromatic distance would be 1 - 4 - 3 respectively, yet the diatonic distance would be 2 - 3 - 3. We can verify, then, that, although the way of obtaining each feature is different, we can obtain the same values in certain cases, as in the last interval in the example. The rhythmic description according to the figure ratio is 1/2 - 2 - 1/2 since the second note is half as long as the first, the third note twice as long as the second and the fourth one half as long as the third (refer to Figure 1).

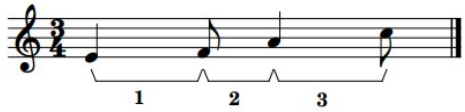

Figure 2: Score Excerpt with Highlighted Intervals 1, 2 and 3

2.2 Representation Formats

Both MusicXML and **kern (Huron, 2022) are schemas that allow the representation of scores with text notation. While the former focuses on rendering, the latter works with a functional format aimed at facilitating score manipulation and analysis. Thus, in this article we opted to rely on the **kern format in order to handle feature management and extraction.

On the other hand, MIDI files are designed to record events. Those events are messages that denote a specific occurrence regarding certain settings. Such files are useful for device-to-device communication and for synthesising the instructions into audio. However, they are not intended to describe scores. Some properties of these MIDI files that allow defining and setting the tempo, and hence defining the rhythmic figuration, may be misformed, mainly when performing a conversion from audio (due to the difficulty in inferring these parameters).

2.3 Lempel-Ziv (LZ)

The idea of Lempel-Ziv compression is based on looking for repeated sequences within the data. When found, this sequence is replaced by a reference to the location of the first sequence along with the length of the repeated pattern. If there are no such occurrences, each byte is written as a literal.

Example of LZ compression: Let the sequence to be compressed be [ABCBCBC]. LZ would phrase this stream as [ABC2:4]. The notation 2:4 indicates that starting from 2 positions behind, the next 4 characters are to be repeated. Therefore, the decompression process would be the following (the character to be written next in the string is highlighted in bold): [A**B**C] - [AB**C**B] - [ABC**B**C] - [ABCB**C**B] - [ABCBCBC].

In this scenario, however, there are different implementations depending on certain factors. A key point in compression is the "historical" window and the "future" window. In other terms, how much of the already processed data we store in memory to check if any future pattern is repeated, and how much ready-to-process data we keep in memory to seek those sequences in the already processed data. Another aspect to consider is the way a pattern is referenced and when should this happen, since the space needed for referencing should be less than the space required for storing the literals alone.

3 Implementation

When extracting features, we need to consider that there are two different source formats: the original score in MusicXML and the queries in audio format. For our particular scenario, we will assume only the Waveform Audio Format (WAV). In the following, we will discuss each of the cases.

Scores

The Folkoteca Galega is a web portal that gathers more than 1,200 pieces of traditional Galician music. It is classified according to the type of composition (jota, muiñeira, pasodoble, polka, etc.). For each song, it provides the score in MusicXML format, as well as in MIDI and PDF. Its main collaborator is PuntoGal, but any contribution is appreciated. Different associations and individuals related to the folk world have submitted scores to this collection.

Taking this corpus from the Folkoteca Galega in MusicXML, we use the *musicxml2hum* tool provided by Humlib (Sapp, 2023) to obtain those same scores in **kern format. We then proceed to remove information from the composition that is not required for defining the characteristics (e.g. dynamics). Afterwards, we extract the different melodic lines (voices). All of this is done by using the *extractx* command, also provided by Humlib. Thus, the 1,275 scores are fragmented into 1,686 melodic lines, i.e. 1,686 files.

After this preliminary preparation, the *humsed* command is used to remove the grace notes (ornaments). We use the *mint* command for the diatonic analysis and the *beat* command for the rhythmic analysis, both from the Humdrum toolkit. Meanwhile, to perform the chromatic analysis, we need to convert the **kern file into **semits, that is, into numerical semitone representations of the pitch (*semits* command). We must also remove rests so that they do not interfere with the calculation (once again with the *humsed* command). Finally, we can compute the semitone offsets with the *xdelta* command plus the *-s ^=* option to prevent the bar lines from interfering.

Having obtained all the features, for the melodic ones (chromatic and diatonic) we generate the files with the corresponding documents to be indexed according to the specification of the LZ implementation proposed by Fariña et al. (2019).

Audio Query

The Spotify basic-pitch converter is used to transform audio queries in WAV format to MIDI. However, even with a clear audio source, anomalies are generated during

the conversion process which disables exact searches. Merely to rectify this situation, it is therefore natural to conduct the research towards approximate searches. Nevertheless, for the simplicity of these first stages, after conversion, the MIDI files are manually polished (e.g. by removing phantom notes and harmonics). Even so, it is noted that the length of the notes is not properly transformed, as expected from the earlier observations regarding the misformation of MIDI files. It is precisely the inaccuracy of the transformation that causes some real notes to be perceived as grace notes which leads for them to not be taken into consideration during feature extraction.

Once the audios have been converted to MIDI, we use Humdrum's *mid2hum* command to get the **kern queries. Thereafter, feature extraction is continued in the same way as for the scores previously. Afterwards, they are adjusted to the requested format by the LZ implementation. The recovery of the expected number of occurrences is verified.

4 Experimental Evaluation

For this case study our aim is to test the proper indexing of the extracted features by running 20 exact search queries. Furthermore, we will verify the robustness of the implementation framework.

Evaluation Environment

The experiments were conducted on a computer with an Intel(R) Core(TM) i7-10750H and 16GB of DDR4 RAM with a Microsoft Windows 10 Pro operating system. The Docker version used was 4.17.0. For this preliminary research we took as a starting point the original corpus of 1,275 documents held in the Folkoteca Galega[4]. The features extracted from this collection yield 1,686 documents and occupy a total of 598.65 KB. Features have been adapted and transferred to the required format and layout in the test framework configuration provided in the form of Docker image with an implementation of LZ-based indexes by Fariña et al. (2019).

Each query is generated by recording in audio format a piano performance of the fragment to be searched. A total of 20 queries of varying lengths are collected. The displayed times reflect the average of 10 runs for the queries.

Results Analysis

Results indicate that the query time depends on the number of occurrences to be retrieved (Tables 2 and 3). For both chromatic and diatonic features, LZ-End offers a lower average recovery time per occurrence and a smaller standard deviation than LZ77, at detriment of a higher space requirement (Table 1). The difference between LZ77 and LZ-End lies in the segmentation of the repeated strands. While LZ77 opts for long repeating strands, LZ-End chooses to frame the repeats in smaller chains. Thus, LZ-End requires more space but offers faster substring extraction.

We also observe that the first four queries do not retrieve any occurrences even though they should. This is caused by the incorrect MIDI file formation during audio conversion. In this particular case, the necessity of moving towards an approximate

[4] *https://folkotecagalega.gal/pezas*

Table 1: Comparison of Sizes (KB) and Times per Occurrence (ms)

Index	Size	Feature	Average Time per Occurrence	Standard Deviation
LZ77	344.43	Chromatic	0.302	0.595
		Diatonic	0.467	0.907
LZ-End	408.43	Chromatic	0.205	0.379
		Diatonic	0.294	0.676

Table 2: Diatonic Results per Query ordered by Number of Occurrences

Number	Query Length	Occurrences	LZ77 Time (ms)	LZ-End Time (ms)
1	24	0	9.86	4.51
2	34	0	2.63	1.89
3	24	0	1.87	1.20
4	31	0	2.40	1.53
5	15	1	3.55	2.96
6	16	2	3.83	0.87
7	25	2	1.84	1.12
8	20	2	1.45	1.02
9	31	2	2.66	1.54
10	13	4	2.94	2.15
11	17	10	1.34	0.88
12	2	57	0.26	0.25
13	3	1,781	4.56	4.24
14	5	2,461	6.17	5.96
15	4	3,857	9.54	9.47
16	4	4,611	11.65	11.39
17	1	5,414	13.15	12.48
18	3	7,867	19.11	19.97
19	2	16,450	40.11	38.25
20	2	36,263	89.73	84.61

search is reflected. Lewenstein (2013) proposes an alternative to enable approximate searches on LZ indexes, which would overcome the loss of information in translations.

Altogether, both implementations, LZ77 and LZ-End, perform at a high level. They are compact, fast, and, consequently, efficient. The incidence is that they do not solve some queries properly due to the different casuistry involved. We, therefore, conclude that the approach should be directed towards approximate searches.

5 Conclusions

This paper introduces a preliminary study to offer score retrieval by humming. In order to take advantage of musical patterns, a feature extraction framework is established to connect the different representation formats. Thereby, the indexation takes place to exploit the intrinsic repetitiveness of the music.

After studying the causalities of some occurrences, it is concluded that the next step should be working with approximate searches in order to achieve the flexibility that

Table 3: Chromatic Results per Query ordered by Number of Occurrences

Number	Query Length	Occurrences	LZ77 Time (ms)	LZ-End Time (ms)
1	24	0	1.56	1.15
2	34	0	2.32	1.48
3	24	0	1.78	1.26
4	31	0	2.06	1.47
5	15	1	1.00	0.71
9	31	1	2.41	1.47
6	16	2	1.05	0.74
7	25	2	1.66	1.08
8	20	2	1.67	1.30
10	13	4	0.84	0.65
11	17	6	1.20	0.98
12	2	45	0.31	0.22
14	5	674	2.84	2.29
13	3	836	2.97	2.12
15	4	1,074	3.81	2.82
16	4	1,645	5.58	4.16
18	3	2,797	9.51	7.13
17	1	5,061	15.72	12.16
19	2	10,564	35.08	25.40
20	2	15,086	50.16	37.48

the system requires. The impact of the issues derived during the translation process from audio to MIDI could be minimize if we focus on approximate search indexes such as those proposed by Moshe Lewenstein in "*Orthogonal Range Searching for Text Indexing*" (Lewenstein, 2013).

Acknowledgements

Work funded by: CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Consellería de Cultura, Educación, Formación Profesional e Universidades and the Galician universities for the reinforcement of the research centres of the Galician University System (CIGUS), 80% through FEDER funds, Galicia Operational Programme FEDER 2014-2020, and the remaining 20% by the "Secretaría Xeral de Universidades" (Grant ED431G 2019/01), Xunta de Galicia/FEDER-UE [ED431C 2021/53]; Ministry of Science and Innovation [PID2020-114635RB-I00; PDC2021-120917-C21; PDC2021-121239-C31; PID2019-105221RB-C41; TED2021-129245-C21].

Bibliography

R. Bittner. Meet basic pitch: Spotify's open source audio-to-midi converter, jun 2022. URL *https://engineering.atspotify.com/2022/06/meet-basic-pitch/*.

A. Fariña, M. A. Martínez-Prieto, F. Claude, G. Navarro, J. J. Lastra-Díaz, N. Prezza, and D. Seco. On the reproducibility of experiments of indexing repetitive document collections. *Information Systems*, 83:181–194, jul 2019.

D. Huron. The humdrum toolkit for computational music analysis, 2022. URL *https://www.humdrum.org/index.html*.

S. Kreft and G. Navarro. On compressing and indexing repetitive sequences. *Theoretical Computer Science*, 483:115–133, 2013. Special Issue Combinatorial Pattern Matching 2011.

K. Kumar. Song stuck in your head? just hum to search, oct 2020. URL *https://blog.google/products/search/hum-to-search/*.

M. Lewenstein. *Orthogonal Range Searching for Text Indexing*, pages 267–302. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

C. S. Sapp. Humlib: Humdrum data parsing library in c++, aug 2023. URL *https://humlib.humdrum.org/*.

A. Wang. An industrial-strength audio search algorithm. *Columbia.edu*, 2003. URL *https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf*.

T. Zhu, R. Fournier-S'niehotta, P. Rigaux, and N. Travers. A framework for content-based search in large music collections. *Big Data and Cognitive Computing*, 6(1), 2022.