An approach to sheet music information retrieval

Yulia Korukhova
Computational Mathematics and Cybernetics faculty
Lomonosov Moscow State University
Moscow, Russia
yulia@cs.msu.ru

Aiken Tolybayeva
Computational Mathematics and Cybernetics faculty
Lomonosov Moscow State University
Moscow, Russia
aikesha28@gmail.com

***Abstract*—The paper presents a sheet music retrieval system. A search query is written in MusicXML format. MusicXML is a very popular format for sheet music, supported by various music notation software. With the appearance of large electronic collections the problem of music retrieval becomes particularly important. Known information retrieval methods are not directly applicable for this particular domain: the task differs from the structured retrieval in texts because, for example, two sheet music files with totally different symbolic content may represent the same composition written in different keys or arranged for different instruments.**

Our search starts from a fragment containing a melody or a part of a composition and the goal is to find the whole composition in a digital collection. The collection should be preliminary indexed and the index is stored as database. For MusicXML files we propose a software tool that performs indexing automatically. For other formats the indexing process can be performed manually and database is extended by means of SQL–queries. To perform a fuzzy search we mark the longest common subsequence in the query and index item and then to remove their differences using automated reasoning method called rippling. The appropriate rewriting rules and differences annotation techniques for musical notation are implemented in prototype sheet music retrieval system.

***Keywords*—music information retrieval, automated reasoning, rippling**

I. Introduction

Nowadays with wide spread of networks, computers and electronic devices, a lot of data, information and knowledge is stored in electronic form for almost every domain. That includes sheet music: many of that is now presented in electronic collections[1]. Such collections are often quite large and information retrieval tools are required to work with them. In addition to traditional search tools that are able to find an object by it's name and/or it's author the retrieval of a piece that contains a given fragment is required. This task has been solved for audio fragments in Shazam application[2], for humming queries and melody fragments in Musipedia [1]. However the search of given sheet music corresponding to a fragment is still a challenging task. From one point of view it is similar to information retrieval for texts, but it has several specific issues. First of all, there is no strict analogy with words in texts, that can be determined syntactically. Also the same music piece can have different symbolic representations, being transposed or arranged for another instrument. In this paper we are presenting a context-based approach to sheet music retrieval.

The paper is organized as follows: first we describe a task of music scores information retrieval, next will discuss an appropriate internal representation, that consider the particular features of the domain, then will pass to index organization. Next we describe our search algorithm and give a short example of it's work. Finally the conclusions are drawn.

II. Music scores retrieval task

As an input data we take a music score fragment, written in MusicXML format [2]. This format has been chosen because it is widely supported, it can be used in more than 150 applications [2] including music notation software like MuseScore[3], Finale[4], Guitar Pro[5] etc. We search for the given fragment in indexed music library. The digital music score library can contain sheet music of any kind and in any form, however it has to be preliminary indexed. As an example in this work we take a test library of MusicXML files and present a software tool that performs indexing automatically. As a search result we expect a list of music scores, containing the given fragment or similar to it. In the case the search query is not presented in our index, we propose to change it by a set of rewriting rules, that can help to correct automatically some possible misprinting. A special measure is introduced to estimate the similarity

[1]MuseScore sheet music library and editor https://musescore.com (accessed 2019, Dec), Petrucci Music Library, https://imslp.org (accessed 2019, Dec) , НотоМания, http://www.notomania.ru (accessed 2019, Dec)

[2]Shazam acoustic fingerprint music search system https://www.shazam.com/ru (accessed 2019, Dec)

[3]MuseScore sheet music library and editor https://musescore.com (accessed 2019, Dec)

[4]Finale, https://www.finalemusic.com (accessed 2019, Dec)

[5]Guitar Pro, https://www.guitar-pro.com (accessed 2019, Dec)

between the search query and a fragment from index. It takes into account the number of differences between fragments and the number of rules, that have been applied to remove differences between the search query and a fragment from index.

III. Internal representation

To solve a problem with different symbolic representation of transposed music fragments or played in different tempos we use normalized representation. Each note is presented as a pair of two numbers (relative pitch, relative duration), the same approach is used in [5], [6]. Relative pitch is computed as a distance in tones from the previous note in the melody. Relative duration for the notes are calculated as relation to the previous note. The particular problem of this representation that it is not valid for chords. We propose to select one note of chord. The relative duration for pauses are calculated as relation to the previous pause duration or to the previous note duration if there is no previous pause.

IV. Indexing of Sheet Music Collection

The indexing of music files is a challenging task, because it is difficult to formalize the process of splitting a composition into melodies for building a dictionary. The automatic discovering logically finished part of melody is a process that is difficult to automate. Usually the search query is much shorter than a full music piece that's why we put in our index short fragments corresponding to a period (8 bars) and its parts (4, 2 bars) taken from the score. The obtained fragments are translated into normalized representation and duplicates are not added to the index. For each of the fragments we have a list of compositions in which it occurs.

We propose to organize index as a database implemented on PosgreSQL [3]. That allows us to use the benefits of this database management system: to perform effective search, and also that give other potential users a possibility to add new items to index by writing SQL–queries either manually or by some external software. In this work the special software tool is developed for automatic indexing of MusicXML files. The program was written on Python and stored as an auxiliary module. However indexing can be performed for other formats manually or using external software and new content is added to PosgreSQL database – index by means of SQL-queries.

V. Music Retrieval Algorithm

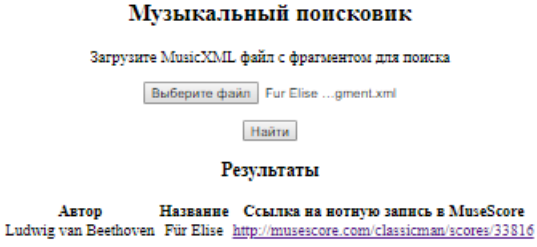
In the our sheet music retrieval system we have to perform fuzzy search. We use an approach based on rippling [5], and continue and extend the ideas proposed in [6], [12]. In the information retrieval we are faced with the situation than we have two similar (but not identic) fragments: a search query and a candidate to be an answer. We need to find if they represent the same composition. This task is quite similar to a mathematical one: while performing mathematical induction proof we have a hypothesis and a goal that are syntactically similar, and we need to rewrite the goal to prove that it is implied by the given. To solve the problem we can use rippling, previously proposed as an heuristic for automation of mathematical induction method [5]. The precondition for using the method is the following: we are rewriting one sentence to another and the sentences have to be similar. The differences are annotated as a wave front and the common parts form a skeleton. We do rewriting of the goal to reduce differences and the skeleton has to be preserved. To estimate differences reduction a measure is proposed. It has to become smaller with every rewriting step and it equals 0 when all the differences are removed. We extend the application of this approach from mathematical formulas to a new domain – a sheet music fragments. For this we have to introduce:

- a set of wave rules for musical domain;
- a measure to estimate differences between musical fragments;
- a proof that each of wave rules reduces the number of differences according to the proposed measure.

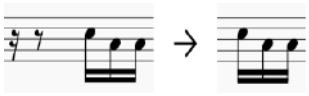
In this work we introduce a set of wave rules (extended version of [6], [12]. Differences are estimated as number of notes in wave front and each wave rule rewrites fragments reducing differences.

The work of our search engine consist of several steps:

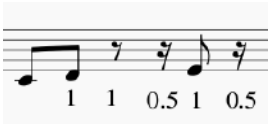
1) Get a search query in MusicXML format.



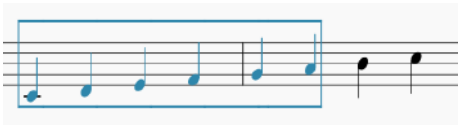
2) Translate a query into internal normalized representation
   a) Remove pauses in the begining of the query

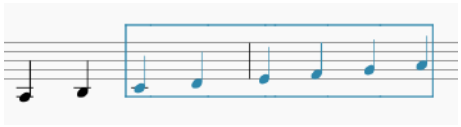


   b) Relative duration of other pauses is calculated based on previous note. The first note after pauses is divided on the length of the previous note.



3) Comparing the internal representation of the search query with fragments from the index. Since we are performing fuzzy search, for each pair (query and database fragments) we perform the following steps:
   a) Marking the longest common sequence (LCS) using the dynamic programming method [7]. If it is shorter than a half of the shortest fragments, not considering them similar. Otherwise mark all common notes as a skeleton and others as a wave-front.
   b) Deleting wave fronts in the beginning (prefix) and in the end (suffix) of sequence.





   The longest common subsequence is marked after deleting prefix and suffix
   c) Remove pauses after the final note in our fragment.
   d) Compare the fragments as they are build from the same note.
   e) Estimating the similarity using the formula

$$Similiriaty(F_1, F_2) = \frac{2 \cdot S}{L_1 + L_2 + \alpha \cdot N}, \quad (1)$$

   where F_1, F_2 - compared fragments, S - length of the longest common sequence, L_1 - the length of fragment F_1, L_2 - the length of F_2,
   α - heuristic coefficient, taken now as 0.5 after experiments with the system, N - number of applied rules.
   f) If the similarity is greater or equal 0.9, the fragments are considered to be similar and the resulted composition from the database should be mentioned in the list of search results.

Otherwise we are trying to remove differences using wave rules from [6] and the set of rules is extended by new ones. We use rules of three categories:

- Error correction rules.
  The aim of the rule is correction of possible errors made by user in the query, because it is written as he had heard it. For example an error in intervals like fifth, sixth and bigger user can make a mistake writing a note half tone higher or lower. Such differences in the border notes of the wave front can be considered as an error made by user in a query and they are proposed to be corrected by rewriting.
- Alternative notations.
  The same musical fragment can be written and performed in different ways, however the melody can be considered the same. This happens in variations and in different arrangements. The rules of this category assume different placement of pauses and elimination of non-chord notes.
- Elimination rules.
  After applying the previously mentioned rules the longest common sequence becomes longer. However the rippling method does not allow to change skeleton. That is why the newly constructed common parts are removed from the sequence. Only the notes that are at the border with the skeleton can be removed by this rule.

Since we have many rules in our database we have to avoid cycles in their applications. We propose to use a measure, that should become smaller with every rewriting step. We take a number of notes in the wave front as such a measure. The rewriting process stops in two situations; either the measure becomes equal 0, or there are no applicable rules. So at every step of rewriting we use only the rules that decreases measure of the sentence and we avoid infinite loops. An example of rule is presented here. A note (x,y), where x – pitch and y – duration and a pause (y) directly after it can be replaced by the same note with longer duration (x,2*y):

[...](x,y)(y)[...] ==> [...](x,2*y)[...]

After applying rules we calculate the similarity again and make a decision about correspondence of the fragment to the search query.

VI. Example of work

A short example of music retrieval performed by our prototype system is presented in the Table I. The search query is presented in line 9, among the library fragments we have 9 similar ones. Their differences are reduced by wave rules and the similarity is calculated using formula (1) and shown in the right column.

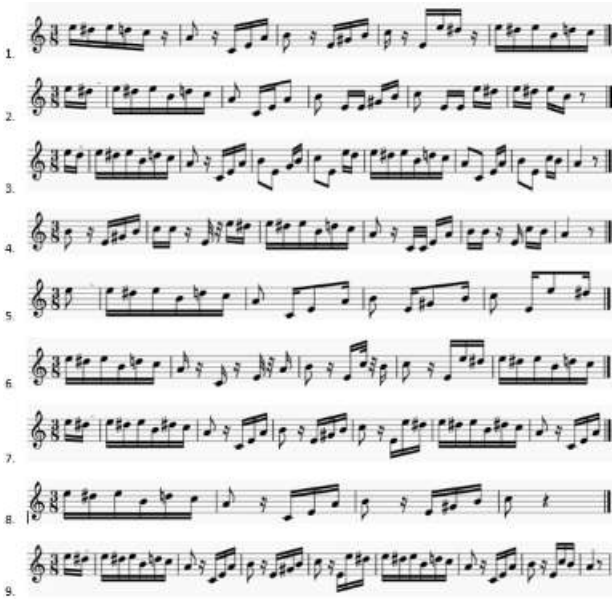


Table I
RETRIEVAL RESULTS WITH ESTIMATION OF SIMILARITY

Query	*Result*	*Similiriaty*
9	1	0.81
	2	0.75
	3	0.83
	4	0.83
	5	0.63
	6	0.97
	7	0.97
	8	1
	9	1

CONCLUSIONS

In the paper we propose an approach to music information retrieval, where the search query is a MusicXML sequence. Being considered as a structured documents retrieval, however it is a challenging task, because specific knowledge about the domain needed to be taken into account. We perform retrieval based on rippling and our index is stored as a PosgreSQL database, that is constructed automatically for MusicXML documents or manually for other formats. The work is still in progress: we analyze the possibility to improve retrieval efficiency, however the approach is already implemented in a prototype system, that allows to perform fuzzy search for sheet music.

REFERENCES

[1] Musipedia, http://www.musipedia.org (accessed 2019, Dec)
[2] MusicXML format, https://www.musicxml.com (accessed 2019, Dec)
[3] PostgreSQL, https://www.postgresql.org (accessed 2019, Dec)
[4] C. D. Manning, P. Raghavan and H. Schütze Introduction to Information Retrieval, Cambridge University Press. 2008.
[5] A. Bundy, D. Basin, D. Hutter, A. Ireland Rippling: Meta-level Guidance for Mathematical Reasoning. Cambridge University Press, 2005.
[6] Korukhova Y.S., Mytrova M.V. Sheet Music Retrieval / Open Systems. DBMS — 2013. — № 7. — p. 57–58, http://www.osp.ru/os/2013/07/13037358/ (accessed 2019, Dec)
[7] Dan Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge University Press. 1997.
[8] Viro V., Peachnote: Music Score Search and Analysis Platform/// Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011. pp. 359–362.
[9] Typke R. Music Retrieval Based on Melodic Similarity. Ph.D. Thesis, University of Utrecht, 2007
[10] Typke R., Giannopoulos P., Veltkamp R.C., Wiering F., van Oostrum R. Using Transportation Distances for Measuring Melodic Similarity // Proceedings of the Fourth International Conference on Music Information Retrieval. 2003. pp. 107–114.
[11] Yang X., Chen Q., Wang X. A Novel Approach Based on Fault Tolerance and Recursive Segmentation to Query by Humming // Advances in Computer Science and Information Technology: AST/UCMA/ISA/ACN. Conferences Joint Proceedings, Springer–Verlag Berlin, Heidelberg, 2010. pp. 544–557.
[12] Korukhova Y.S., Mytrova M.V. Information retrieval for music scores // Preprint of Keldysh Institute of Applied Mathematics — 2013. — № 48. — p. 1–16, https://keldysh.ru/papers/2013/prep2013_48.pdf (accessed 2019, Dec)

Об одном подходе к музыкальному информационному поиску нотных записей Корухова Ю.С., Толыбаева А.

В работе рассматривается задача поиска нотных записей музыкальных произведений по заданному в виде последовательности нот фрагменту мелодии. В связи с появлением больших электронных нотных библиотек, такая задача становится особенно актуальной.

Поисковый запрос формулируется в виде файла формата MusicXML, который поддерживается большинством программ – нотных редакторов наряду с их собственными форматами. Поиск ведется в электронной нотной библиотеке, которая должна быть предварительно проиндексирована. Индекс хранится в виде базы данных PostgreSQL. Для автоматического построения индекса по MusicXML файлам создан специальный программный инструмент. Файлы других форматов могут быть проиндексированы вручную либо с использованием дополнительных программ. Взаимодействие с индексом – базой данных возможно с помощью SQL – запросов.

При реализации поиска возникает задача сравнения фрагмента – запроса с элементом индекса, причем представляет интерес не только поиск точно совпадающих, но и похожих на запрос произведений. Реализация нечеткого поиска выполнена на основе волновых правил, традиционно применяемых в автоматическом доказательстве теорем методом математической индукции. При построении такого доказательства гипотеза и заключение индукции являются синтаксически похожими, и стоит задача устранения символьных различий между ними. Для ее решения применяются специальные правила переписывания, которые по построению уменьшают количество различий после их применения. В работе составлены волновые правила для устранения различий похожих музыкальных фрагментов и возможности корректировки неточности в поисковом запросе.

Предложенный в работе подход реализован в прототипной программной системе поиска нотных записей.

Received 15.12.2019