A Description of the CLAS system at C@merata 2014

Stephen Wan
CSIRO
Sydney, Australia
Stephen.Wan@csiro.au

Abstract

This paper describes the CLAS system at C@merata 2014. Given a natural language query in the domain of music theory, the system retrieves matching passages from a musical score. The system uses a domain-specific parser to interpret the query which is mapped to a representation of search features. Candidate answers are then generated using feature unification. Performance on this task was encouraging with \(0.76\) precision and \(0.96\) recall.

We found that the system works well on queries for notes but performance diminishes for queries that consider chords.

1 Introduction

This paper describes the CLAS system which selects processes and retrieves potentially relevant answers from structured data given a natural language query. In this work, the queries and the structured data are in the domain of music theory, as defined by the C@merata 2014 task [11]. The CLAS system produces candidate answers by selecting passages from an XML-ised musical score. Answers may be any consecutive time points spanning multiple whole and partial bars.

For example, a query “4 crotchets” should retrieve any sequence of four consecutive elements in the score where each element is a note and each note has the time duration of a crotchet (one quarter of a whole note). In such a system, expert knowledge is needed to interpret the query. However, this not just limited to definitions of musical concepts (e.g., “crotchet”). For example, the query “4 crotchets” should be interpreted not just as any four notes with crotchet duration within the music (compare this to a general knowledge query “4 composers” requiring any four musical composers to be provided) but specifically four notes in sequence. Furthermore, these four notes would typically be expected to be in the same voice or part; for example, if it were a piano score for two hands, the four crotchets might be a sequence written in the treble clef, played by the right hand.

The C@merata retrieval task differs from document retrieval (for an overview, see [10]) in that the retrieved objects are not text passages (phrase, passages or documents) that are easily indexed based on content-bearing tokens. Thus, approaches that rely on an overlap of content-bearing tokens between the input query terms and the output answer are not applicable here. The required answers are not documents but portions of score. Furthermore, the answers may be based on different granularities of data, depending on the query. For example, the answer passage may be chosen by considering notes, chords, or stylistic properties of the score.

This retrieval task also differs from traditional information retrieval in that each answer object is not considered independently as a candidate answer. Music elements in the score must be viewed in context of each other. For example, when a matching a note as a candidate answer the system must consider its function within the wider context of the surrounding notes, harmonically, melodically and rhythmically.

A structured representation of the score encodes an organisation of notes into this context of musical, harmonic and rhythmic structures. Figure 1 presents the XML representation of this information in the MusicXML standard\(^1\). Although there has been much work in XML retrieval (for example, see the Initiative for the Evaluation of XML (INEX) series starting with [2]) these approaches are also not immediately amenable to the C@merata task. For example, certain queries expressing relationships between musical notes may not easily mapped to structure query languages

\(^1\)http://www.musicxml.com
like XPath\(^2\) expressions, which are often used to select portions of the XML tree structure as an answer (for an example of system using such a language, see [12]). Furthermore, XPath returns a set of distinct references to XML subtrees, however, answers in this task may not be neatly represented in this form, given the relationship between musical objects in the score.

Handling the inherent structure of the input query must do more than represent linguistic aspects such as syntax, for example using methods from question answering research (for an overview on domain-specific work, see [8]). While syntactic relationships might help with text-based question answering, a further mapping to a deeper level semantics is required for this task. Such processing will require a mapping from the query terms to musical concepts including pitch, musical expression, sequential (rhythmic) order, voice, to name but a few.

In this paper, we describe a system that processes the input query, mapping from words in English to music metadata corresponding to the search criteria, or features, represented as a set of attribute-value pairs. The MusicXML is processed using the Music21 library\(^3\) and an exhaustive search of a score is performed, note by note, for candidate answers using feature unification.

This system achieved an overall performance of 0.76 precision and 0.96 recall. The remainder of the paper outlines the system in more detail and presents the C@merata evaluation results.

2 Related Work

The C@merata task is similar to the retrieval of structured data which has been explored through the Initiative for the Evaluation of XML (INEX) retrieval research series of shared tasks (for the first task, see [2]; for an overview, see [7]). Of relevance to this task are the INEX Content and Structure queries, in which the XML structure must be taken into account. Researchers have also examined the use of Natural Language queries with XML retrieval (for example, [12] and [13]). As the C@merata task answers are timestamped tuples denoting the start and end of the music passage matching the query, rather than operating on the raw XML, in this work, we transform the XML into an internal representation of notes which can

\(^2\)http://www.w3schools.com/XPath
\(^3\)http://web.mit.edu/music21
then be checked for sequences matching the query. There are different families of methods for handling the syntactically well-formed input query and mapping them to a semantic representation suitable for finding candidate answers. Given the domain specificity of the data set, one might conceivably draw on related work on Controlled Natural Languages (CNL) (for an overview, see [6]). One approach to the problem might be to define a CNL to capture the sublanguage used to refer to “specialized knowledge about a restricted semantic domain” ([4], cited in [6]).

Examples of related CNL work include BioQuery-CNL [1], a language defining accepted biomedical queries. In this work, an example query is “What are the genes that are targeted by all the drugs that belong to the category Hmg-coa reductase?” (example taken from [6]). Another example of a related CNL is the “CLEF Query Language”, which defines queries to electronic health records [3]. An example is “For all patients with cancer of the pancreas, what is the percentage alive at five years for those who had a course of gemcitabine?” (example again taken from [6]). In this work, we do not define a CNL to represent the acceptable queries, though we do draw on conventions for noun phrases in the music theory sublanguage. Our approach focuses on flexibility of handling a diverse range of queries, without being overly prescriptive about the grammar for such queries.

In the C@merata task, queries are not questions but referring expressions used to denote spans of notes. As such, relationships between multiple notes are often an aspect of the query. For example, the query “C# followed by a D” refers to two notes in a sequence, a “rising 5th from a C” denotes the two notes C and G above. Computational models of how one may refer to multiple objects have been developed in the field of Natural Language Generation (for an overview, see [9]), particularly in the subfield of Referring Expression Generation (REG). For example, [5] present models accounting for relationships between objects, for example the relationship “next to”. Although REG algorithms compute the optimal description that might be used to specify a series of objects, such as a note passage, the linguistic models used with such algorithms may also shed light on the relevant sublanguage for music theory. In this work, we build a bespoke process for musical referring expressions and leave a comprehensive parser informed by REG research to future work.

3 General Approach

The CLAS system interprets the natural language query (NLQ) to find candidate answer passages from the score. In the proposed system, there are four stages:

1. Transform Natural Language Query into a Concept Representation
2. Examine the Concept Representation for the answer scope
3. Transform Concept Representation into Query Representation
4. Match Query Representation to Scoped Data

Briefly, the system: (1) preprocesses tokens and maps these to concepts, resulting in a list of concepts, or the Concept Representation (CR). It then (2) scans the CR, consuming concepts if they define the scope of the answer. The system then (3) parses the remaining CR list to construct the Query Representation (QR), using handwritten parsing rules which implicitly capture the domain-specific interpretation of the NLQ. Generally, the QR is a sequence of feature structures that indicate the type of answer required. Finally, in (4) the QR is then compared with Scoped Data (SD), subset of the data in the XML, represented as a list of Feature Structures (FS), from which candidate answers can be found using feature unification.

4 Mapping query terms to concepts

The system uses a handcrafted lexicon that maps from NLQ terms to a concept in the music theory domain, using the following five steps.

1. Detect multi-word entities;
2. Separate compound words;
3. Handle quotations;
4. Tokenise; and
5. Convert to concepts

In Step 1, multi-word entities such as “down bow” are mapped to a single token “down_bow” to allow correct tokenisation. A pre-defined list of multi-word entities is used to perform this step. Other examples of such phrases include “first inversion” and “second inversion”.

In Step 2, compound tokens which should be treated as separate tokens are divided into their components. For example, “Vb”, denoting the dominant chord (“V”) in the first inversion (“b”),
are separated into the two parts. Other examples include the sharp and flat sign. Notes with a specified octave, such as “C4” are normalised as “C” and “-4”. The decision as to which concepts to treat as a multi-word phrasal units as opposed to distinct words with composite semantics is based on domain modelling decisions captured within the lexicon; concepts represented in the lexicon are designed to be explicitly handled by later search mechanisms.

Out-of-domain words, which may be drawn from the full vocabulary of natural language, are possible in the query but are not included in the lexicon. Such words are accepted if they occur within quotation marks. In this case, they are assumed to either be lyrics or the name of some part. In Step 3, all words between double quotation marks are marked with metadata for these two cases. If the word “lyric” or “word” appears in the query, then the metadata is set to “note:lyric”. Otherwise, the metadata is set to “data:voice”.

In Step 4, tokens are separated using whitespace as a delimiter. Other punctuation is ignored.

Finally, in Step 5, tokens are mapped to their conceptual form using the lexicon. A sample of the lexicon is presented in Figure 2. A CR is made of three parts: a musical object, an attribute type and a value. For example, the word “crotchet” is represented as “_note:length.1”, indicating that the word relates to a “note” FS type, where the feature “length” takes the value “1”. Similarly, the word “quarter” (as in “quarter note”) is also mapped to this sense “_note:length.1”. The word “note” is mapped to “_note:class.NOTE” indicating its class; this is essentially the head noun within the noun phrase.

Note that parts-of-speech are not explicitly encoded. Although “crotchet” would syntactically be a noun, it is simply represented as a note object and used to specify the property of time duration.

Non-contentful words that are not used to construct the QR (e.g., the article “a” or prepositions such as “on”) are mapped to a null token and are ignored. If a token is not found in the lexicon, a simple morphological rule is applied to consider plurals by stripping “s” to the end of the unmatched token, if the letter is present.

Words can have multiple meanings. For example, the word “perfect” is mapped to “_sequence:int_quality.PERFECT;_chord_sequence:cadence.PERFECT”, indicating

Figure 2: A sample of lexical entries.
Query: “2 dotted crotchets on a D#”

1. Detect multi-word entities
   Query: “2 dotted crotchets on a D#”

2. Separate compound words
   Query: “2 dotted crotchets on a D -#”

3. Handle quotations
   Query: “2 dotted crotchets on a D -#”

4. Tokenise
   Tokens: ['2', 'dotted', 'crotchets', 'on', 'a', 'D', '-#']

5. Convert to concepts
   Conceptual Representation:
   ['_sequence:cardinal.2',
    '_note:length_modifier.1-5',
    '_note:length.1',
    '_note:name.D',
    '_note:accidental.SHARP']

Figure 3: Transforming a query into a conceptual representation.

There are two senses: one referring to the quality of an interval (e.g., “a perfect fifth”), or a type of chord sequence (e.g., “a perfect cadence”).

Consider they query “2 dotted crotchets on a D#”. Figure 3 presents the results of each preprocessing stage. The “D#” is separated into the two components. The functions words “on” and “a” are deleted. Finally, since the plural “crotchets” does not exist in the lexicon, the singular form “crotchet” is considered.

5 Building a Query Representation (QR)

The system attributes each NLQ with a type T specifying the type of answer required and the scope of the XML data to be examined for an answer (i.e., the SD). In this work, we defined four types: (i) harmonic, (ii) cadence, (iii) style; and (iv) note. Each type specifies rules for: (1) converting from the XML representation into an SD; (2) parsing rules to convert the CR into a QR; and (3) candidate generation rules.

5.1 Building Scoped Data

A scan of the CR is used to determine the type T by searching for concepts specifying the data “granularity”. If any are found, these are removed from CR and used to set the type. For example, “simultaneous”, as in “simultaneous second” (referring to an interval of a second where both notes are sounded concurrently), is mapped to the concept “_data:granularity.HARMONIC”, indicating the harmonic type. In this case, the SD is defined as a list of chordal notes, taken from a block chord view of the score.4

The cadence and style types also scope the data as a list of chords. If no other type is indicated by a concept in CR, the default note type is used which defines the SD is the concatenation of the sequence of notes in each voice.

For queries where the voice or clef is specified, for example “treble clef” or “soprano part”, the corresponding concepts are used to filter the data to include just that voice.

5.2 QR Parsing

The remaining tokens in CR are used to create a list of FSs of type T following a bespoke rule-based parsing process. The CR is processed in reverse order (assuming head-final noun phrases) and FSs are constructed in a process loosely based on reduction in a shift-reduce parser, where the CR is assumed to be the result of a series of shift operations in which conceptual representations are placed on a stack. The reduction step creates a feature structure of a type based on the the conceptual representation for a token.

Figure 4 presents an example of the resulting feature structures. The concepts “[_note:name.D, _note:length.1]” are consumed first and used to populate a FS. At this point, the “_note:length.1” concept is encountered. Because the current FS already has a note length value (a “minim”), the FS is popped off and pushed onto the QR list. A new FS is then used to consume the remaining tokens: “[_note:name.C, _note:accidental.SHARP, _note:length.1]”. The CR is now empty and the QR is a list of two FSs corresponding to the notes. The length property is scaled using a default function provided in the C@merata task which chooses an appropriate time duration granularity for the answer.

For the example in Figure 3, the “dotted” property applies a 1.5 multiplier to the existing time duration in the feature structure, which in this case

4The method chordify from the music21 package (http://web.mit.edu/music21/) is used to produce this view.
Query: “A C sharp crotchet and a D minim”

Conceptual Representation:

```
[note:name:C, 
  accidental:SHARP, 
  length:1, 
  name:D, 
  length:2]
```

Resulting Query Representation:

```
[
  [class:note, 
    name:C, 
    length:1]
  [class:note, 
    name:D, 
    length:2]
]
```

Figure 4: Parsing “A C# crotchet and a D minim”.

Query: “2 dotted crotchets on a D#”

Resulting Query Representation:

```
[
  [class:note, 
    name:D, 
    accidental:sharp, 
    length:1.5]
  [class:note, 
    name:D, 
    accidental:sharp, 
    length:1.5]
]
```

Figure 5: Parsing “2 dotted crotchets on a D#”.

has the value 1 (“crotchet”). The cardinal number “2” is used to duplicate the feature structure for the note. The output is presented in Figure 5.

Parsing works similarly for the other types. For example, cadences are sequences of chord FSs.

6 Generating candidate answer passages

Once a QR is generated, the SD sequence is then iterated through and at each position a match to the first FS of the QR is attempted using feature unification. If a match is found for the first element of the QR, the matches for the remainder of the QR are attempted. If a match for the entire QR is found, then a candidate answer passage is stored. Figure 6 presents an example of what the SD would look like for the note scope for the music snippet in Figure 1.

For style answers, a different process is used based on simple heuristics that examines patterns of note movement in multiple voices once they are collapsed to a chordal representation of the score. For example, homophony and polyphony processes consider chord notes for note duration, with a mechanism for handling for “passing” notes as indicated by implicit ties. If all notes do not share the same pattern of duration and ties, then the passage is deemed to be polyphonic. Consequently, the QR for this type is an empty list since no feature unification takes place.

For monophony, sequences of chords with 1 note are collected, once the score has been transformed into block chords. For accompaniment, the number of parts in the score is counted.

7 Evaluation

Performance for this system is encouraging. The overall results are presented in Table 1, which lists the recall and precision for answers at two granularities of answers: the correct bars and also the correct beats. Considering the hand-crafted lexicon and the bespoke parsing mechanism, the system performs reasonably well with precision at both granularity answer types, with precision around 0.7 and recall at around 0.9. At the time of writing, the average performance of systems participating in the C@merata task is not available.

In Table 2, we present an analysis of performance according to the type of query. The system does well with queries related to the properties of notes in a sequence, similar to the ones presented...
Table 1: Overall results.

<table>
<thead>
<tr>
<th>Ans.Type</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beat</td>
<td>0.713</td>
<td>0.904</td>
</tr>
<tr>
<td>Bar</td>
<td>0.764</td>
<td>0.967</td>
</tr>
</tbody>
</table>

in the examples in Figure 3 and 4. For these categories, “simple pitch” (e.g., “G”), “simple length” (e.g., “quarter note rest”), “pitch and length” (e.g., “half note C”), “expression” (e.g., “fermata A natural”), precision and recall > 0.86. Indeed in some cases, recall and precision is 1.0.

Interestingly, the performance in handling out-of-domain tokens differs depending on the type of query. For “lyrics” (e.g., “G on the word “praise”’), another note property, perfect precision and recall is achieved. When used to specific the “stave” (e.g., “dotted half note in the “Violine 1” part”, precision drops to 0.5).

The general approach of creating sequences of feature structures (the “followed by” query type, e.g., “quaver C# followed by crotchet B” performed reasonably, with precision of 0.748 and recall of 0.859 for the beat answer types (performance increases for the bar answer type). From this, we infer that the general assumptions underpinning the way noun phrases about notes are transformed into the query representations using the reduction process performs adequately.

The remaining question types proved more difficult for the CLAS system to answer. The query set for “melodic intervals” (e.g., “falling fourth”) showed precision around 0.6 and recall around 0.8. The CLAS system treated melodic intervals as a pairwise sequence of notes. It may be the case that this feature structure matched too many candidates as shown by the reduced precision.

Similarly, the CLAS system does not fare as well with “harmonic intervals” (e.g., “harmonic major tenth”). For these intervals, two notes at the specified interval are sounded simultaneously. The music score is first turned into chord to find concurrent notes. In this version of the system, notes that tied are considered as potential candidates. This may account for the decrease in precision.

Queries regarding “cadences” (e.g., “perfect cadence”), or sequences of harmonic function, also performed poorly with respect to precision. To determine the harmonic function of concurrently sounding notes, the score is first turned into a chordal form. Using the music21 library, each chord is automatically labelled with a scale degree (from 1-7). This labelling is a potentially error-prone process which may result in spurious candidates being generated.

The “triad” queries (e.g., “Ia triad”) also require the chordal view of score to find chords of a particular type. The music21 library also makes its best attempt to determine the inversion of the chord (permutations of the order of the chord notes). Again this is a potentially error-prone process which may generate an excessive number of candidates.

Finally, queries about “texture” (e.g., “homophony”) are also based on the chordal view of the score. This query type was added to the CLAS system at the last moment and is based on a simple examination of the ties and duration of chord notes.

Table 2: Results by question category.

<table>
<thead>
<tr>
<th>Ans.Type</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple pitch</td>
<td>Beat</td>
<td>0.959</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.982</td>
</tr>
<tr>
<td>Simple length</td>
<td>Beat</td>
<td>0.904</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.915</td>
</tr>
<tr>
<td>Pitch &amp; Length</td>
<td>Beat</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.895</td>
</tr>
<tr>
<td>Expression</td>
<td>Beat</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>1.0</td>
</tr>
<tr>
<td>Lyrics</td>
<td>Beat</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>1.0</td>
</tr>
<tr>
<td>Stave</td>
<td>Beat</td>
<td>0.568</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.568</td>
</tr>
<tr>
<td>Followed by</td>
<td>Beat</td>
<td>0.748</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.83</td>
</tr>
<tr>
<td>Mel. intervals</td>
<td>Beat</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.699</td>
</tr>
<tr>
<td>Harm. intervals</td>
<td>Beat</td>
<td>0.158</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.353</td>
</tr>
<tr>
<td>Cadences</td>
<td>Beat</td>
<td>0.238</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.286</td>
</tr>
<tr>
<td>Triads</td>
<td>Beat</td>
<td>0.348</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.391</td>
</tr>
<tr>
<td>Texture</td>
<td>Beat</td>
<td>0.182</td>
</tr>
<tr>
<td></td>
<td>Bar</td>
<td>0.273</td>
</tr>
</tbody>
</table>
8 Future Work

In this work, time constraints affected the choice of methods for the stages in the CLAS system. For example, the bespoke parsing process to map from the query tokens to the feature structures in the Query Representation. An alternative method might be to create a context-free grammar for the domain sublanguage and to use a tool like NLTK\(^5\) to parse the tokens, resulting in a syntactic parse. This linguistic structure can then be mapped to the feature structures. In future work, we will examine the parsing of noun phrase structures in which the features for matching are propagated up to an appropriate node in the tree. These can then be collected to form the Query Representation.

Finally, instead of enumerating exhaustively through all notes, in future work, we will examine the use of search engines to find candidate starting positions, from which feature unification processes can then start. In this approach, notes might be treated as quasi-documents, allowing them to be indexed by metadata based on musical properties.

9 Conclusion

We described the CLAS system in the C@merata 2014 shared task, which uses a bespoke parser and lexicon used to interpret a music NLQ, and a scoping process to reduce the space for candidate answers. Parsing was performed using a reduce-style process. Matches were performed using feature unification. Performance on this task was encouraging with 0.76 precision and 0.96 recall.

References


\(^5\)http://www.nltk.org/