2017 International Conference on Tools with Artificial Intelligence

BIRITS: A Music Information Retrieval System Using Query-by-Playing Techniques

Lucas Martiniano, IEEE Graduate Student Member* and Carlos Silla, IEEE Member†
Graduate Program in Computer Science (PPGIa)
Pontifical Catholic University of Paraná (PUCPR)
Curitiba, PR, Brazil 80215–901
Email: *lucas.martiniano@gmail.com, †carlos.sillajr@gmail.com

***Abstract*—With the steady growth of the Internet many search websites have emerged, however most of these websites only allow you to perform the search using a textual interface. A more intuitive method for the user would be to use melody snippets of a song to perform this search. In this paper, we present a two-stage query-by-playing (QBP) system using symbolic representations where from a melody snippet of a song it is performed the retrieval of a set of similar songs. On the first stage we perform the alignments and calculate its result rank, which may have draws. Stage two was created to help solving this problem, where a classifier-based filter is applied on the results, reducing the number of draws. A new methodology for evaluating QBP is presented, aiming to understand the impact of the number of notes in the query and some alternatives to verify the robustness of the methods considering wrong and missing notes. In addition, a new method based on machine learning (ML) is presented to filter the results. In the experiments this method always improves the performance of queries regardless of the noise.**

I. Introduction

The constant growth of musical data on the Internet has encouraged several researchers to develop appropriate tools for the analysis and classification of these data [1]. The main purpose of those tools is to extract the information in a compact and representative way of the contents of databases [2]. Hence it is necessary to introduce new ways to search and retrieve those contents efficiently. Although there are several websites on the Internet that allow you to find songs, most of these sites only allow you to perform the search using a textual interface, i.e., the user must provide the data about the title, author or album of the song he is looking for. However this approach is not adequate, since many users are unaware of the title or origin of the desired songs [3].

This paper proposes an intelligent recovery system of music scores where a melody snippet of a song is used as a query to retrieve the full music score. The search is performed using local sequence alignment algorithms in a database composed of 10,199 Traditional Irish music scores in MIDI format. Where the result of this search is the most similar scores with the melody snippet used as query by the user. We found that the system proved effective in dealing with perfect and imperfect queries.

Figure 1 illustrates an example of a query by playing system where given an input several music scores are retrieved. It should be noted that each score has an associated score (given by an algorithm) and that sometimes draws can happen such as in the example.

Therefore, the main contributions of this work are:

- A new MUSic SUbstitution Matrix (MUSSUM), created to assess the quality of the musical notes alignment, assigning a score for match or mismatch occurrences.
- A machine learning based filter, that aims to reduce the number of draws by trying to learn the different music genres and using this information to filter out potential wrong music scores that have the same score but are from different music genres.
- To propose an evaluation framework that tries to answer the following questions: When using perfect queries, is there a minimum number of notes that must be played in a query by playing system in order to always retrieve the desired music score? Does it matter whether we use the beginning, middle or end parts of the music score to perform the query? How efficient is the music retrieval system regarding to the number of notes in an imperfect query in different parts of the song? Does the proposed ML-based filter approach improve the results?

In addition, the database, substitution matrix and alignment algorithm are available for reproduction[1].

The remainder of this paper is organized as follows: Section II presents the related work. Section III presents the proposed method. Section IV presents the experimental protocol used in this work. Section V presents the results obtained. Section VI presents the conclusions.

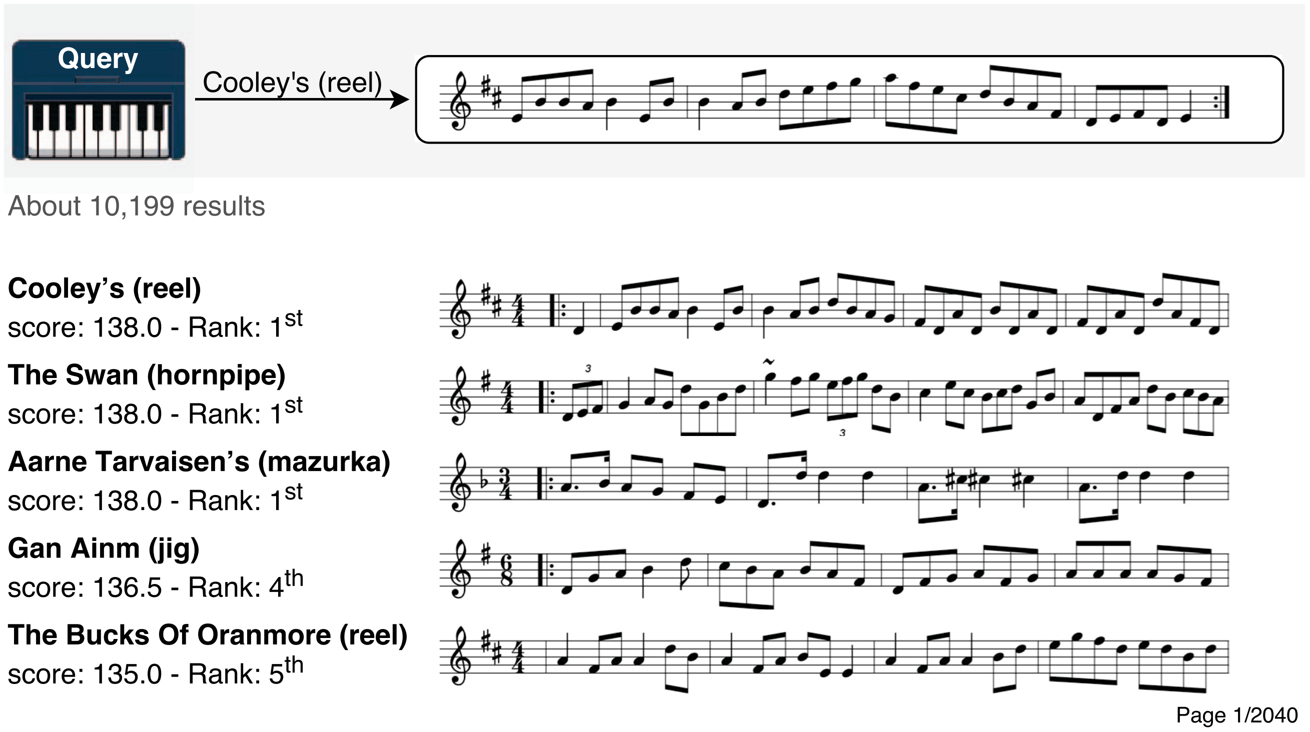
II. Related Work

A considerable number of papers have been published with a focus on music information retrieval (MIR) systems using different query inputs (i.e., Query-by-Playing, Query-By-Humming, Query-By-Example, Query-by-Tapping, etc.).

According to [4, 5], most methods that deal with MIR use the music symbolic data to perform the retrieval. Furthermore, even when using audio as input to retrieve songs, the authors usually transform the audio in symbolic data [6–9].

The authors of [4] present a survey with the state of the art works in symbolic melodic similarity, however only three out of fifteen use sequence alignment to retrieve the most similar

[1]https://github.com/martinianodl/birits



Cooley's (reel)
score: 138.0 - Rank: 1st

The Swan (hornpipe)
score: 138.0 - Rank: 1st

Aarne Tarvaisen's (mazurka)
score: 138.0 - Rank: 1st

Gan Ainm (jig)
score: 136.5 - Rank: 4th

The Bucks Of Oranmore (reel)
score: 135.0 - Rank: 5th

Fig. 1. Example of a query by playing system

songs. Besides this, to the best of our knowledge, none of them use a machine learning-based filter to increase the accuracy of the system.

We organized the symbolic similarity-finding algorithms in five categories based on: distance, machine learning, mathematics, sequence alignment and text. In works like [10–13], distance-based algorithms were used to calculate the similarity of the songs, for example, Manhattan distance, correlation distance, Euclidean distance and geometric distance.

In [10, 14–17], feature vectors are extracted from the songs and then fed to a ML algorithm such as k-Nearest Neighbor (kNN) [18] and Support Vector Machines (SVM) [19].

Some works use mathematical methods to calculate similarity such as Markov Models [7, 20], Minimum area between polygonal chains [21], Structure induction [22], n-grams [10, 11, 23–26], and Number of matching elements between melodies [27]. Only a few works use text-based information retrieval techniques to calculate similarity [8, 10, 28].

The most used method is the sequence alignment [6–11, 24, 29–35]. According to [36], the local sequence alignment algorithm outperformed most of the others pattern-matching approaches.

III. BIRITS

We propose an intelligent recovery system of music scores where a melody snippet of a song is used as a query to retrieve the full music score. Figure 2 presents the proposed method. It has a pre-processing stage (Section III-A) and two main stages: Alignment (Section III-B) and Filtering (Section III-C).

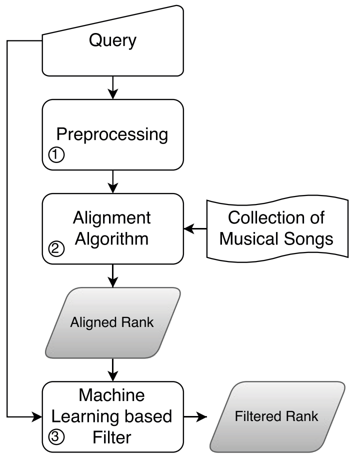


Fig. 2. Overview of BIRITS

A. Pre-processing

To be able to compare the sequences from musical scores, we developed a MusicXML/MIDI parser. The algorithm reads the mXML or the MIDI files and generates a list of tuples. After the creation of the list of tuples, we transposed all songs to the C major key.

B. Alignment

Since we aim to query different sections of the songs, we employ the Smith-Waterman alignment algorithm [37], which has been demonstrated to be a powerful tool for computing local sequence alignment.

1) Smith-Waterman Algorithm: The Smith-Waterman algorithm is a local sequence alignment algorithm developed by Temple F Smith and Michael S Waterman [37]. This algorithm is based on an older algorithm known as Needleman-Wunsch [38]. It uses dynamic programming techniques to find optimal alignment using sequences of any size. The algorithm uses a scoring matrix to measure the similarity between characters. It has the parameters: match, mismatch and gap penalty to fill the scoring matrix.

Assume two sequences X and Y, of sizes m and n, respectively. Determine the substitution matrix s (Section III-B2) and the penalty of a gap W_k that has length k, having the open v and extend u gap penalty[2] (used as default: $v = 15$ and $u = 0.5$), where $W_k = u(k-1) + v$.

F is a scoring matrix of size $m+1 \times n+1$. First, it is allocated $F(i,0) = 0 \ \forall i \in [0,n]$ and $F(0,j) = 0 \ \forall j \in [0,m]$, $n, m \in \mathcal{N}$.

To fill the rest of the matrix, we need to calculate the scores that comes from the upper left diagonal, the top and the left of each cell. To do so, we use Equation 1.

$$F(i,j) = max\begin{cases} 0 \\ F(i-1,j-1) + s(x_i,y_j) \\ F(i-k,j) - W_k \\ F(i,j-l) - W_l \end{cases} \quad \begin{pmatrix} 1 \le i \le n \\ 1 \le j \le m \end{pmatrix} \tag{1}$$

Where,

- 0 means there is no similarity up to x_i and y_j;
- $F(i-1,j-1) + s(x_i,y_j)$ means that x_i and y_j are associated;
- $F(i-k,j) - W_k$ means that x_i is at the end of a gap of length k;
- $F(i,j-l) - W_l$ means that y_j is at the end of a gap of length l;

After calculating all the scores and filling the F matrix, we choose the highest score on the matrix.

2) Substitution Matrix: In order to have a more precise alignment, we propose a novel substitution matrix called MUSSUM (MUSic SUbstitution Matrix). The values for the mismatch were calculated according to the distance between each semitone of the C major scale. The main diagonal was calculated considering the frequency of each note in the database. The notes that obtained the highest frequency were scored with lower values than ones with small frequencies. This was done because the match between notes that have a low frequency in the database presents a greater chance of being the desired sequence. The representation of the matrix is shown on Figure 3.

[2]Gap opening: cost to create a gap
Gap extension: cost to make a gap bigger

	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
C	8	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
C#	-1	8	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
D	-2	-1	4	-1	-2	-3	-4	-5	-6	-7	-8	-9
D#	-3	-2	-1	10	-1	-2	-3	-4	-5	-6	-7	-8
E	-4	-3	-2	-1	5	-1	-2	-3	-4	-5	-6	-7
F	-5	-4	-3	-2	-1	5	-1	-2	-3	-4	-5	-6
F#	-6	-5	-4	-3	-2	-1	6	-1	-2	-3	-4	-5
G	-7	-6	-5	-4	-3	-2	-1	5	-1	-2	-3	-4
G#	-8	-7	-6	-5	-4	-3	-2	-1	10	-1	-2	-3
A	-9	-8	-7	-6	-5	-4	-3	-2	-1	4	-1	-2
A#	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	10	-1
B	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	5

Fig. 3. Music Substitution Matrix (MUSSUM)

C. Machine Learning based Filter

It is possible for several songs to have the same match score sharing the same position in the rank, i.e. a draw. For that reason, we created a ML-based filter to reduce the number of draws in a query, consequently having better results.

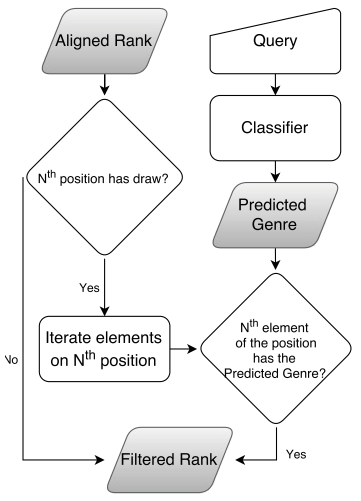

Aligned Rank
Query
Classifier
N^th position has draw?
Predicted Genre
Yes
Iterate elements on N^th position
N^th element of the position has the Predicted Genre?
No
Filtered Rank
Yes

Fig. 4. Overview of the filter

1) Classifier: A total of 1,225 features were extracted from the MIDI database using the jSymbolic software [39]. We have employed the Support Vector Machine (SVM) classifier for the task of music genre classification. SVM is a supervised machine learning algorithm that uses hyper-planes to separate different classes of data.

2) Filtering the Rank: Figure 4 shows an overview of the proposed filter. In order to perform the filtering, we use the query as our classifier input. The classifier predicts the genre of the query. We then compare each element of each position of the rank and check whether or not the element has the same genre as the query. If it does, we add it to the Filtered Rank if not we just discard it. In some cases no element of the position will contain the predicted genre. In that case we keep the original elements on that position.

Some preliminary tests shows that the average accuracy of the classifier is 82.48%.

IV. Experimental Design

Our experiments are based on the Irish Traditional Music database presented in [40]. This database contains 10,199 traditional Irish songs in MIDI format extracted from the thesession.org website annotated in 11 different genres (Barndance, Hornpipe, Jig, Mazurka, Polka, Reel, Slide, Slip Jig, Strathspey, Three Two, Waltz).

From this database, we created a corpus composed of 1,376,865 queries, along with the Irish Traditional Music database as our ground-truth. The corpus simulates real life scenarios where users of the query by playing system will probably make some mistakes when performing the query. For this reason, we created three types of query modification: adding extra notes, removing some notes and replacing some notes. Furthermore, in all queries we used different positions of the music score, namely the beginning, middle and end parts.

Regarding the algorithms, we have adapted the Smith-Waterman implementation from [41] and used the SVM implementation from the WEKA tool [42]. All experiments were executed using a stratified 10-fold cross-validation technique. Care has been taken to use the exact same cross validation folds in all experiments.

A. Evaluation Metrics

Since we are going to have only one relevant item on our result rank we should evaluate the quality of our music retrieval system using the Mean Reciprocal Rank (MRR) shown in Equation 2.

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{r_i} \tag{2}$$

where r_i refers to the position in the rank of the correct song for the i-th query.

However, in some cases, it is possible for several songs to have the same score, consequently sharing the same position in the rank. In order to evaluate the quality of our system considering the possible draws, we introduce the Mean draw Reciprocal Rank at n (MdRR@n), given by:

$$MdRR@n = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{d_i \times r_i} \tag{3}$$

where r_i and d_i refers to the position and the number of draws in the rank of the correct song for the i-th query, respectively.

We also evaluate the MdRR at different precision levels (@1, @5, @10, and @25). According to Van Deursen and Van Dijk [43] 91% of searchers do not go past the first page of the search results. For that reason we evaluate our results until 25th position in the rank, i.e. if the user sees ten results at a time, in the worst-case scenario the desired result will be in the third page.

TABLE I
MdRR@1 for perfect queries using 10, 15, 20, 30, 50 and 100 notes

Num. of Notes	Part	Aligned	Filtered
10	Beginning	0.8802	0.9143
	Middle	0.9188	0.9441
	End	0.8778	0.9147
15	Beginning	0.9915	0.9925
	Middle	0.9952	0.9957
	End	0.9938	0.9948
20	Beginning	0.9960	0.9963
	Middle	0.9976	0.9978
	End	0.9974	0.9977
30	Beginning	0.9981	0.9982
	Middle	0.9988	0.9989
	End	0.9984	0.9985
50	Beginning	0.9987	0.9988
	Middle	0.9992	0.9993
	End	0.9989	0.9990
100	Beginning	0.9993	0.9994
	Middle	0.9993	0.9994
	End	0.9993	0.9994

V. Results

In this section we are interested in answering the following questions by using controlled experiments: When using perfect queries, is there a minimum number of notes that must be played in a query by playing system in order to always retrieve the desired music score? Does it matter whether we use the beginning, middle or end parts of the music score to perform the query? i.e. to which extent does the location of the query impact the retrieval results (Section V-A)? How efficient is the music retrieval system regarding to the number of notes in an imperfect query in different parts of the song (Section V-B)? Does the proposed ML-based filter approach improve the results (Section V-C)?

A. Is there a minimum number of notes in order to always retrieve the desired music score?

Table I presents the MdRR for a query generated from a particular position of the original music score with 10, 15, 20, 30, 50 and 100 notes. Since we aim to always retrieve the desired music score, i.e. to have it on the first position of the rank, we use the $MdRR@1$ where we only consider the first position of the rank. By analyzing Table I we can assume that any perfect query with more than 15 notes will have a MdRR@1 of approximately 1. Therefore, we can consider that the minimum number of notes in order to always retrieve the desired music score is 15.

1) Does the played part matters?: A Friedman test was conducted in order to evaluate if there is a statistically significant difference in the beginning, middle and end locations of a perfect query. There was a statistically significant difference between the part of the songs, $\chi^2(2) = 28.1739$,

$p = 7.6228\text{e}{-7}$. Post hoc analysis with Wilcoxon signed-rank tests was conducted with a Bonferroni correction applied, resulting in a significance level set at $p < 0.017$. There were no significant differences between the beginning and end locations ($Z = -2.2022$, $p = 0.0276$). However, there was a statistically significant difference in the beginning vs. middle locations ($Z = -3.9254$, $p = 0.000087$) and middle vs. end locations ($Z = -4.0053$, $p = 0.000062$).

B. How efficient is the music retrieval system for imperfect queries? Does the played part matters?

Figures 5 to 8 and Figures 13 to 16, present the performance of our music information retrieval system according to the number of notes added and substituted in the queries, respectively.

We can notice that for all cases adding or substituting up to five notes the system will retrieve the desired music score within the top 5 results. However, analyzing Figures 9 to 12, which present the performance of our system according to the number of notes deleted in the queries, in order to retrieve the desired music score within the top 5 results, our system has a tolerance of up to four missing notes in a query.

In order to evaluate if the part of the music score with modifications (i.e. imperfect) used as a query matters, we have performed the Friedman test for each note played in the beginning, middle and end locations. There was a statistically significant difference between the part of the songs, $\chi^2(2) = 16.5028$, $p = 0.0003$. Post hoc analysis with Wilcoxon signed-rank tests was conducted with a Bonferroni correction applied, resulting in a significance level set at $p < 0.017$. There were no significant differences between the beginning and end locations ($Z = -2.1164$, $p = 0.0343$). However, there was a statistically significant difference in the beginning vs. middle locations ($Z = -3.1887$, $p = 0.0014$) and middle vs. end locations ($Z = -3.9329$, $p = 0.0001$).

It happens because typically Irish songs are constructed of two parts (‘A’ and ‘B’). Each part is usually eight bars in length and each part is played twice, i.e. AABB. In the beginning of the song there are usually extra notes before starting the ‘A’ part, consequently adding noise to the query. The end of the song consists of the last notes of the ‘B’ part, which are not as relevant as the beginning of ‘B’ part. The beginning of the ‘B’ part will most of the time be the middle part of the song, which is usually the chorus. The chorus is the main part of a song. In most songs the chorus reflects what the title of the song means.

C. Does the proposed ML-based filter approach improves the results?

Figure 17 shows the average gains, which is the average of the gains for each part of the song ($avg_{gain} = 1/3 * gain_{beginning} + gain_{middle} + gain_{end}$), for adding, deleting and substituting notes in a query. It is important to notice that for all cases between 3 and 7 modified notes the ML-based filter approach improved the results in more than 1%. Considering that the Irish database has 10,199 songs, after

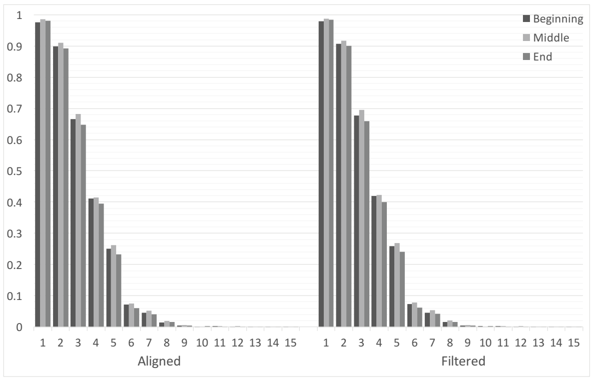


Fig. 5. MdRR@1 for Adding notes on the queries

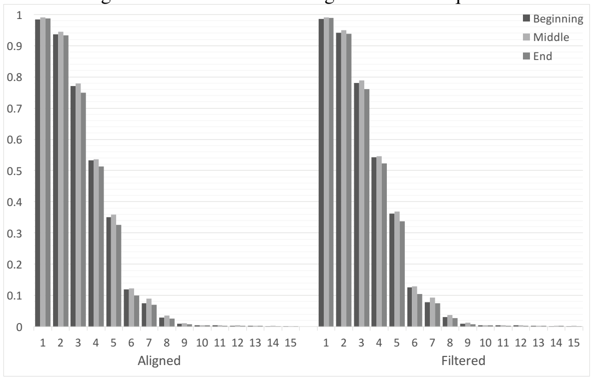


Fig. 6. MdRR@5 for Adding notes on the queries

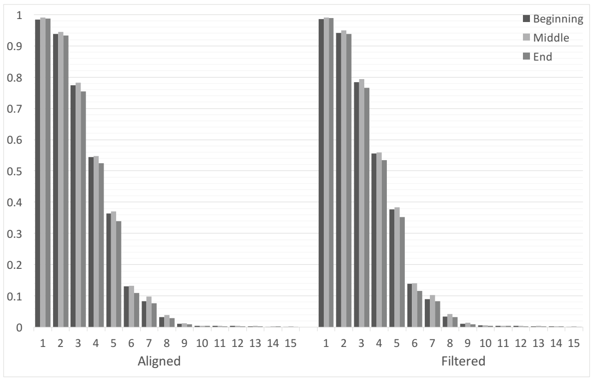


Fig. 7. MdRR@10 for Adding notes on the queries

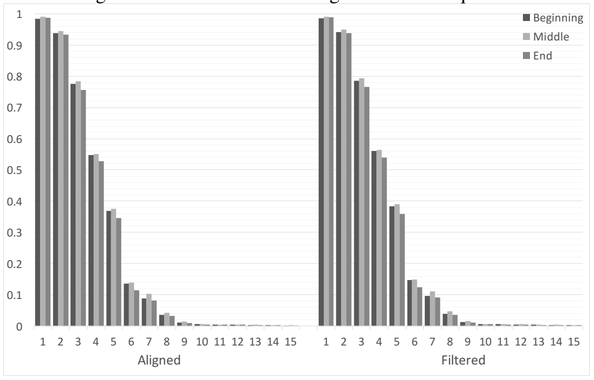


Fig. 8. MdRR@25 for Adding notes on the queries

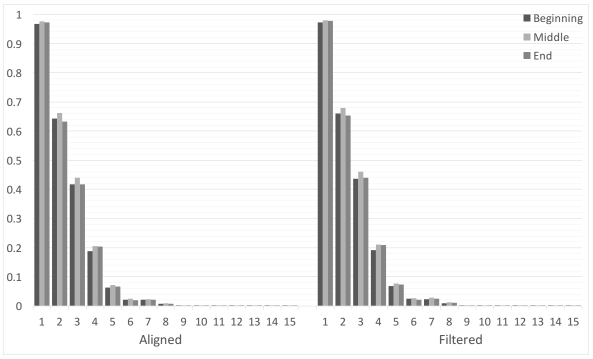


Fig. 9. MdRR@1 for Deleting notes on the queries

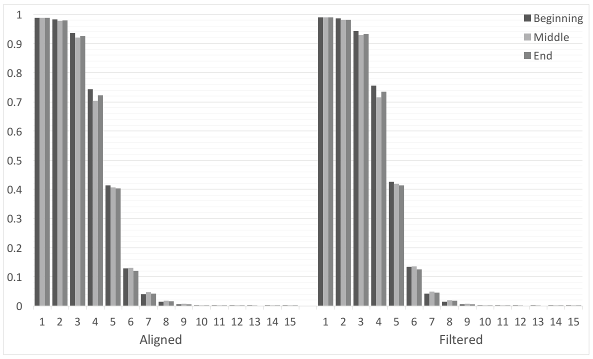


Fig. 13. MdRR@1 for Substituting notes on the queries

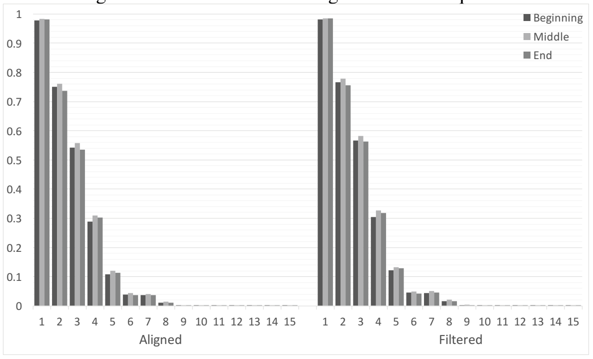


Fig. 10. MdRR@5 for Deleting notes on the queries

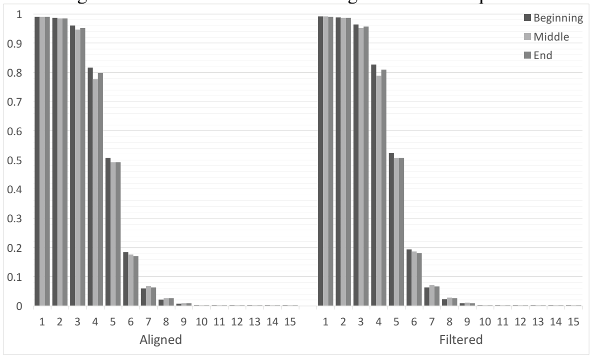


Fig. 14. MdRR@5 for Substituting notes on the queries

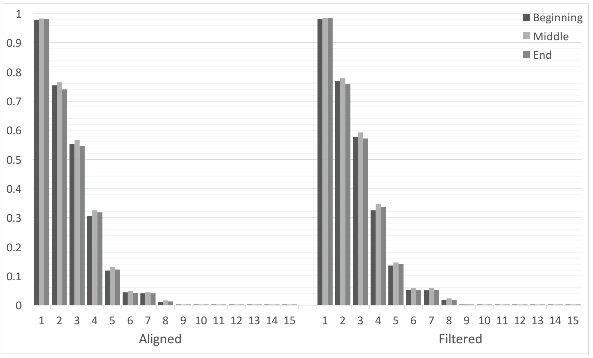


Fig. 11. MdRR@10 for Deleting notes on the queries

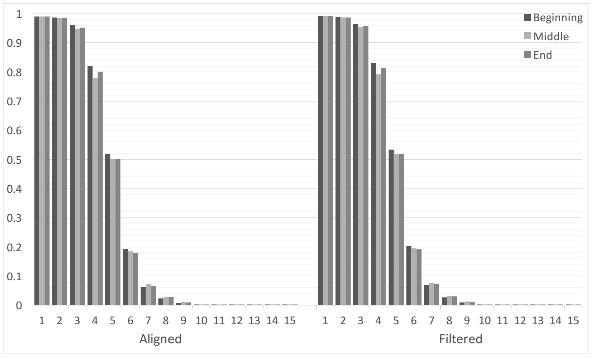


Fig. 15. MdRR@10 for Substituting notes on the queries

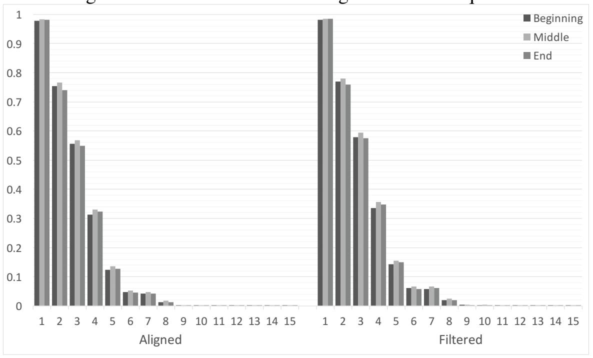


Fig. 12. MdRR@25 for Deleting notes on the queries

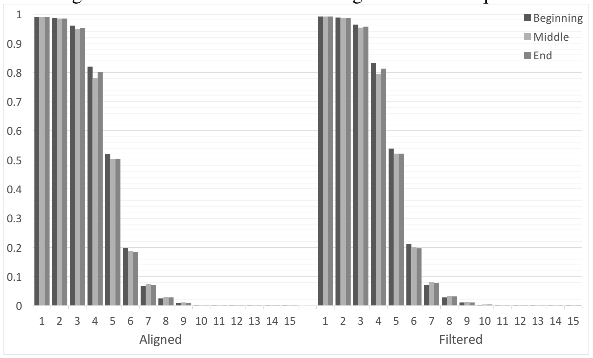


Fig. 16. MdRR@25 for Substituting notes on the queries

applying the filter it will reduce the result rank in at least 102 songs.

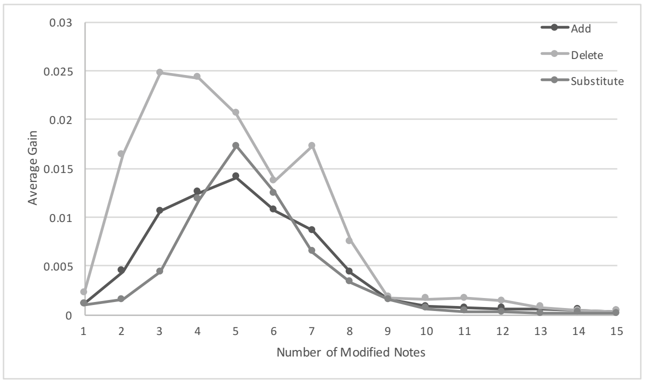


Fig. 17. Average gain for the modified notes in the queries

Also a Wilcoxon signed-rank test showed that the use of the ML-based filter approach is statistically significantly higher than using only the alignment algorithm ($Z = -10.0808$, $p = 6.7138\text{e}{-24}$).

VI. Conclusion and Future Work

In this work we proposed an intelligent music score retrieval system where a melody snippet of a song is used as a query to retrieve the full music score. The search is performed using local sequence alignment algorithm, with a new cost matrix, called MUSSUM. The experiments were performed using a database of 10,199 Traditional Irish music scores MIDI format. Experimental results on the database demonstrate that the proposed approach achieves greater results in retrieving songs from snippets of a song even with mistaken notes played along the query.

Acknowledgment

We would like to thank CAPES (Coordination for the Improvement of Higher Level Personnel) and CNPq (National Council for Scientific and Technological Development) from the Brazilian government and Fundação Araucária for their financial support on the project.

References

[1] C. Yang, "Music database retrieval based on spectral similarity," Stanford, Tech. Rep., 2001.

[2] H. H. Hoos, K. Renz, and M. Görg, "Guido/mir-an experimental musical information retrieval system based on guido music notation.," in *ISMIR*, 2001, pp. 41–50.

[3] S. Driscoll, "A trio of internet stars: Abc's," *Fiddler Magazine*, vol. 11, no. 2, 2004.

[4] V. Velardo, M. Vallati, and S. Jan, "Symbolic melodic similarity: State of the art and future challenges," *Computer Music Journal*, vol. 40, no. 2, pp. 70–83, 2016.

[5] A. Kotsifakos, P. Papapetrou, J. Hollmén, D. Gunopulos, and V. Athitsos, "A survey of query-by-humming similarity methods," in *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*, ACM, 2012, p. 5.

[6] B. Duggan and B. O'Shea, "Tunepal: Searching a digital library of traditional music scores," *OCLC Systems & Services: International digital library perspectives*, vol. 27, no. 4, pp. 284–297, 2011.

[7] E. Batlle, H. Neuschmied, P. Uray, and G. Ackermann, "Recognition and analysis of audio for copyright protection: The raa project," *Journal of the Association for Information Science and Technology*, vol. 55, no. 12, pp. 1084–1091, 2004.

[8] D. Bainbridge, "Music information retrieval research and its context at the university of waikato," *Journal of the Association for Information Science and Technology*, vol. 55, no. 12, pp. 1092–1099, 2004.

[9] M. J. Dovey, "Overview of the omras project: Online music retrieval and searching," *Journal of the Association for Information Science and Technology*, vol. 55, no. 12, pp. 1100–1107, 2004.

[10] R. Hillewaere, B. Manderick, and D. Conklin, "String methods for folk tune genre classification," in *ISMIR*, vol. 2012, 2012, 13th.

[11] D. Müllensiefen and K. Frieler, "Optimizing measures of melodic similarity for the exploration of a large folk song database.," in *ISMIR*, 2004.

[12] N. Orio and A. Rodà, "A measure of melodic similarity based on a graph representation of the music structure," in *ISMIR*, 2009, pp. 543–548.

[13] D. Rizo and J. M. Inesta, "Trees and combined methods for monophonic music similarity evaluation," *Proceedings of the Annual Music Information Retrieval Evaluation exchange*, 2010.

[14] C. Bohak and M. Marolt, "Calculating similarity of folk song variants with melody-based features.," in *ISMIR*, 2009, pp. 597–602.

[15] A. D. C. Junior and L Batista, "Sms identification using ppm, psychophysiological concepts, and melodic and rhythmic elements," *Music Information Retrieval Evaluation eXchange*, 2012.

[16] C. Roig, L. J. Tardón, A. M. Barbancho, and I. Barbancho, "Submission to mirex 2013 symbolic melodic similarity," *Proceedings of the Annual Music Information Retrieval Evaluation Exchange*, 2013.

[17] N. N. Vempala and F. A. Russo, "An empirically derived measure of melodic similarity," *Journal of New Music Research*, vol. 44, no. 4, pp. 391–404, 2015.

[18] E. Fix and J. L. Hodges Jr, "Discriminatory analysis-nonparametric discrimination: Consistency properties," California Univ Berkeley, Tech. Rep., 1951.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] K. Kim, D. Lee, T.-B. Yoon, and J.-H. Lee, "A music recommendation system based on personal preference

analysis," in *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*, IEEE, 2008, pp. 102–106.

[21] G. Aloupis, T. Fevens, S. Langerman, T. Matsui, A. Mesa, Y. Nuñez, D. Rappaport, and G. Toussaint, "Algorithms for computing geometric measures of melodic similarity," *Computer Music Journal*, vol. 30, no. 3, pp. 67–76, 2006.

[22] D. Meredith, "Point-set algorithms for pattern discovery and pattern matching in music," in *Content-Based Retrieval*, T. Crawford and R. C. Veltkamp, Eds., ser. Dagstuhl Seminar Proceedings, Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2006/652.

[23] K. Frieler, "Generalized n-gram measures for melodic similarity," *Data Science and Classification*, pp. 289–298, 2006.

[24] I. S. Suyoto and A. L. Uitdenbogerd, "Simple orthogonal pitch with ioi symbolic music matching," *Proceedings of the Annual Music Information Retrieval Evaluation exchange*, 2010.

[25] V. Viro, "Peachnote: Music score search and analysis platform.," in *ISMIR*, 2011, pp. 359–362.

[26] S. Yazawa, Y. Hasegawa, K. Kanamori, and M. Hamanaka, "Melodic similarity based on extension implication-realization model," *MIREX Symbolic Melodic Similarity Results*, p. 83, 2013.

[27] M. Laitinen and K. Lemström, "Geometric algorithms for melodic similarity," *Proceedings of the Annual Music Information Retrieval Evaluation exchange*, 2010.

[28] J. Wolkowicz and V. Kešelj, "Text information retrieval approach to music information retrieval," *Music Information Retrieval Evaluation eXchange*, 2011.

[29] J Urbano, "A geometric model supported with hybrid sequence alignment," *Proceedings of the Annual Music Information Retrieval Evaluation Exchange*, p. 82, 2013.

[30] C. Gómez, S. Abad-Mota, and E. Ruckhaus, "An analysis of the mongeau-sankoff algorithm for music information retrieval.," in *ISMIR*, 2007, pp. 109–110.

[31] P. Ferraro, P. Hanna, J. Allali, M. Robine, *et al.*, "Mirex query-by-humming/singing," in *Annual Music Information Retrieval Evaluation exchange (MIREX) as part of the 7th International Conference on Music Information Retrieval (ISMIR), Victoria, Canada*, 2006.

[32] K Frieler and D Müllensiefen, "Mirex: Three algorithms for symbolic similarity computation," *Proceedings of the Annual Music Information Retrieval Evaluation exchange (MIREX)*, 2006.

[33] M. Grachten, J.-L. Arcos, and R. L. De Mántaras, "Melodic similarity: Looking for a good abstraction level," *representations*, vol. 2, p. 7, 2004.

[34] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham, "Towards the digital music library: Tune retrieval from acoustic input," in *Proceedings of the first ACM international conference on Digital libraries*, ACM, 1996, pp. 11–18.

[35] M. Mongeau and D. Sankoff, "Comparison of musical sequences," *Computers and the Humanities*, vol. 24, no. 3, pp. 161–175, 1990.

[36] B. Janssen, P. van Kranenburg, and A. Volk, "Finding occurrences of melodic segments in folk songs employing symbolic similarity measures," *Journal of New Music Research*, vol. 46, no. 2, pp. 118–134, 2017.

[37] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[38] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[39] C. McKay and I. Fujinaga, "Jsymbolic: A feature extractor for midi files.," in *ICMC*, 2006.

[40] M. L. Martins and C. N. Silla Jr, "Irish traditional ethnomusicology analysis using decision trees and high level symbolic features," in *Proceedings of the Sound and Music Computing Conference (SMC 2015)*, 2015, pp. 455–462.

[41] A. Moustafa, *Jaligner: Open source java implementation of smith-waterman*, 2006.

[42] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[43] A. J. Van Deursen and J. A. Van Dijk, "Using the internet: Skill related problems in users online behavior," *Interacting with computers*, vol. 21, no. 5, pp. 393–402, 2009.