The DMUN System at the MediaEval 2017 C@merata Task

Andreas Katsiavalos
De Montfort University, Leicester, UK
andreas.katsiavalos@gmail.com

ABSTRACT

This paper presents a system that was developed for the C@merata task to perform music information retrieval using text-based queries. The system is built on findings from previous attempts and achieved best results and functionality so far. The C@merata task is split in two modules that handle the query-parsing and music-information retrieval separately. The sub-tasks are connected with a formal-information-request, a dictionary that contains the parsing information. The system is not fully extended but key issues and methods are identified.

1 INTRODUCTION

The C@merata task [1] represents a challenging task that aims to bind text and music content-based retrieval. The challenges of the task are important mainly because of the multiplicity of contexts within which the content that is searched needs to be defined. The variance in score formats, e.g. orchestral-scores in contrast to piano or single staff scores, the ambiguity in musical-concept descriptions and their exact positioning on the score, and, the technicalities of transferring the results of text-parsing to music-retrieval are some of the problems that need to be solved.

The C@merata task is important because it is addressing a fundamental need in music research, that of a simplified content-based music information retrieval system. Content-based retrieval systems are implemented in fields such as music informatics, with highly specialized applications, and, in general text- and multimedia-based systems in web-search engines. However, there are no user-friendly applications to perform what the C@merata task is challenging. Thus, the development of text-based query systems for music-information retrieval will fill the gap between specialized and non-content-based retrieval services for music.

A service that will satisfy the needs of the C@merata task would be helpful to everyone related with music and especially in higher-level music education where research often requires the identification of diverse and complex musical elements in large corpora. The textual-interface that is suggested from the task is also very practical for novice music enthusiasts that begin to discover the theoretic establishment of tonal music.

2 RELATED WORK

This paper draws from works in previous C@merata events and studies in music information retrieval generally. The clear distinction of query parsing and music-information retrieval between the C@merata sub-tasks enabled independent developments for each system. In 2015 [3], the focus was on the development of highly-parameterized music-information retrieval functions for high-level musical concepts, such as arpeggios and scales, while the system's text parsing was relying on Collins' Stravinsqi algorithm. The following year [2], the focus shifted to language processing for the development of an automated query parser. The results where promising and key tasks where identified and addressed, however, the connection between the query-parser and the music-information retrieval functions was very poor.

3 APPROACH

3.1 Overview

The system presented in this paper is a prototype method to connect text parsing and music information retrieval. The C@merata task is handled in two main stages: a) the text parsing, and, b) the music information retrieval. A shell was developed that integrates and connects the above elements, also handling I/O operations. The two stages are operating independently and are connected by the use of a data structure, named Formal Information Request (see below).

Each stage uses custom code that is not dependent on any high-level external libraries for either language processing or music information processing. Concerning language processing, the system is not able to handle completely 'natural' language but rather a collection of word constructs where each valid sentence is viewed as a structure of valid terms, types and type combinations. In this prototype system, only selected constructs were implemented for proof of concept; however, the language is easily extensible. While text parsing is carried out completely from scratch, the reading of musicXML files and some dictionary-related operations were facilitated by music21.

Two important notions of the system are the Formal Information Request (FIR) and the notion of (musical) 'durational element'. The FIR is a method to connect the output of the query parsing with the music-information retrieval functions. It basically transfers all the parsing data to a music function selector that further processes the parsing elements to be inputted to the music information retrieval functions. The notion of the durational element is very helpful in chaining input and output between music information retrieval functions.

Overall, as displayed in Figure 1, the system inputs a text query and initializes a query parser object by loading a .json language file, a dictionary with single term types for keys and sets of terms for values. The query-parser converts the text of the query into a Formal Information Request (FIR), another

MediaEval'17, September 2017, Dublin, Ireland

dictionary, by gradually identifying and replacing the terms, term types and compound types of the query with their types found in the language file, until a top-level description of the query is found. The FIR is then sent to the music information retrieval (MIR) module which in turn selects the corresponding information request retrieval function. All the currently possible information requests are implemented as combinations between three core types of MIR functions that find, relate and constrain music-entities such as notes/rests and note-sets (melodies, chords, etc.). Lastly, the output of the MIR functions, which are music elements, are converted into passages.

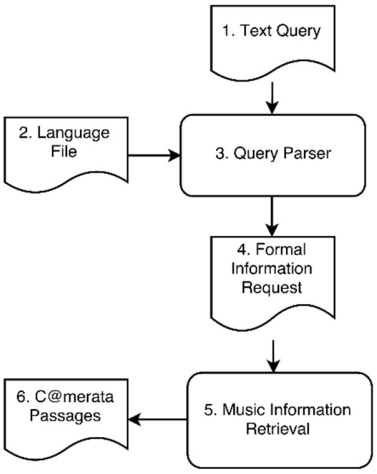

Figure 1: The overall workflow diagram.

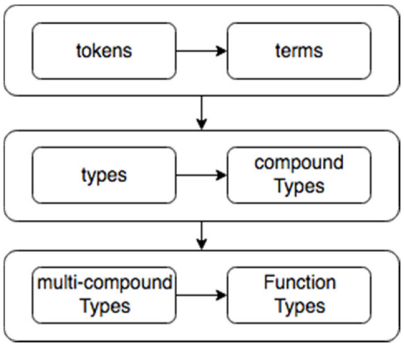

Figure 2: The text query parsing steps.

As shown in Figure 2 from top to bottom, the query parsing process starts with breaking down the query phrase into word tokens (terms) while commas (',') are removed. Next, the TYPE of each TERM is identified based on the language TERMS set. Next, compound types (cTYPE) are identified by searching for the maximal subset of adjacent parsed TYPES. Next, the query is parsed again to check if there are any multi-compound-types (mcTYPE). At this point, the query is viewed as a high-level pattern of musical-entities, relations and qualifications. These patterns cannot integrate more and since their content, context, and requirements are identified, they are viewed as high-level functions.

3.2 Parsing of text queries

The query parsing module inputs the query phrase and after a sequence of parsing operations it outputs the FIR . The parsing is based on a 'language' file that holds all the information that is required to identify the type of the query. Parts of the language file are generated algorithmically.

Table 1: Example parsing of query number 58

query	chord C# E G# in the bass clef
terms	'chord', 'C#', 'E', 'G#', 'in', 'the', 'bass', 'clef'
types	'primaryType', 'pitch', 'pitch', 'pitch', 'contextRel', 'contextRel', 'partId', 'primaryType'
cTypes	[0,3, 'chord'], [4,5, 'contextRel'], [6,7, 'partContext']
mcTypes	[0,3, 'chord'], [4,7, 'partQualification]
function	getEntityInContext()

Since all the questions where converted into combinations of Entities(E), Relations(R), and. Qualifications(Q), the set of valid combinations can be given from the graph shown in Figure 3, starting with an entity (E). Following this graph in text parsing was revealing in what kind of patterns are used and what kind of functions need to be developed.

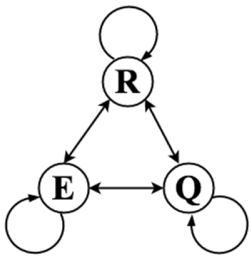

Figure 3. Starting with an Entity (E), a query can have any combination of paths in this cyclic graph, however, not all of them are implemented.

Currently some of the functions that are implemented are (using the abbreviations from Figure 3): E, E-E, E-En , E-R, E-Q, E-Q-Q, E-R-E-Q and E-Q-R-E-Q.

3.3 Music information retrieval

The music information retrieval module starts with the formal information request of the query parser and outputs the music elements that satisfy the query question. In general the reverse

process of text parsing is followed: while in query parsing the language dictionary was used to find integrations of terms in order to identify the top query description, once the function is identified, the descriptions are broken down into elements but this time removing and combining terms to read values and perform music content searching.

The music information retrieval operations are handled by a simple script that was developed for this reason. The system operates with 'datapoint' lists, where notes and rests are the atoms. The music entities that are identified in the text parser as (E)ntities are shown in Figure 4; the MIR functions can currently retrieve the elements from the top three rows. Note that all the combinations between them are possible.

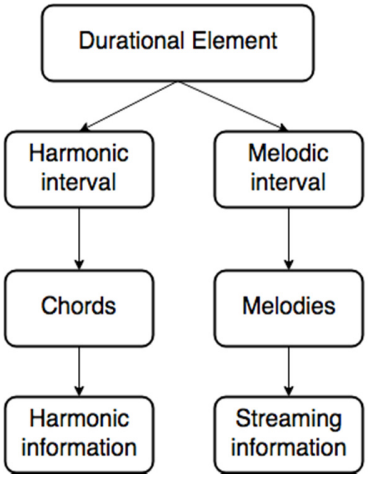

Figure 4: The musical entities.

There are generally two extremes in declaring and identifying Entities in queries and each one has different approach in retrieval. An entity may contain the specific constituents of the element, from highly specific e.g. a query 'C4 E4 G4 chord', to more abstract e.g. 'major chord'.

Table 2: The MIR functions

getEntity	Note, rest, harmonic/ melodic interval, chord, melody
getEntityAfterEntity	Only the 'followed by'
getEntityInContext	'Part' and 'measure' qualification

The Entities in Figure 4 are durational entities, meaning that they all have similar attributes such as a starting point and an ending point in time. The system makes use of these generic properties with robust MIR functions that can handle and mix any of them. For example a query 'G4 followed by minor' is served by an MIR function that handles 'Entity-After-Entity' and not 'Chord-After-Note'. This is an interesting feature with only partial exploitation.

4 RESULTS AND ANALYSIS

The system found great difficulties with text parsing and for that reason two groups of answers were made:

1. 'auto', where the queries were inputted 'as is' from the C@merata questions file without any alterations.
2. 'altered', where some parts of the query had to be altered to match the parsing capabilities.

Table 3: The 'auto' and 'altered' query groups

Type	Question numbers
Auto (7)	4, 58, 60, 63, 64, 92, 132
Altered (23)	1, 2, 3, 7, 11, 12, 18, 19, 23, 27, 33, 36, 39, 40, 42, 43, 52, 53, 61, 62, 70, 103,189

The main reasons to alter the original queries were:

- The 'bar' qualification is not implemented yet and the results had to be manually checked for that range. (e.g. 1, 11, 12, 13, 18, 19, 23, 42).
- The 'left' and 'right' 'hand' qualifications are also not implemented and these queries are altered to use part names instead (e.g. 11, 12, 13, 18, 19, 23, 36, 40, 43, 52)
- All the terms where altered to match a single language (e.g. 2, 7, 11, 18, 27, 33, 36, 39. 40, 62). For example query 27 'D D D C# C# C# B E E D D D in crotchets' is altered to 'D D D C# C# C# B E E D D D in quarters'.
- When not all the information given is used (e.g. 3, 39, 42, 53, 61, 70). For example in query 70 'theme' is considered a 'melody'.

Due to the small number of 'auto' answers and also to the fact that the alterations that had to be made are considered trivial, the results for the two groups were summed. The alterations are considered trivial because the methods to parse the original queries is known but not implemented. Also, all the answered questions were manually selected so that the MIR functions would be able to run them. This explains the overall low recall and high precision of the results shown in Figure 5 meaning that when the FIR was produced then the MIR was usually successful.

In general, as shown in Figure 5, the overall Beat Recall and Measure Recall did not exceed 0.2 percent (0.155 and 0.172 respectively), and from the total of 200 questions only 30 were answered. The generally high precision (0.833 for beat and 0.924 for measure) is, as stated earlier, due to the manual selection of queries into feasible and not feasible, and to minor alterations to their text. More specifically, the 'synch' category was completely excluded and very few 'follow' and 'texture' queries where tested. Most of the emphasis was given to the 'melodic' and 'harmonic' queries trying to answer as many as possible, but still with low recall in both.

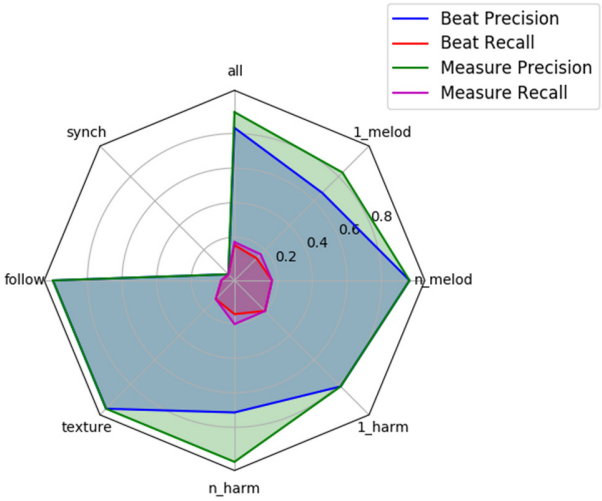

Figure 5: The results of the system.

5 CONCLUSIONS

The current system presents a working paradigm for the complete C@merata task, however as a prototype, it doesn't reach its potential. Although multi language support was not tested, this can be easily achieved by using a different language file. This way, apart from the differences in terms, different grammar constructs can also be used as the language file is fully customizable allowing the user to add their own grammatical constructs.

REFERENCES

[1] Sutcliffe, R. F. E., Ó Maidín, D. S., Hovy, E. (2017). The C@merata task at MediaEval 2017: Natural Language Queries about Music, their JSON Representations, and Matching Passages in MusicXML Scores. Proceedings of the MediaEval 2017 Workshop, Trinity College Dublin, Ireland, September 13-15, 2017.

[2] Katsiavalos, A. (2016). DMUN: A Textual Interface for Content-Based Music Information Retrieval in the C@ merata task for MediaEval 2016. Proceedings of the MediaEval 2016 Workshop, Hilversum, The Netherlands, October 20-21, 2016.

[3] Katsiavalos, A., & Collins, T. (2015). DMUN at the MediaEval 2015 C@merata Task: The Stravinsqi Algorithm. Proceedings of the MediaEval 2015 Workshop, Dresden, Germany, September 14-15 2015.